Package ‘CSUV’

May 4, 2020

Type Package
Title Combined Selection and Uncertainty Visualiser (CSUV)
Version 0.1.0
Author Christine Yuen
Maintainer Christine Yuen <yuenl@lse.ac.uk>
BugReports https://github.com/christineyuen/CSUV/issues
Imports doParallel, relaxo, caret, futile.logger, glmnet, ncvreg, shiny, shinyjs, DT, stats, MASS, datasets, grDevices, utils, graphics, HDCI, parcor, reshape2, ggplot2
License GPL-2
Encoding UTF-8
LazyData true
RoxygenNote 7.1.0
NeedsCompilation no
Repository CRAN
Date/Publication 2020-05-04 14:10:02 UTC

R topics documented:

  csuv ................................................................. 2
  csuv.ci ............................................................. 3
  csuv.plot.helper ............................................... 4
  get.compare.fit ............................................... 5
  get.compare.methods ......................................... 6
  get.csuv.final.mod ............................................ 6
  get.csuv.unique.fit ......................................... 7
Get the fitted results from Combined Selection and Uncertainty Visualiser (CSUV) method

**Description**

Get the fitted results from Combined Selection and Uncertainty Visualiser (CSUV) method

**Usage**

```r
csvg(X, Y, intercept, method.names = NULL, coef.est.method = lm.ols, B = 100, q = 0, fit.percent = 0.5, selection.criterion = "mse", num.core = 1, all.fits = NULL, log.level = NULL)
```

**Arguments**

- **X**: covariates (n times p matrix, n: number of entries, p: number of covariates)
- **Y**: response (vector with n entries)
- **intercept**: TRUE to fit the data with an intercept, FALSE to fit the data without an intercept
- **method.names**: vector of method names to be used in CSUV. Choose among "lasso", "elastic", "relaxo", "mcp" and "scad". Default is to use all methods listed above
- **coef.est.method**: method to estimate the coefficients of covariates after variable selection. User can provide his/her function. Default is ordinary least square
- **B**: number of subsampling. Default is 100
percentile of fitted models used per each subsampling in CSUV, according to the selection criterion on out-of-sample data in ascending order. Default is q = 0 (only the fitted model with the lowest MSE in a subsampling data is used)

fit.percent percentage of observations used in fitting in CSUV

selection.criterion = c("mse", "ebic"). Measure to select fitted models in subsampling dataset. "mse" is mean square error and "ebic" is extended BIC. Default is mse

num.core number of cores to use. Default is 1 (i.e. no parallel running)

all.fits (optional) all fitted models. If all.fits is provided, then CSUV will use the fitted models in all.fitted instead of fitting using subsampling data

log.level log level to set. Default is NULL, which means no change in log level. See the function CSUV::set.log.level for more details

Value

a list, which includes estimated coefficients (est.b), subsampling fitted models (mod.collection), number of times a method is selected (method.freq), relative frequency of each covariate (variable.freq), covariates ordered by relative frequency (variable.order).

Examples

X = matrix(rnorm(1000), nrow = 100)
Y = rowSums(X[,1:3])+rnorm(100)
mod.0 = csuv(X, Y, intercept = FALSE, q = 0, method.names = NULL)
print(mod.0)
mod.5 = csuv(X, Y, intercept = FALSE, q = 5, all.fits = mod.0$all.fits)
print(mod.5)

--

Confidence interval-like interval for uncertainty illustration

Description

Confidence interval-like interval for uncertainty illustration

Usage

csvu.ci(csuv.fit, level, type = "original", log.level = NULL)
Arguments

- **csuv.fit**: fitted results from `CSUV::csuv`
- **level**: significance level
- **type**: the type of the interval. When the type is "original", all estimated coefficients are used to calculate the interval. When the type is "conditional", only non-zero estimated coefficients are used. The type "conditional.1" is still in experimental stage, please do not use. Default is "original"
- **log.level**: log level to set. Default is NULL, which means no change in log level. See the function `CSUV::set.log.level` for more details

Value

a matrix. Each column represents an interval for a corresponding covariate

Examples

```r
X = matrix(rnorm(1000), nrow = 100)
Y = rowSums(X[,1:3])+rnorm(100)
mod.0 = csuv(X, Y, intercept = FALSE, q = 0, method.names = NULL)
print(csuv.ci(mod.0, level = 0.1, type = "original"))
```

```

Usage

```r
csv.plot.helper(new.fit, 
    with.unconditional = FALSE, 
    compare.method.fit = NULL, 
    compare.method.names = NULL, 
    cv.mod = NULL, 
    print.compare.method.points = FALSE, 
    ci.method = "conditional", 
    with.thr = TRUE, 
    with.violin = FALSE, 
    to.shade = TRUE, 
    level = 0.1, 
    var.freq.thr = 0.1, 
    ...
)
```

```
Arguments

new.fit fitted results from CSUV::csuv()
with.unconditional TRUE to get a unconditional boxplot on the same graph. Default is FALSE
compare.method.fit (optional) fitted results from CSUV::lm.compare.methods()
compare.method.names (optional) names of method to compare
cv.mod (optional) fitted results from cross validation
print.compare.method.points Default is FALSE
ci.method how the confidence interval should be calculated. Default is "conditional"
with.thr whether the selection by the CSUV should be show. Default is TRUE
with.violin whether the graph with violin plot
to.shade whether to shade the graph by the relative frequency calculated by CSUV. Default is TRUE
level the significant level of the whiskers. Default is 0.1
var.freq.thr minimum variable frequency to show, default is 0.1
... additional argument for plot

Value

a ggplot object

get.compare.fit Helper function, please do not use

Description

Helper function, please do not use

Usage

get.compare.fit(x, y, intercept, method.names, current.compare.fit = NULL)

Arguments

x covariates (n times p matrix, n: number of entries, p: number of covariates)
y response (vector with n entries)
intercept TRUE to fit the data with an intercept, FALSE to fit the data without an intercept
method.names vector of method names to be used in cross validation. Choose among "lasso", "elastic", "relaxo", "mcp" and "scad". Default is to use all methods listed above
current.compare.fit (optional)
Value

a list which includes the estimated coefficients (est.b) and the corresponding ordinary least square fit from stats::lm()

get.compare.methods  

Get a list of variable selection methods implemented in the CSUV package

Description

Get a list of variable selection methods implemented in the CSUV package

Usage

get.compare.methods()

Value

a list of functions

Examples

X = matrix(rnorm(1000), nrow = 100)
Y = rowSums(X[,1:3])+rnorm(100)
lasso.method = get.compare.methods()$lasso
lasso.mod = lasso.method(X, Y, intercept = FALSE)
print(lasso.mod$est.b)

g.get.csuv.final.mod  

Helper function, please do not use it

Description

Helper function, please do not use it

Usage

g.get.csuv.final.mod(
  X,
  Y,
  intercept,
  unique.fit,
  selection.criterion,
  coef.est.method = lm.ols,
  q,
  method.names,
  B
)

get.csuv.unique.fit

Arguments

- **X**: covariates (n times p matrix, n: number of entries, p: number of covariates)
- **Y**: response (vector with n entries)
- **intercept**: TRUE to fit the data with an intercept, FALSE to fit the data without an intercept
- **unique.fit**: from get.csuv.unique.fit
- **selection.criterion**: = c("mse", "ebic"). Measure to select fitted models in subsampling dataset. "mse" is mean square error and "ebic" is extended BIC. Default is mse
- **coef.est.method**: method to estimate the coefficients of covariates after variable selection. User can provide his/her function. Default is ordinary least square
- **q**: percentile of fitted models used per each subsampling in CSUV, according to the selection criterion on out-of-sample data in ascending order. Default is q = 0 (only the fitted model with the lowest MSE in a subsampling data is used)
- **method.names**: vector of method names to be used in CSUV. Choose among "lasso", "elastic", "relaxo", "mcp" and "scad". Default is to use all methods listed above
- **B**: number of subsampling. Default is 100

Value

a list of current fit

Description

Helper function, please do not use it

Usage

get.csuv.unique.fit(
  X,
  Y,
  intercept,
  method.names,
  B,
  fit.percent,
  current.fit = NULL,
  num.core = 1
)
Arguments

X  
covariates (n times p matrix, n: number of entries, p: number of covariates)

Y  
response (vector with n entries)

intercept  
TRUE to fit the data with an intercept, FALSE to fit the data without an intercept

method.names  
vector of method names to be used in CSUV. Choose among "lasso", "elastic", "relaxo", "mcp" and "scad". Default is to use all methods listed above

B  
number of subsampling. Default is 100

fit.percent  
percentage of observations used in fitting in CSUV

current.fit  
(optional) all fitted models

num.core  
number of cores to use. Default is 1 (i.e. no parallel running)

Value

a list of current fit

Description

Interactive version of the uncertainty illustration

Usage

interactive.uncertainty.illustration(log.level = NULL)

Arguments

log.level  
log level to set. Default is NULL, which means no change in log level. See the function CSUV::set.log.level for more details

Examples

interactive.uncertainty.illustration()
lm.compare.method

---

### Description

Get fitted models by fitting some variable selection methods

### Usage

```r
lm.compare.method(X, Y, intercept, method.names = NULL, log.level = NULL)
```

### Arguments

- **X**: covariates (n times p matrix, n: number of entries, p: number of covariates)
- **Y**: response (vector with n entries)
- **intercept**: TRUE to fit the data with an intercept, FALSE to fit the data without an intercept
- **method.names**: vector of method names to be used for fitting. Choose among "lasso", "elastic", "relaxo", "mcp" and "scad". Default is to fit the data using all methods listed above
- **log.level**: log level to set. Default is NULL, which means no change in log level. See the function CSUV::set.log.level for more details

### Value

estimated coefficients in a form of matrix. Each row corresponds to a method and each column corresponds to a covariate, with the first column corresponds to the intercept

### Examples

```r
X = matrix(rnorm(1000), nrow = 100)
Y = rowSums(X[,1:3])+rnorm(100)
compare.mod = lm.compare.method(X, Y, intercept = FALSE)
print(compare.mod)
```

---

lm.cv

---

### Description

Get a fitted model selected by cross validation

### Usage

```r
lm.cv(X, Y, intercept = TRUE, method.names = NULL, log.level = NULL)
```

### Arguments

- **X**: covariates (n times p matrix, n: number of entries, p: number of covariates)
- **Y**: response (vector with n entries)
- **intercept**: TRUE to fit the data with an intercept, FALSE to fit the data without an intercept
- **method.names**: vector of method names to be used for fitting. Choose among "lasso", "elastic", "relaxo", "mcp" and "scad". Default is to fit the data using all methods listed above
- **log.level**: log level to set. Default is NULL, which means no change in log level. See the function CSUV::set.log.level for more details

### Examples

```r
X = matrix(rnorm(1000), nrow = 100)
Y = rowSums(X[,1:3])+rnorm(100)
compare.mod = lm.compare.method(X, Y, intercept = FALSE)
print(compare.mod)
```
Usage

\[ \text{lm.cv}( \]
\[ X, \]
\[ Y, \]
\[ \text{intercept}, \]
\[ \text{fit.percent}, \]
\[ \text{num.repeat}, \]
\[ \text{method.names} = \text{NULL}, \]
\[ \text{num.core} = 1, \]
\[ \text{log.level} = \text{NULL} \]
\)

Arguments

- **X**: covariates (n times p matrix, n: number of entries, p: number of covariates)
- **Y**: response (vector with n entries)
- **intercept**: TRUE to fit the data with an intercept, FALSE to fit the data without an intercept
- **fit.percent**: percentage of observations used in fitting in cross validation. Each set of subsampling data will have (n times fit.percent) observations for fitting and n times (1-fit.percent) observations for calculating the mse
- **num.repeat**: number of sets of subsampling data used in cross validation
- **method.names**: vector of method names to be used in cross validation. Choose among "lasso", "elastic", "relaxo", "mcp" and "scad". Default is to use all methods listed above
- **num.core**: number of cores to use. Default is 1 (i.e. no parallel running)
- **log.level**: log level to set. Default is NULL, which means no change in log level. See the function CSUV::set.log.level for more details

Value

a list which includes the estimated coefficients (est.b) and the corresponding ordinary least square fit from stats::lm()

Examples

```r
set.log.level(futile.logger::WARN)
X = matrix(rnorm(1000), nrow = 100)
Y = rowSums(X[,1:3])+rnorm(100)
cv.mod = lm.cv(X, Y, intercept = FALSE, fit.percent = 0.5, num.repeat = 50)
print(cv.mod$est.b)
```
\texttt{lm.mse} \hspace{1cm} \textit{Calculate mse}

**Description**

Calculate mse

**Usage**

\begin{verbatim}
lm.mse(X, Y, mod = NULL, est.b = NULL, log.level = NULL)
\end{verbatim}

**Arguments**

\begin{itemize}
  \item \texttt{X} \hspace{1cm} covariates (n times p matrix, n: number of entries, p: number of covariates)
  \item \texttt{Y} \hspace{1cm} response (vector with n entries)
  \item \texttt{mod} \hspace{1cm} fitted model from \texttt{lm.cv} or \texttt{csuv}. Only provide mod or \texttt{est.b}
  \item \texttt{est.b} \hspace{1cm} estimated coefficient (with intercept). Only provide mod or \texttt{est.b}
  \item \texttt{log.level} \hspace{1cm} log level to set. Default is NULL, which means no change in log level. See the function \texttt{CSUV::set.log.level} for more details
\end{itemize}

**Value**

the value of estimated mean square error

**Examples**

\begin{verbatim}
X = matrix(rnorm(1000), nrow = 100)
Y = rowSums(X[,1:3])+rnorm(100)
compare.mod = lm.compare.method(X, Y, intercept = FALSE)
lm.mse(X, Y, est.b = compare.mod)
\end{verbatim}

\texttt{lm.ols.refit} \hspace{1cm} \textit{Get the ordinary least square estimated coefficients on a set of previously selected covariates}

**Description**

Get the ordinary least square estimated coefficients on a set of previously selected covariates

**Usage**

\begin{verbatim}
lm.ols.refit(X, Y, intercept, est.betas, log.level = NULL)
\end{verbatim}
Arguments

- **X**  
  Covariates (n times p matrix, n: number of entries, p: number of covariates)
- **Y**  
  Response (vector with n entries)
- **intercept**  
  TRUE to fit the data with an intercept, FALSE to fit the data without an intercept
- **est.betas**  
  Estimated betas from previous fitted result. It can be a vector with p+1 entries (first entry as intercept) or a matrix with p+1 columns. Non-zero coefficient means the corresponding covariate is selected
- **log.level**  
  Log level to set. Default is NULL, which means no change in log level. See the function CSUV::set.log.level for more details

Value

A list of estimated coefficients

Examples

```r
X = matrix(runif(1000), nrow = 100)
Y = rowSums(X[,1:3])+runif(100)
est.beta = rep(0, 11)
est.beta[2:5] = 1
ols.mod = lm.ols.refit(X, Y, intercept = FALSE, est.betas = est.beta)
print(ols.mod$est.b)
```

Description

Graphical illustration of selection uncertainty

Usage

```r
## S3 method for class 'csuv'
plot(
  x,
  with.unconditional = FALSE,
  compare.method.fit = NULL,
  cv.mod = NULL,
  with.thr = TRUE,
  with.violin = FALSE,
  to.shade = TRUE,
  ci.method = "conditional",
  level = 0.1,
  var.freq.thr = 0.1,
  log.level = NULL,
  ...
)
```
Arguments

- **x**: fitted results from `CSUV::csuv()` with `unconditional = TRUE` to get an unconditional boxplot on the same graph. Default is `FALSE`.
- **compare.method.fit**: (optional) fitted results from `CSUV::lm.compare.methods()`. Alternatively, user can provide a data frame with each row containing the estimated coefficients from a method. The name of each row should correspond to the name of the method. The first value of each row should be the value of the intercept.
- **cv.mod**: (optional) a vector of estimated coefficients from cross validation. The first value should be the value of the intercept.
- **with.thr**: whether the selection by the CSUV should be shown. Default is `TRUE`.
- **with.violin**: whether the graph with violin plot.
- **to.shade**: whether to shade the graph by the relative frequency calculated by CSUV. Default is `TRUE`.
- **ci.method**: how the confidence interval should be calculated. Default is "conditional".
- **level**: the significant level of the whiskers. Default is 0.1.
- **var.freq.thr**: minimum variable frequency to show. Default is 0.1.
- **log.level**: log level to set. Default is `NULL`, which means no change in log level. See the function `CSUV::set.log.level` for more details.
- **...**: additional argument for plot.

Value

a ggplot object

Examples

```r
X = matrix(rnorm(1000), nrow = 100)
Y = rowSums(X[,1:3])+rnorm(100)
mod.0 = csuv(X, Y, intercept = FALSE, q = 0, method.names = NULL)
cv.mod = lm.cv(X, Y, intercept = FALSE, fit.percent = 0.5, num.repeat = 50)
compare.mod = lm.compare.method(X, Y, intercept = FALSE)
plot(mod.0, compare.method.fit = compare.mod, cv.mod = cv.mod$est.b)
```

---

**print.csuv**

*Print the coefficients of csuv*

Description

Print the coefficients of csuv
set.log.level

Usage

## S3 method for class 'csuv'
print(x, ...)

Arguments

x output of csuv()
... additional arguments to "print"

Value

return value from print(x$est.b)

Description

Set the level of logger

Usage

set.log.level(level)

Arguments

level log level, setting the level to futile.logger::DEBUG provides most details log,
whereas setting the level to futile.logger::WARN provides least details log

Value

None

Examples

set.log.level(futile.logger::DEBUG)
set.log.level(futile.logger::INFO)
set.log.level(futile.logger::WARN)
Index

csv, 2
csv.ci, 3
csv.plot.helper, 4

get.compare.fit, 5
get.compare.methods, 6
get.csv.unique.fit, 7

interactive.uncertainty.illustration, 8

lm.compare.method, 9
lm.cv, 9
lm.mse, 11
lm.ols.refit, 11

plot.csv, 12
print.csv, 13

set.log.level, 14