

Package ‘robustsubsets’

July 13, 2021

Type Package

Title Robust Subset Selection

Version 1.1.0

Author Ryan Thompson

Maintainer Ryan Thompson <ryan.thompson@monash.edu>

Description Provides functionality for robust subset selection in linear regression.

URL <https://github.com/ryan-thompson/robustsubsets>

BugReports <https://github.com/ryan-thompson/robustsubsets/issues>

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

Imports stats,

utils,
gurobi,
parallel,
abind,
ggplot2,
Rcpp

LinkingTo Rcpp,
RcppArmadillo

R topics documented:

coef.cv.rss	2
coef.rss	2
cv.bss	3
cv.rss	4
plot.cv.rss	6
plot.rss	7
predict.cv.rss	7
predict.rss	8
rss	9

Index[12](#)

coef.cv.rss	<i>Coefficient function for cv.rss object</i>
-------------	---

Description

Extracts coefficients for a given parameter pair (k,h).

Usage

```
## S3 method for class 'cv.rss'
coef(object, k = "k.min", h = "h.min", ...)
```

Arguments

object	an object of class rss
k	the number of predictors indexing the desired fit; 'k.min' uses best k from cross-validation
h	the number of observations indexing the desired fit; 'h.min' uses best h from cross-validation
...	any other arguments

Value

An array of coefficients.

Author(s)

Ryan Thompson <ryan.thompson@monash.edu>

coef.rss	<i>Coefficient function for rss object</i>
----------	--

Description

Extracts coefficients for a given parameter pair (k,h).

Usage

```
## S3 method for class 'rss'
coef(object, k = NULL, h = NULL, ...)
```

Arguments

object	an object of class <code>rss</code>
k	the number of predictors indexing the desired fit
h	the number of observations indexing the desired fit
...	any other arguments

Value

An array of coefficients.

Author(s)

Ryan Thompson <ryan.thompson@monash.edu>

cv.bss	<i>Cross-validated best subset selection</i>
--------	--

Description

Fits a sequence of regression models using best subset selection and then cross-validates these models. This function is just a wrapper for the `cv.rss` function. The function solves the robust subset selection problem with $h=n$, using nonrobust measures of location and scale to standardise, and a nonrobust measure of prediction error in cross-validation.

Usage

```
cv.bss(
  x,
  y,
  k = 0:min(nrow(x) - 1, ncol(x), 20),
  mio = "min",
  nfold = 10,
  cv.loss = mspe,
  ...
)
```

Arguments

x	a predictor matrix
y	a response vector
k	the number of predictors to minimise sum of squares over; by default a sequence from 0 to 20
mio	one of 'min', 'all', or 'none' indicating whether to run the mixed-integer solver on the k that minimises the cv error, all k, or none at all
nfold	the number of folds to use in cross-validation

`cv.loss` an optional cross-validation loss-function to use; should accept a vector of errors; by default mean square prediction error

`...` any other arguments

Value

See documentation for the `cv.rss` function.

Author(s)

Ryan Thompson <ryan.thompson@monash.edu>

Examples

```
# Generate training data
set.seed(123)
n <- 100
p <- 10
p0 <- 5
beta <- c(rep(1, p0), rep(0, p - p0))
x <- matrix(rnorm(n * p), n, p)
e <- rnorm(n)
y <- x %*% beta + e

# Best subset selection with cross-validation
fit <- cv.bss(x, y)

# Extract model coefficients, generate predictions, and plot cross-validation results
coef(fit)
predict(fit, x)
plot(fit)
```

<code>cv.rss</code>	<i>Cross-validated robust subset selection</i>
---------------------	--

Description

Fits a sequence of regression models using robust subset selection and then cross-validates these models.

Usage

```
cv.rss(
  x,
  y,
  k = 0:min(nrow(x) - 1, ncol(x), 20),
  h = function(n) round(seq(0.75, 1, 0.05) * n),
  mio = "min",
  nfold = 10,
```

```

    cv.loss = tmspe,
    cluster = NULL,
    ...
)

```

Arguments

<code>x</code>	a predictor matrix
<code>y</code>	a response vector
<code>k</code>	the number of predictors to minimise sum of squares over; by default a sequence from 0 to 20
<code>h</code>	a function that takes the sample size that returns the number of observations to minimise sum of squares over; by default produces a sequence from 75 to 100 percent of sample size (in increments of 5 percent); a function is used here to facilitate varying sample sizes in cross-validation
<code>mio</code>	one of 'min', 'all', or 'none' indicating whether to run the mixed-integer solver on the <code>k</code> and <code>h</code> that minimise the cv error, all <code>k</code> and <code>h</code> , or none at all
<code>nfold</code>	the number of folds to use in cross-validation
<code>cv.loss</code>	an optional cross-validation loss-function to use; should accept a vector of errors; by default trimmed mean square prediction error with 25% trimming
<code>cluster</code>	an optional cluster for running cross-validation in parallel; must be set up using <code>parallel::makeCluster</code>
<code>...</code>	any other arguments

Value

An object of class `cv.rss`; a list with the following components:

<code>cv</code>	a matrix with the cross-validated values of <code>cv.loss</code> ; rows correspond to <code>k</code> and columns to <code>h</code>
<code>k</code>	a vector containing the values of <code>k</code> used in the fit
<code>h</code>	a vector containing the values of <code>h</code> used in the fit
<code>k.min</code>	the <code>k</code> yielding the lowest cross-validated <code>cv.loss</code>
<code>h.min</code>	the <code>h</code> yielding the lowest cross-validated <code>cv.loss</code>
<code>fit</code>	the fit from running <code>rss()</code> on the full data

Author(s)

Ryan Thompson <ryan.thompson@monash.edu>

Examples

```

# Generate training data with mixture error
set.seed(123)
n <- 100
p <- 10

```

```

p0 <- 5
ncontam <- 5
beta <- c(rep(1, p0), rep(0, p - p0))
x <- matrix(rnorm(n * p), n, p)
e <- rnorm(n, c(rep(10, ncontam), rep(0, n - ncontam)))
y <- x %*% beta + e

# Robust subset selection with cross-validation
fit <- cv.rss(x, y)

# Extract model coefficients, generate predictions, and plot cross-validation results
coef(fit)
predict(fit, x)
plot(fit)

```

plot.cv.rss

Plot function for cv.rss object

Description

Plot the cross-validation results from robust subset selection.

Usage

```

## S3 method for class 'cv.rss'
plot(x, ...)

```

Arguments

x	an object of class cv.rss
...	any other arguments

Value

A plot of the cross-validation results.

Author(s)

Ryan Thompson <ryan.thompson@monash.edu>

plot.rss	<i>Plot function for rss object</i>
----------	-------------------------------------

Description

Plot the coefficient profiles from robust subset selection.

Usage

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

Arguments

x	an object of class <code>rss</code>
...	any other arguments

Value

A plot of the coefficient profiles.

Author(s)

Ryan Thompson <ryan.thompson@monash.edu>

predict.cv.rss	<i>Predict function for cv.rss object</i>
----------------	---

Description

Generate predictions given new data using a given parameter pair (k, h).

Usage

```
## S3 method for class 'cv.rss'
predict(object, x.new, k = "k.min", h = "h.min", ...)
```

Arguments

object	an object of class <code>rss</code>
x.new	a matrix of new values for the predictors
k	the number of predictors indexing the desired fit; 'k.min' uses best k from cross-validation
h	the number of observations indexing the desired fit; 'h.min' uses best h from cross-validation
...	any other arguments

Value

An array of predictions.

Author(s)

Ryan Thompson <ryan.thompson@monash.edu>

predict.rss

Predict function for rss object

Description

Generate predictions for new data using a given parameter pair (k, h).

Usage

```
## S3 method for class 'rss'  
predict(object, x.new, k = NULL, h = NULL, ...)
```

Arguments

object	an object of class <code>rss</code>
x.new	a matrix of new values for the predictors
k	the number of predictors indexing the desired fit
h	the number of observations indexing the desired fit
...	any other arguments

Value

An array of predictions.

Author(s)

Ryan Thompson <ryan.thompson@monash.edu>

rss

Robust subset selection

Description

Fits a sequence of regression models using robust subset selection.

Usage

```
rss(
  x,
  y,
  k = 0:min(nrow(x) - 1, ncol(x), 20),
  h = round(seq(0.75, 1, 0.05) * nrow(x)),
  k.mio = NULL,
  h.mio = NULL,
  params = list(TimeLimit = 60, OutputFlag = 0),
  tau = 1.5,
  warm.start = TRUE,
  robust = TRUE,
  max.ns.iter = 100,
  max.gd.iter = 1e+05,
  eps = 1e-04
)
```

Arguments

x	a predictor matrix
y	a response vector
k	the number of predictors to minimise sum of squares over; by default a sequence from 0 to 20
h	the number of observations to minimise sum of squares over; by default a sequence from 75 to 100 percent of sample size (in increments of 5 percent)
k.mio	the subset of k for which the mixed-integer solver should be run
h.mio	the subset of h for which the mixed-integer solver should be run
params	a list of parameters (settings) to pass to Gurobi
tau	a positive number greater than 1 used to tighten coefficient bounds in the mixed-integer formulation; small values give quicker run times but can also exclude the optimal solution; can be Inf
warm.start	a logical indicating whether to warm start the mio solver using the heuristics
robust	a logical indicating whether to standardise the data robustly; median/mad for true and mean/sd for false
max.ns.iter	the maximum number of neighbourhood search iterations allowed
max.gd.iter	the maximum number of gradient descent iterations allowed per value of k and h
eps	a numerical tolerance parameter used to declare convergence

Details

The function first computes solutions over all combinations of k and h using heuristics. The heuristics include projected block-coordinate gradient descent and neighbourhood search (see [arXiv](#)). The solutions produced by the heuristics can be refined further using the mixed-integer solver. The tuning parameters that the solver operates on are specified by the `k.mio` and `h.mio` parameters, which must be subsets of k and h .

By default, the mixed-integer optimization problem is formulated with SOS constraints and bound constraints. The bound constraints are estimated as $\tau \|\hat{\beta}\|_{\infty}$, where $\hat{\beta}$ is output from the heuristics. For finite values of τ , the mixed-integer solver automatically converts the SOS constraints to Big-M constraints, which are numerically simpler to optimise.

Value

An object of class `rss`; a list with the following components:

<code>beta</code>	an array of estimated regression coefficients; columns correspond to k and matrices to h
<code>weights</code>	an array of binary weights; weights equal to one correspond to good observations selected for inclusion in the least squares fit; columns correspond to k and matrices to h
<code>objval</code>	a matrix with the objective function values; rows correspond to k and columns to h
<code>mipgap</code>	a matrix with the optimality gap values; rows correspond to k and columns to h
<code>k</code>	a vector containing the values of k used in the fit
<code>h</code>	a vector containing the values of h used in the fit

Author(s)

Ryan Thompson <ryan.thompson@monash.edu>

References

Thompson, R. (2021). 'Robust subset selection'. arXiv: [2005.08217](#).

Examples

```
# Generate training data with mixture error
set.seed(123)
n <- 100
p <- 10
p0 <- 5
ncontam <- 5
beta <- c(rep(1, p0), rep(0, p - p0))
x <- matrix(rnorm(n * p), n, p)
e <- rnorm(n, c(rep(10, ncontam), rep(0, n - ncontam)))
y <- x %*% beta + e
```

```
# Robust subset selection
fit <- robustsubsets::rss(x, y, k.mio = p0, h.mio = n - ncontam)

# Extract model coefficients, generate predictions, and plot cross-validation results
coef(fit, k = p0, h = n - ncontam)
predict(fit, x, k = p0, h = n - ncontam)
plot(fit)
```

Index

`coef.cv.rss`, [2](#)

`coef.rss`, [2](#)

`cv.bss`, [3](#)

`cv.rss`, [4](#)

`plot.cv.rss`, [6](#)

`plot.rss`, [7](#)

`predict.cv.rss`, [7](#)

`predict.rss`, [8](#)

`rss`, [9](#)