Package 'srlars'

July 2, 2026

Type Package

Title Fast and Scalable Cellwise-Robust Ensemble

Version 3.0.1

Date 2026-07-01

Maintainer Anthony Christidis <anthony.christidis@stat.ubc.ca>

Description Functions to perform robust variable selection and regression using the Fast and Scalable Cellwise-Robust Ensemble (FSCRE) algorithm. The approach establishes a robust foundation using the Detect Deviating Cells (DDC) algorithm and robust correlation estimates. It then employs a competitive ensemble architecture where a robust Least Angle Regression (LARS) engine proposes candidate variables and cross-validation arbitrates their assignment. A final robust MM-estimator is applied to the selected predictors.

License GPL (>= 2)

Encoding UTF-8

Biarch true

Imports cellWise, robustbase, mvnfast

Suggests testthat

RoxygenNote 7.3.3

NeedsCompilation no

Author Anthony Christidis [aut, cre],
Gabriela Cohen-Freue [aut]

Repository CRAN

Date/Publication 2026-07-02 02:30:02 UTC

Contents

coef.srlars 2
predict.srlars 4
srlars 6

Index **9**

`coef.srlars` *Coefficients for srlars Object*

Description

`coef.srlars` returns the averaged coefficients for a srlars object.

Usage

```
## S3 method for class 'srlars'
coef(object, model_index = NULL, ...)
```

Arguments

`object`	An object of class srlars.
`model_index`	Indices of the sub-models to include in the ensemble average. Default is NULL, which includes all models.
`...`	Additional arguments for compatibility.

Value

A numeric vector containing the averaged intercept (first element) and slope coefficients.

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

See Also

`srlars`, `predict.srlars`

Examples

```
# Required libraries
library(mvnfast)
library(cellWise)
library(robustbase)

# Simulation parameters
n <- 50
p <- 100
rho.within <- 0.8
rho.between <- 0.2
p.active <- 20
group.size <- 5
snr <- 3
contamination.prop <- 0.1
```

```
# Setting the seed
set.seed(0)

# Block correlation structure
sigma.mat <- matrix(0, p, p)
sigma.mat[1:p.active, 1:p.active] <- rho.between
for(group in 0:(p.active/group.size - 1))
  sigma.mat[(group*group.size+1):(group*group.size+group.size),
  (group*group.size+1):(group*group.size+group.size)] <- rho.within
diag(sigma.mat) <- 1

# Simulation of beta vector
true.beta <- c(runif(p.active, 0, 5)*(-1)^rbinom(p.active, 1, 0.7), rep(0, p - p.active))

# Setting the SD of the variance
sigma <- as.numeric(sqrt(t(true.beta) %*% sigma.mat %*% true.beta)/sqrt(snr))

# Simulation of uncontaminated data
x <- mvnfast::rmvn(n, mu = rep(0, p), sigma = sigma.mat)
colnames(x) <- paste0("V", 1:p)
y <- x %*% true.beta + rnorm(n, 0, sigma)

# Cellwise contamination
contamination_indices <- sample(1:(n * p), round(n * p * contamination.prop))
x_train <- x
x_train[contamination_indices] <- runif(length(contamination_indices), -10, 10)

# FSCRE Ensemble model
ensemble_fit <- srlars(x_train, y,
                       n_models = 5,
                       tolerance = 1e-4,
                       x_preprocess = "ddc",
                       y_preprocess = "wrap",
                       cor_estimator = "wrap",
                       cv_preprocess = "global",
                       cv_fit = "ls",
                       cv_loss = "huber",
                       compute_coef = TRUE)

# Ensemble coefficients
# Default: Average over all models
ensemble_coefs <- coef(ensemble_fit)

# Sensitivity (Recall)
active_selected <- which(ensemble_coefs[-1] != 0)
true_active <- which(true.beta != 0)
recall <- length(intersect(active_selected, true_active)) / length(true_active)
print(paste("Recall:", recall))

# Precision
if(length(active_selected) > 0){
  precision <- length(intersect(active_selected, true_active)) / length(active_selected)
} else {
```

```
  precision <- 0
}
print(paste("Precision:", precision))
```

`predict.srlars` *Predictions for srlars Object*

Description

`predict.srlars` returns the predictions for a srlars object.

Usage

```
## S3 method for class 'srlars'
predict(object, newx, model_index = NULL, dynamic = TRUE, ...)
```

Arguments

`object`	An object of class srlars.
`newx`	New data matrix for predictions.
`model_index`	Indices of the sub-models to include in the ensemble. Default is NULL (all models).
`dynamic`	Logical. If TRUE, and the model was trained robustly, the new data `newx` is cleaned using `DDCpredict` before prediction. This ensures consistency with the robust training phase. Default is TRUE.
`...`	Additional arguments for compatibility.

Value

A numeric vector of predictions.

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

See Also

`srlars`

Examples

```
# Required libraries
library(mvnfast)
library(cellWise)
library(robustbase)

# Simulation parameters
n <- 50
p <- 100
rho.within <- 0.8
rho.between <- 0.2
p.active <- 20
group.size <- 5
snr <- 3
contamination.prop <- 0.1

# Setting the seed
set.seed(0)

# Block correlation structure
sigma.mat <- matrix(0, p, p)
sigma.mat[1:p.active, 1:p.active] <- rho.between
for(group in 0:(p.active/group.size - 1))
  sigma.mat[(group*group.size+1):(group*group.size+group.size),
  (group*group.size+1):(group*group.size+group.size)] <- rho.within
diag(sigma.mat) <- 1

# Simulation of beta vector
true.beta <- c(runif(p.active, 0, 5)*(-1)^rbinom(p.active, 1, 0.7), rep(0, p - p.active))

# Setting the SD of the variance
sigma <- as.numeric(sqrt(t(true.beta) %*% sigma.mat %*% true.beta)/sqrt(snr))

# Simulation of uncontaminated data
x <- mvnfast::rmvn(n, mu = rep(0, p), sigma = sigma.mat)
colnames(x) <- paste0("V", 1:p)
y <- x %*% true.beta + rnorm(n, 0, sigma)

# Cellwise contamination
contamination_indices <- sample(1:(n * p), round(n * p * contamination.prop))
x_train <- x
x_train[contamination_indices] <- runif(length(contamination_indices), -10, 10)

# FSCRE Ensemble model
ensemble_fit <- srlars(x_train, y,
                       n_models = 5,
                       tolerance = 1e-4,
                       x_preprocess = "ddc",
                       y_preprocess = "wrap",
                       cor_estimator = "wrap",
                       cv_preprocess = "global",
                       cv_fit = "ls",
```

```
                        cv_loss = "huber",
                        compute_coef = TRUE)

# Generate Test Data
x_test <- mvnfast::rmvn(50, mu = rep(0, p), sigma = sigma.mat)
colnames(x_test) <- paste0("V", 1:p)
y_test <- x_test %*% true.beta + rnorm(50, 0, sigma)

# Predict on Test Data
preds <- predict(ensemble_fit, x_test)

# Calculate MSPE
mspe <- mean((y_test - preds)^2)
print(paste("MSPE:", mspe))
```

`srlars` *Fast and Scalable Cellwise-Robust Ensemble (FSCRE)*

Description

`srlars` performs the FSCRE algorithm for robust variable selection and regression.

Usage

```
srlars(
  x,
  y,
  n_models = 5,
  tolerance = 1e-08,
  max_predictors = NULL,
  x_preprocess = c("ddc", "none"),
  y_preprocess = c("wrap", "robust_z", "none"),
  cor_estimator = c("wrap", "pearson"),
  cv_preprocess = c("global", "foldwise"),
  cv_fit = c("huber", "ls"),
  cv_loss = c("huber", "trimmed", "mse"),
  cv_folds = 5,
  compute_coef = TRUE
)
```

Arguments

`x`	Design matrix (n x p).
`y`	Response vector (n x 1).
`n_models`	Number of models in the ensemble (K). Default is 5.
`tolerance`	Relative improvement tolerance for stopping (tau). Default is 1e-8.

max_predictors	Maximum total number of variables to select across all models. Default is n * n_models.
x_preprocess	Character. "ddc" (default) for cellwise cleaning, or "none".
y_preprocess	Character. "wrap" (default) for univariate robustification, "robust_z", or "none".
cor_estimator	Character. "wrap" (default) for robust PSD correlation, or "pearson".
cv_preprocess	Character. "global" (default) or "foldwise" (to prevent data leakage).
cv_fit	Character. "huber" (default) or "ls" for the inner arbiter fitting method.
cv_loss	Character. "huber" (default), "trimmed", or "mse" for arbiter scoring.
cv_folds	Integer. Number of cross-validation folds. Default is 5.
compute_coef	Logical. If TRUE, fits the final robust MM-models. Default is TRUE.

Value

An object of class `srlars` containing the selected variables and coefficients.

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

See Also

`coef.srlars`, `predict.srlars`

Examples

```
# Required libraries
library(mvnfast)
library(cellWise)
library(robustbase)

# Simulation parameters
n <- 50
p <- 100
rho.within <- 0.8
rho.between <- 0.2
p.active <- 20
group.size <- 5
snr <- 3
contamination.prop <- 0.1

# Setting the seed
set.seed(0)

# Block correlation structure
sigma.mat <- matrix(0, p, p)
sigma.mat[1:p.active, 1:p.active] <- rho.between
for(group in 0:(p.active/group.size - 1))
  sigma.mat[(group*group.size+1):(group*group.size+group.size),
  (group*group.size+1):(group*group.size+group.size)] <- rho.within
```

```
diag(sigma.mat) <- 1

# Simulation of beta vector
true.beta <- c(runif(p.active, 0, 5)*(-1)^rbinom(p.active, 1, 0.7), rep(0, p - p.active))

# Setting the SD of the variance
sigma <- as.numeric(sqrt(t(true.beta) %*% sigma.mat %*% true.beta)/sqrt(snr))

# Simulation of uncontaminated data
x <- mvnfast::rmvn(n, mu = rep(0, p), sigma = sigma.mat)
colnames(x) <- paste0("V", 1:p)
y <- x %*% true.beta + rnorm(n, 0, sigma)

# Cellwise contamination
contamination_indices <- sample(1:(n * p), round(n * p * contamination.prop))
x_train <- x
x_train[contamination_indices] <- runif(length(contamination_indices), -10, 10)

# FSCRE Ensemble model
ensemble_fit <- srlars(x_train, y,
                       n_models = 5,
                       tolerance = 1e-4,
                       x_preprocess = "ddc",
                       y_preprocess = "wrap",
                       cor_estimator = "wrap",
                       cv_preprocess = "global",
                       cv_fit = "huber",
                       cv_loss = "huber",
                       compute_coef = TRUE)

# Check selected variables
print(ensemble_fit$active.sets)
```

Index

coef.srlars, 2, *7*

DDCpredict, *4*

predict.srlars, *2*, 4, *7*

srlars, *2*, *4*, 6