Package ‘hdme’

November 11, 2019

Type Package

Title High-Dimensional Regression with Measurement Error

Version 0.3.2

Encoding UTF-8

Author Oystein Sorensen

Maintainer Oystein Sorensen <oystein.sorensen.1985@gmail.com>


License GPL-3

LazyData TRUE

RoxygenNote 6.1.1

Imports glmnet (>= 3.0.0), ggplot2 (>= 2.2.1), Rdpack, Rcpp (>= 0.12.15), Rglpk (>= 0.6-1), stats

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 2019-11-11 08:50:02 UTC
R topics documented:

coef.corrected_lasso ............................................. 2
coef.gds .......................................................... 3
coef.gmus .......................................................... 3
corrected_lasso .................................................... 4
cv_corrected_lasso .................................................. 6
cv_gds ............................................................... 7
gds ................................................................. 9
gmus ................................................................. 10
gmu_lasso ............................................................ 12
mus ................................................................. 13
plot.corrected_lasso ................................................ 14
plot.cv_corrected_lasso ............................................. 15
plot.cv_gds .......................................................... 16
plot.gds ............................................................. 16
plot.gmus ........................................................... 17
plot.gmu_lasso ...................................................... 18
print.corrected_lasso ............................................... 19
print.cv_corrected_lasso ............................................ 19
print.cv_gds .......................................................... 20
print.gds ............................................................ 20
print.gmus ........................................................... 21
print.gmu_lasso ...................................................... 21

Index 22

| coef.corrected_lasso | Extract Coefficients of a Corrected Lasso object |

**Description**

Default coef method for a corrected_lasso object.

**Usage**

```r
## 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).
**corrected_lasso**

** 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.  
  Only used when `delta` is a single value.
- `...`  
  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.
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
# 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)
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")
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)
# Measurement error covariance matrix
sigmaUU <- diag(x = 0.2, nrow = p, ncol = p)
# Measurement matrix (this is the one we observe)
W <- X + rnorm(n, sd = diag(sigmaUU))
logit <- function(x) (1+exp(-x))^(<-1)
# Response
y <- rbinom(n, size = 1, prob = logit(X %*% c(rep(5, 5), rep(0, p-5))))
fit <- corrected_lasso(W, y, sigmaUU, family = "binomial")
plot(fit)
coef(fit)

cv_corrected_lasso  Cross-validated 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 algorithm 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

# 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)

---

cv_gds

Cross-Validated Generalized Dantzig Selector

Description

Generalized Dantzig Selector with cross-validation.

Usage

cv_gds(X, y, family = "gaussian", no_lambda = 10, lambda = NULL,  
n_folds = 5, weights = rep(1, length(y)))

Arguments

X  Design matrix.

y  Vector of the continuous response value.
family
Use "gaussian" for linear regression, "binomial" for logistic regression and "poisson" for Poisson regression.
	no_lambda
Length of the vector lambda of regularization parameters. Note that if lambda is not provided, the actual number of values might differ slightly, due to the algorithm used by glmnet::glmnet in finding a grid of lambda values.

lambda
Regularization parameter. If not supplied and if no_lambda > 1, a sequence of no_lambda regularization parameters is computed with glmnet::glmnet. If no_lambda = 1 then the cross-validated optimum for the lasso is computed using glmnet::cv.glmnet.

n_folds
Number of cross-validation folds to use.

weights
A vector of weights for each row of X. Defaults to 1 per observation.

Details
Cross-validation loss is calculated as the deviance of the model divided by the number of observations. For the Gaussian case, this is the mean squared error. Weights supplied through the weights argument are used both in fitting the models and when evaluating the test set deviance.

Value
An object of class cv_gds.

References


Examples
## Not run:
# 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
cv_fit <- cv_gds(X, y, family = "binomial", no_lambda = 50, n_folds = 10)
print(cv_fit)
plot(cv_fit)

# Now fit a single GDS at the optimum lambda value determined by cross-validation
fit <- gds(X, y, lambda = cv_fit$lambda_min, family = "binomial")
plot(fit)

# Compare this to the fit for which lambda is selected by GDS
# This automatic selection is performed by glmnet::cv.glmnet, for
# the sake of speed
fit2 <- gds(X, y, family = "binomial")

The following plot compares the two fits.
library(ggplot2)
library(tidyr)
df <- data.frame(fit = fit$beta, fit2 = fit2$beta, index = seq(1, p, by = 1))
ggplot(gather(df, key = "Model", value = "Coefficient", -index),
       aes(x = index, y = Coefficient, color = Model)) +
       geom_point() +
       theme(legend.title = element_blank())

## End(Not run)

---

**gds**

*Generalized Dantzig Selector*

**Description**

Generalized Dantzig Selector

**Usage**

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Design matrix.</td>
</tr>
<tr>
<td>y</td>
<td>Vector of the continuous response value.</td>
</tr>
<tr>
<td>lambda</td>
<td>Regularization parameter. Only a single value is supported.</td>
</tr>
<tr>
<td>family</td>
<td>Use &quot;gaussian&quot; for linear regression, &quot;binomial&quot; for logistic regression and &quot;poisson&quot; for Poisson regression.</td>
</tr>
<tr>
<td>weights</td>
<td>A vector of weights for each row of X.</td>
</tr>
</tbody>
</table>

**Value**

Intercept and coefficients at the values of lambda specified.

**References**


Examples

# 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

```r
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 <- gmus(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 <- gmus(W, y, delta = 0.08)
# Plot the coefficients
plot(fit2)
coef(fit2)
coef(fit2, all = TRUE)

---

**gmui_lasso**  
*Generalized Matrix Uncertainty Lasso*

**Description**

Generalized Matrix Uncertainty Lasso

**Usage**

```r
gmui_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 "gmui_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**

```r
mus(W, y, lambda = NULL, delta = 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 `gmus(W,y,lambda,delta,family = "gaussian")`. 
Value

An object of class "gmus".

References


Examples

# 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

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)
```

Description

Plot the output of `cv_corrected_lasso`.

Usage

```r
## 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).

Description

Plot the output of `cv_gds`.

Usage

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

Arguments

x  The object to be plotted, returned from `cv_gds`.

... Other arguments to plot (not used).

Description

Plot the estimates returned by `gds`.

Usage

```r
## 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)
```

---

### plot.gmus

**Plot the estimates returned by gmus and mus**

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)

---

**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**

## S3 method for class 'gmu_lasso'

```r
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))
```
```r
print.corrected_lasso

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

print.corrected_lasso  \textit{Print a Corrected Lasso object}

\textbf{Description}

Default print method for a \texttt{corrected_lasso} object.

\textbf{Usage}

\begin{verbatim}
## S3 method for class 'corrected_lasso'
print(x, ...)
\end{verbatim}

\textbf{Arguments}

- \texttt{x} Fitted model object returned by \texttt{corrected_lasso}.
- \texttt{...} Other arguments (not used).

print.cv_corrected_lasso

\textit{Print a Cross-Validated Corrected Lasso object}

\textbf{Description}

Default print method for a \texttt{cv_corrected_lasso} object.

\textbf{Usage}

\begin{verbatim}
## S3 method for class 'cv_corrected_lasso'
print(x, ...)
\end{verbatim}

\textbf{Arguments}

- \texttt{x} Fitted model object returned by \texttt{cv_corrected_lasso}.
- \texttt{...} Other arguments (not used).
print.cv_gds  
Print a Cross-Validated GDS Object

Description

Default print method for a cv_gds object.

Usage

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

Arguments

- `x`  
  Fitted model object returned by `cv_gds`.

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

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

c coef.corrected_lasso, 2
c coef.gds, 3
c coef.gmu_lasso, 4
c coef.gmus, 3
c corrected_lasso, 2, 4, 19
c cv_corrected_lasso, 6, 15, 16, 19
c cv_gds, 7, 16, 20
gds, 3, 9, 20
gmu_lasso, 4, 12, 21
gmus, 3, 10, 21

mus, 13

plot.corrected_lasso, 14
plot.cv_corrected_lasso, 15
plot.cv_gds, 16
plot.gds, 16
plot.gmu_lasso, 18
plot.gmus, 17
print.corrected_lasso, 19
print.cv_corrected_lasso, 19
print.cv_gds, 20
print.gds, 20
print.gmu_lasso, 21
print.gmus, 21