

# Package ‘RMSS’

July 21, 2025

**Type** Package

**Title** Robust Multi-Model Subset Selection

**Version** 1.2.0

**Date** 2025-07-17

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

## Description

Efficient algorithms for generating ensembles of robust, sparse and diverse models via robust multi-model subset selection (RMSS). The robust ensembles are generated by minimizing the sum of the least trimmed square loss of the models in the ensembles under constraints for the size of the models and the sharing of the predictors. Tuning parameters for the robustness, sparsity and diversity of the robust ensemble are selected by cross-validation.

**License** GPL ( $\geq 2$ )

**Encoding** UTF-8

**Biarch** true

**Imports** Rcpp ( $\geq 1.0.9$ ), srlars, robStepSplitReg, cellWise, robustbase

**Suggests** testthat, mvnfast

**LinkingTo** Rcpp, RcppArmadillo

**RoxygenNote** 7.2.3

**NeedsCompilation** yes

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

**Repository** CRAN

**Date/Publication** 2025-07-17 15:40:13 UTC

## Contents

|                        |   |
|------------------------|---|
| coef.cv.RMSS . . . . . | 2 |
| coef.RMSS . . . . .    | 4 |
| cv.RMSS . . . . .      | 7 |

|                           |    |
|---------------------------|----|
| predict.cv.RMSS . . . . . | 10 |
| predict.RMSS . . . . .    | 12 |
| RMSS . . . . .            | 15 |
| trimmed_samples . . . . . | 17 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>21</b> |
|--------------|-----------|

---

|              |                                        |
|--------------|----------------------------------------|
| coef.cv.RMSS | <i>Coefficients for cv.RMSS Object</i> |
|--------------|----------------------------------------|

---

**Description**

coef.cv.RMSS returns the coefficients for a cv.RMSS object.

**Usage**

```
## S3 method for class 'cv.RMSS'
coef(
  object,
  h_ind = NULL,
  t_ind = NULL,
  u_ind = NULL,
  individual_models = FALSE,
  group_index = NULL,
  ...
)
```

**Arguments**

- object            An object of class cv.RMSS.
- h\_ind            Index for robustness parameter.
- t\_ind            Index for sparsity parameter.
- u\_ind            Index for diversity parameter.
- individual\_models    Argument to determine whether the coefficients of each model are returned. Default is FALSE.
- group\_index       Groups included in the ensemble. Default setting includes all the groups.
- ...              Additional arguments for compatibility.

**Value**

The coefficients for the cv.RMSS object.

**Author(s)**

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

**See Also**[cv.RMSS](#)**Examples**

```

# Simulation parameters
n <- 50
p <- 100
rho <- 0.8
rho.inactive <- 0.2
group.size <- 5
p.active <- 15
snr <- 2
contamination.prop <- 0.3

# Setting the seed
set.seed(0)

# Block Correlation
sigma.mat <- matrix(0, p, p)
sigma.mat[1:p.active, 1:p.active] <- rho.inactive
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
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 test data
m <- 2e3
x_test <- mvnfast::rmvn(m, mu = rep(0, p), sigma = sigma.mat)
y_test <- x_test %*% true.beta + rnorm(m, 0, sigma)

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

# Contamination of data
contamination_indices <- 1:floor(n*contamination.prop)
k_lev <- 2
k_slo <- 100
x_train <- x
y_train <- y
beta_cont <- true.beta
beta_cont[true.beta!=0] <- beta_cont[true.beta!=0]*(1 + k_slo)
beta_cont[true.beta==0] <- k_slo*max(abs(true.beta))
for(cont_id in contamination_indices){

```

```

a <- runif(p, min = -1, max = 1)
a <- a - as.numeric((1/p)*t(a) %>% rep(1, p))
x_train[cont_id,] <- mvnfast::rmvn(1, rep(0, p), 0.1^2*diag(p)) + k_lev * a /
  as.numeric(sqrt(t(a) %>% solve(sigma.mat) %>% a))
y_train[cont_id] <- t(x_train[cont_id,]) %>% beta_cont
}

# CV RMSS
rmss_fit <- cv.RMSS(x = x_train, y = y_train,
  n_models = 3,
  h_grid = c(35), t_grid = c(6, 8, 10), u_grid = c(1:3),
  initial_estimator = "robStepSplitReg",
  tolerance = 1e-1,
  max_iter = 1e3,
  neighborhood_search = FALSE,
  neighborhood_search_tolerance = 1e-1,
  n_folds = 5,
  alpha = 1/4,
  gamma = 1,
  n_threads = 1)
rmss_coefs <- coef(rmss_fit,
  h_ind = rmss_fit$h_opt,
  t_ind = rmss_fit$t_opt,
  u_ind = rmss_fit$u_opt,
  group_index = 1:rmss_fit$n_models)
sens_rmss <- sum(which((rmss_coefs[-1]!=0)) <= p.active)/p.active
spec_rmss <- sum(which((rmss_coefs[-1]!=0)) <= p.active)/sum(rmss_coefs[-1]!=0)
rmss_preds <- predict(rmss_fit, newx = x_test,
  h_ind = rmss_fit$h_opt,
  t_ind = rmss_fit$t_opt,
  u_ind = rmss_fit$u_opt,
  group_index = 1:rmss_fit$n_models,
  dynamic = FALSE)
rmss_mspe <- mean((y_test - rmss_preds)^2)/sigma^2

```

coef.RMSS

*Coefficients for RMSS Object***Description**

coef.RMSS returns the coefficients for a RMSS object.

**Usage**

```

## S3 method for class 'RMSS'
coef(
  object,
  h_ind,

```

```

    t_ind,
    u_ind,
    individual_models = FALSE,
    group_index = NULL,
    ...
)

```

### Arguments

|                   |                                                                                              |
|-------------------|----------------------------------------------------------------------------------------------|
| object            | An object of class RMSS.                                                                     |
| h_ind             | Index for robustness parameter.                                                              |
| t_ind             | Index for sparsity parameter.                                                                |
| u_ind             | Index for diversity parameter.                                                               |
| individual_models | Argument to determine whether the coefficients of each model are returned. Default is FALSE. |
| group_index       | Groups included in the ensemble. Default setting includes all the groups.                    |
| ...               | Additional arguments for compatibility.                                                      |

### Value

The coefficients for the RMSS object.

### Author(s)

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

### See Also

[RMSS](#)

### Examples

```

# Simulation parameters
n <- 50
p <- 100
rho <- 0.8
rho.inactive <- 0.2
group.size <- 5
p.active <- 15
snr <- 2
contamination.prop <- 0.3

# Setting the seed
set.seed(0)

# Block Correlation
sigma.mat <- matrix(0, p, p)
sigma.mat[1:p.active, 1:p.active] <- rho.inactive

```

```

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
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 test data
m <- 2e3
x_test <- mvnfast::rmvn(m, mu = rep(0, p), sigma = sigma.mat)
y_test <- x_test %*% true.beta + rnorm(m, 0, sigma)

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

# Contamination of data
contamination_indices <- 1:floor(n*contamination.prop)
k_lev <- 2
k_slo <- 100
x_train <- x
y_train <- y
beta_cont <- true.beta
beta_cont[true.beta!=0] <- beta_cont[true.beta!=0]*(1 + k_slo)
beta_cont[true.beta==0] <- k_slo*max(abs(true.beta))
for(cont_id in contamination_indices){

  a <- runif(p, min = -1, max = 1)
  a <- a - as.numeric((1/p)*t(a) %*% rep(1, p))
  x_train[cont_id,] <- mvnfast::rmvn(1, rep(0, p), 0.1^2*diag(p)) + k_lev * a /
    as.numeric(sqrt(t(a) %*% solve(sigma.mat) %*% a))
  y_train[cont_id] <- t(x_train[cont_id,]) %*% beta_cont
}

# RMSS
rmss_fit <- RMSS(x = x_train, y = y_train,
                n_models = 3,
                h_grid = c(35), t_grid = c(6, 8, 10), u_grid = c(1:3),
                initial_estimator = "robStepSplitReg",
                tolerance = 1e-1,
                max_iter = 1e3,
                neighborhood_search = FALSE,
                neighborhood_search_tolerance = 1e-1)
rmss_coefs <- coef(rmss_fit,
                  h_ind = 1, t_ind = 2, u_ind = 1,
                  group_index = 1:rmss_fit$n_models)
sens_rmss <- sum(which((rmss_coefs[-1]!=0)) <= p.active)/p.active
spec_rmss <- sum(which((rmss_coefs[-1]!=0)) <= p.active)/sum(rmss_coefs[-1]!=0)

```

```
rmss_preds <- predict(rmss_fit, newx = x_test,
                     h_ind = 1, t_ind = 2, u_ind = 1,
                     group_index = 1:rmss_fit$n_models,
                     dynamic = FALSE)
rmss_mspe <- mean((y_test - rmss_preds)^2)/sigma^2
```

---

cv.RMSS

---

*Cross-Validatoion for Robust Multi-Model Subset Selection*


---

## Description

cv.RMSS performs the cross-validation procedure for robust multi-model subset selection.

## Usage

```
cv.RMSS(
  x,
  y,
  n_models,
  h_grid,
  t_grid,
  u_grid,
  initial_estimator = c("robStepSplitReg", "srlars")[1],
  tolerance = 0.1,
  max_iter = 1000,
  neighborhood_search = FALSE,
  neighborhood_search_tolerance = 0.1,
  cv_criterion = c("tau", "trimmed")[1],
  n_folds = 5,
  alpha = 1/4,
  gamma = 1,
  n_threads = 1
)
```

## Arguments

|                   |                                                                                            |
|-------------------|--------------------------------------------------------------------------------------------|
| x                 | Design matrix.                                                                             |
| y                 | Response vector.                                                                           |
| n_models          | Number of models into which the variables are split.                                       |
| h_grid            | Grid for robustness parameter.                                                             |
| t_grid            | Grid for sparsity parameter.                                                               |
| u_grid            | Grid for diversity parameter.                                                              |
| initial_estimator | Method used for initial estimator. Must be one of "robStepSplitReg" (default) or "srlars". |

|                               |                                                                                                             |
|-------------------------------|-------------------------------------------------------------------------------------------------------------|
| tolerance                     | Tolerance level for convergence of PSBGD algorithm.                                                         |
| max_iter                      | Maximum number of iterations in PSBGD algorithm.                                                            |
| neighborhood_search           | Neighborhood search to improve solution. Default is FALSE.                                                  |
| neighborhood_search_tolerance | Tolerance parameter for neighborhood search. Default is 1e-1.                                               |
| cv_criterion                  | Criterion to use for cross-validation procedure. Must be one of "tau" (default) or "trimmed".               |
| n_folds                       | Number of folds for cross-validation procedure. Default is 5.                                               |
| alpha                         | Proportion of trimmed samples for cross-validation procedure. Default is 1/4.                               |
| gamma                         | Weight parameter for ensemble MSPE (gamma) and average MSPE of individual models (1 - gamma). Default is 1. |
| n_threads                     | Number of threads used by OpenMP for multithreading over the folds. Default is 1.                           |

**Value**

An object of class cv.RMSS

**Author(s)**

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

**See Also**

[coef.cv.RMSS](#), [predict.cv.RMSS](#)

**Examples**

```
# Simulation parameters
n <- 50
p <- 100
rho <- 0.8
rho.inactive <- 0.2
group.size <- 5
p.active <- 15
snr <- 2
contamination.prop <- 0.3

# Setting the seed
set.seed(0)

# Block Correlation
sigma.mat <- matrix(0, p, p)
sigma.mat[1:p.active, 1:p.active] <- rho.inactive
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
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 test data
m <- 2e3
x_test <- mvnfast::rmvn(m, mu = rep(0, p), sigma = sigma.mat)
y_test <- x_test %*% true.beta + rnorm(m, 0, sigma)

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

# Contamination of data
contamination_indices <- 1:floor(n*contamination.prop)
k_lev <- 2
k_slo <- 100
x_train <- x
y_train <- y
beta_cont <- true.beta
beta_cont[true.beta!=0] <- beta_cont[true.beta!=0]*(1 + k_slo)
beta_cont[true.beta==0] <- k_slo*max(abs(true.beta))
for(cont_id in contamination_indices){

  a <- runif(p, min = -1, max = 1)
  a <- a - as.numeric((1/p)*t(a) %*% rep(1, p))
  x_train[cont_id,] <- mvnfast::rmvn(1, rep(0, p), 0.1^2*diag(p)) + k_lev * a /
    as.numeric(sqrt(t(a) %*% solve(sigma.mat) %*% a))
  y_train[cont_id] <- t(x_train[cont_id,]) %*% beta_cont
}

# CV RMSS
rmss_fit <- cv.RMSS(x = x_train, y = y_train,
                  n_models = 3,
                  h_grid = c(35), t_grid = c(6, 8, 10), u_grid = c(1:3),
                  initial_estimator = "robStepSplitReg",
                  tolerance = 1e-1,
                  max_iter = 1e3,
                  neighborhood_search = FALSE,
                  neighborhood_search_tolerance = 1e-1,
                  n_folds = 5,
                  alpha = 1/4,
                  gamma = 1,
                  n_threads = 1)
rmss_coefs <- coef(rmss_fit,
                  h_ind = rmss_fit$h_opt,
                  t_ind = rmss_fit$t_opt,
                  u_ind = rmss_fit$u_opt,
                  group_index = 1:rmss_fit$n_models)

```

```

sens_rmss <- sum(which((rmss_coefs[-1]!=0)) <= p.active)/p.active
spec_rmss <- sum(which((rmss_coefs[-1]!=0)) <= p.active)/sum(rmss_coefs[-1]!=0)
rmss_preds <- predict(rmss_fit, newx = x_test,
                     h_ind = rmss_fit$h_opt,
                     t_ind = rmss_fit$t_opt,
                     u_ind = rmss_fit$u_opt,
                     group_index = 1:rmss_fit$n_models,
                     dynamic = FALSE)
rmss_mspe <- mean((y_test - rmss_preds)^2)/sigma^2

```

---

predict.cv.RMSS

*Predictions for cv.RMSS Object*


---

### Description

predict.cv.RMSS returns the predictions for a cv.RMSS object.

### Usage

```

## S3 method for class 'cv.RMSS'
predict(
  object,
  newx,
  h_ind = NULL,
  t_ind = NULL,
  u_ind = NULL,
  group_index = NULL,
  dynamic = FALSE,
  ...
)

```

### Arguments

|             |                                                                                                        |
|-------------|--------------------------------------------------------------------------------------------------------|
| object      | An object of class cv.RMSS.                                                                            |
| newx        | New data for predictions.                                                                              |
| h_ind       | Index for robustness parameter.                                                                        |
| t_ind       | Index for sparsity parameter.                                                                          |
| u_ind       | Index for diversity parameter.                                                                         |
| group_index | Groups included in the ensemble. Default setting includes all the groups.                              |
| dynamic     | Argument to determine whether dynamic predictions are used based on deviating cells. Default is FALSE. |
| ...         | Additional arguments for compatibility.                                                                |

### Value

The predictions for the cv.RMSS object.

**Author(s)**

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

**See Also**

[cv.RMSS](#)

**Examples**

```
# Simulation parameters
n <- 50
p <- 100
rho <- 0.8
rho.inactive <- 0.2
group.size <- 5
p.active <- 15
snr <- 2
contamination.prop <- 0.3

# Setting the seed
set.seed(0)

# Block Correlation
sigma.mat <- matrix(0, p, p)
sigma.mat[1:p.active, 1:p.active] <- rho.inactive
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
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 test data
m <- 2e3
x_test <- mvnfast::rmvn(m, mu = rep(0, p), sigma = sigma.mat)
y_test <- x_test %*% true.beta + rnorm(m, 0, sigma)

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

# Contamination of data
contamination_indices <- 1:floor(n*contamination.prop)
k_lev <- 2
k_slo <- 100
x_train <- x
y_train <- y
```

```

beta_cont <- true.beta
beta_cont[true.beta!=0] <- beta_cont[true.beta!=0]*(1 + k_slo)
beta_cont[true.beta==0] <- k_slo*max(abs(true.beta))
for(cont_id in contamination_indices){

  a <- runif(p, min = -1, max = 1)
  a <- a - as.numeric((1/p)*t(a) %*% rep(1, p))
  x_train[cont_id,] <- mvnfast::rmvn(1, rep(0, p), 0.1^2*diag(p)) + k_lev * a /
    as.numeric(sqrt(t(a) %*% solve(sigma.mat) %*% a))
  y_train[cont_id] <- t(x_train[cont_id,]) %*% beta_cont
}

# CV RMSS
rmss_fit <- cv.RMSS(x = x_train, y = y_train,
  n_models = 3,
  h_grid = c(35), t_grid = c(6, 8, 10), u_grid = c(1:3),
  initial_estimator = "robStepSplitReg",
  tolerance = 1e-1,
  max_iter = 1e3,
  neighborhood_search = FALSE,
  neighborhood_search_tolerance = 1e-1,
  n_folds = 5,
  alpha = 1/4,
  gamma = 1,
  n_threads = 1)
rmss_coefs <- coef(rmss_fit,
  h_ind = rmss_fit$h_opt,
  t_ind = rmss_fit$t_opt,
  u_ind = rmss_fit$u_opt,
  group_index = 1:rmss_fit$n_models)
sens_rmss <- sum(which((rmss_coefs[-1]!=0)) <= p.active)/p.active
spec_rmss <- sum(which((rmss_coefs[-1]!=0)) <= p.active)/sum(rmss_coefs[-1]!=0)
rmss_preds <- predict(rmss_fit, newx = x_test,
  h_ind = rmss_fit$h_opt,
  t_ind = rmss_fit$t_opt,
  u_ind = rmss_fit$u_opt,
  group_index = 1:rmss_fit$n_models,
  dynamic = FALSE)
rmss_mspe <- mean((y_test - rmss_preds)^2)/sigma^2

```

---

predict.RMSS

*Predictions for RMSS Object*


---

## Description

predict.RMSS returns the predictions for a RMSS object.

**Usage**

```
## S3 method for class 'RMSS'
predict(
  object,
  newx,
  h_ind,
  t_ind,
  u_ind,
  group_index = NULL,
  dynamic = FALSE,
  ...
)
```

**Arguments**

|                          |                                                                                                        |
|--------------------------|--------------------------------------------------------------------------------------------------------|
| <code>object</code>      | An object of class RMSS.                                                                               |
| <code>newx</code>        | New data for predictions.                                                                              |
| <code>h_ind</code>       | Index for robustness parameter.                                                                        |
| <code>t_ind</code>       | Index for sparsity parameter.                                                                          |
| <code>u_ind</code>       | Index for diversity parameter.                                                                         |
| <code>group_index</code> | Groups included in the ensemble. Default setting includes all the groups.                              |
| <code>dynamic</code>     | Argument to determine whether dynamic predictions are used based on deviating cells. Default is FALSE. |
| <code>...</code>         | Additional arguments for compatibility.                                                                |

**Value**

The predictions for the RMSS object.

**Author(s)**

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

**See Also**

[RMSS](#)

**Examples**

```
# Simulation parameters
n <- 50
p <- 100
rho <- 0.8
rho.inactive <- 0.2
group.size <- 5
p.active <- 15
snr <- 2
contamination.prop <- 0.3
```

```

# Setting the seed
set.seed(0)

# Block Correlation
sigma.mat <- matrix(0, p, p)
sigma.mat[1:p.active, 1:p.active] <- rho.inactive
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
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 test data
m <- 2e3
x_test <- mvnfast::rmvn(m, mu = rep(0, p), sigma = sigma.mat)
y_test <- x_test %*% true.beta + rnorm(m, 0, sigma)

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

# Contamination of data
contamination_indices <- 1:floor(n*contamination.prop)
k_lev <- 2
k_slo <- 100
x_train <- x
y_train <- y
beta_cont <- true.beta
beta_cont[true.beta!=0] <- beta_cont[true.beta!=0]*(1 + k_slo)
beta_cont[true.beta==0] <- k_slo*max(abs(true.beta))
for(cont_id in contamination_indices){

  a <- runif(p, min = -1, max = 1)
  a <- a - as.numeric((1/p)*t(a) %*% rep(1, p))
  x_train[cont_id,] <- mvnfast::rmvn(1, rep(0, p), 0.1^2*diag(p)) + k_lev * a /
    as.numeric(sqrt(t(a) %*% solve(sigma.mat) %*% a))
  y_train[cont_id] <- t(x_train[cont_id,]) %*% beta_cont
}

# RMSS
rmss_fit <- RMSS(x = x_train, y = y_train,
                n_models = 3,
                h_grid = c(35), t_grid = c(6, 8, 10), u_grid = c(1:3),
                initial_estimator = "robStepSplitReg",
                tolerance = 1e-1,
                max_iter = 1e3,

```

```

neighborhood_search = FALSE,
neighborhood_search_tolerance = 1e-1)
rmss_coefs <- coef(rmss_fit,
  h_ind = 1, t_ind = 2, u_ind = 1,
  group_index = 1:rmss_fit$n_models)
sens_rmss <- sum(which((rmss_coefs[-1]!=0)) <= p.active)/p.active
spec_rmss <- sum(which((rmss_coefs[-1]!=0)) <= p.active)/sum(rmss_coefs[-1]!=0)
rmss_preds <- predict(rmss_fit, newx = x_test,
  h_ind = 1, t_ind = 2, u_ind = 1,
  group_index = 1:rmss_fit$n_models,
  dynamic = FALSE)
rmss_mspe <- mean((y_test - rmss_preds)^2)/sigma^2

```

---

RMSS

---

*Robust Multi-Model Subset Selection*


---

## Description

RMSS performs robust multi-model subset selection.

## Usage

```

RMSS(
  x,
  y,
  n_models,
  h_grid,
  t_grid,
  u_grid,
  initial_estimator = c("robStepSplitReg", "srlars")[1],
  tolerance = 0.1,
  max_iter = 1000,
  neighborhood_search = FALSE,
  neighborhood_search_tolerance = 0.1
)

```

## Arguments

|                       |                                                      |
|-----------------------|------------------------------------------------------|
| <code>x</code>        | Design matrix.                                       |
| <code>y</code>        | Response vector.                                     |
| <code>n_models</code> | Number of models into which the variables are split. |
| <code>h_grid</code>   | Grid for robustness parameter.                       |
| <code>t_grid</code>   | Grid for sparsity parameter.                         |
| <code>u_grid</code>   | Grid for diversity parameter.                        |

**initial\_estimator** Method used for initial estimator. Must be one of "robStepSplitReg" (default) or "srlars".

**tolerance** Tolerance level for convergence of PSBGD algorithm.

**max\_iter** Maximum number of iterations in PSBGD algorithm.

**neighborhood\_search** Neighborhood search to improve solution. Default is FALSE.

**neighborhood\_search\_tolerance** Tolerance parameter for neighborhood search. Default is 1e-1.

**Value**

An object of class RMSS

**Author(s)**

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

**See Also**

[coef.RMSS](#), [predict.RMSS](#)

**Examples**

```
# Simulation parameters
n <- 50
p <- 100
rho <- 0.8
rho.inactive <- 0.2
group.size <- 5
p.active <- 15
snr <- 2
contamination.prop <- 0.3

# Setting the seed
set.seed(0)

# Block Correlation
sigma.mat <- matrix(0, p, p)
sigma.mat[1:p.active, 1:p.active] <- rho.inactive
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
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 test data
m <- 2e3
x_test <- mvnfast::rmvn(m, mu = rep(0, p), sigma = sigma.mat)
y_test <- x_test %*% true.beta + rnorm(m, 0, sigma)

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

# Contamination of data
contamination_indices <- 1:floor(n*contamination.prop)
k_lev <- 2
k_slo <- 100
x_train <- x
y_train <- y
beta_cont <- true.beta
beta_cont[true.beta!=0] <- beta_cont[true.beta!=0]*(1 + k_slo)
beta_cont[true.beta==0] <- k_slo*max(abs(true.beta))
for(cont_id in contamination_indices){

  a <- runif(p, min = -1, max = 1)
  a <- a - as.numeric((1/p)*t(a) %*% rep(1, p))
  x_train[cont_id,] <- mvnfast::rmvn(1, rep(0, p), 0.1^2*diag(p)) + k_lev * a /
    as.numeric(sqrt(t(a) %*% solve(sigma.mat) %*% a))
  y_train[cont_id] <- t(x_train[cont_id,]) %*% beta_cont
}

# RMSS
rmss_fit <- RMSS(x = x_train, y = y_train,
  n_models = 3,
  h_grid = c(35), t_grid = c(6, 8, 10), u_grid = c(1:3),
  initial_estimator = "robStepSplitReg",
  tolerance = 1e-1,
  max_iter = 1e3,
  neighborhood_search = FALSE,
  neighborhood_search_tolerance = 1e-1)
rmss_coefs <- coef(rmss_fit,
  h_ind = 1, t_ind = 2, u_ind = 1,
  group_index = 1:rmss_fit$n_models)
sens_rmss <- sum(which((rmss_coefs[-1]!=0)) <= p.active)/p.active
spec_rmss <- sum(which((rmss_coefs[-1]!=0)) <= p.active)/sum(rmss_coefs[-1]!=0)
rmss_preds <- predict(rmss_fit, newx = x_test,
  h_ind = 1, t_ind = 2, u_ind = 1,
  group_index = 1:rmss_fit$n_models,
  dynamic = FALSE)
rmss_mspe <- mean((y_test - rmss_preds)^2)/sigma^2

```

**Description**

`trimmed_samples` returns the coefficients for a RMSS or `cv.RMSS` object.

**Usage**

```
trimmed_samples(
  object,
  h_ind = NULL,
  t_ind = NULL,
  u_ind = NULL,
  group_index = NULL,
  ...
)
```

**Arguments**

|                          |                                                                           |
|--------------------------|---------------------------------------------------------------------------|
| <code>object</code>      | An object of class RMSS                                                   |
| <code>h_ind</code>       | Index for robustness parameter.                                           |
| <code>t_ind</code>       | Index for sparsity parameter.                                             |
| <code>u_ind</code>       | Index for diversity parameter.                                            |
| <code>group_index</code> | Groups included in the ensemble. Default setting includes all the groups. |
| <code>...</code>         | Additional arguments for compatibility.                                   |

**Value**

The trimmed samples for the RMSS or `cv.RMSS` object.

**Author(s)**

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

**See Also**

[RMSS](#)

**Examples**

```
# Simulation parameters
n <- 50
p <- 100
rho <- 0.8
rho.inactive <- 0.2
group.size <- 5
p.active <- 15
snr <- 2
contamination.prop <- 0.3

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

```

# Block Correlation
sigma.mat <- matrix(0, p, p)
sigma.mat[1:p.active, 1:p.active] <- rho.inactive
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
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 test data
m <- 2e3
x_test <- mvnfast::rmvn(m, mu = rep(0, p), sigma = sigma.mat)
y_test <- x_test %*% true.beta + rnorm(m, 0, sigma)

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

# Contamination of data
contamination_indices <- 1:floor(n*contamination.prop)
k_lev <- 2
k_slo <- 100
x_train <- x
y_train <- y
beta_cont <- true.beta
beta_cont[true.beta!=0] <- beta_cont[true.beta!=0]*(1 + k_slo)
beta_cont[true.beta==0] <- k_slo*max(abs(true.beta))
for(cont_id in contamination_indices){
  a <- runif(p, min = -1, max = 1)
  a <- a - as.numeric((1/p)*t(a) %*% rep(1, p))
  x_train[cont_id,] <- mvnfast::rmvn(1, rep(0, p), 0.1^2*diag(p)) + k_lev * a /
    as.numeric(sqrt(t(a) %*% solve(sigma.mat) %*% a))
  y_train[cont_id] <- t(x_train[cont_id,]) %*% beta_cont
}

# RMSS
rmss_fit <- RMSS(x = x_train, y = y_train,
                n_models = 3,
                h_grid = c(35), t_grid = c(6, 8, 10), u_grid = c(1:3),
                tolerance = 1e-1,
                max_iter = 1e3,
                neighborhood_search = FALSE,
                neighborhood_search_tolerance = 1e-1)
rmss_coefs <- coef(rmss_fit,
                  h_ind = 1, t_ind = 2, u_ind = 1,

```

```
      group_index = 1:rmss_fit$n_models)
sens_rmss <- sum(which((rmss_coefs[-1]!=0)) <= p.active)/p.active
spec_rmss <- sum(which((rmss_coefs[-1]!=0)) <= p.active)/sum(rmss_coefs[-1]!=0)
rmss_preds <- predict(rmss_fit, newx = x_test,
                      h_ind = 1, t_ind = 2, u_ind = 1,
                      group_index = 1:rmss_fit$n_models,
                      dynamic = FALSE)
rmss_mspe <- mean((y_test - rmss_preds)^2)/sigma^2
trimmed_id <- trimmed_samples(rmss_fit, h_ind = 1, t_ind = 1, u_ind = 1)
```

# Index

`coef.cv.RMSS`, [2](#), [8](#)  
`coef.RMSS`, [4](#), [16](#)  
`cv.RMSS`, [3](#), [7](#), [11](#)  
  
`predict.cv.RMSS`, [8](#), [10](#)  
`predict.RMSS`, [12](#), [16](#)  
  
`RMSS`, [5](#), [13](#), [15](#), [18](#)  
  
`trimmed_samples`, [17](#)