Package 'CIpostSelect'

October 4, 2024

Type Package

Title Confidence Interval Post-Selection of Variable

Version 0.2.0

Maintainer Boubacar DIALLO <boubacar.diallo0@yahoo.com>

Description Calculates confidence intervals after variable selection using repeated data splits. The package offers methods to address the challenges of post-selection inference, ensuring more accurate confidence intervals in models involving variable selection. The two main functions are 'lmps', which records the different models selected across multiple data splits as well as the corresponding coefficient estimates, and 'cips', which takes the lmps object as input to select variables and perform inferences using two types of voting.

License MIT + file LICENSE

Imports MASS, doParallel, foreach, ggplot2, glmnet, magrittr, mlbench, tictoc

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

Author Boubacar DIALLO [aut, cre]

Repository CRAN

Date/Publication 2024-10-04 14:20:05 UTC

Contents

CIpostSelect-package *CIpostSelect*

Description

This package calculates post-selection confidence intervals for variables. It uses repeated data splitting with a voting mechanism and offers two methods for post-selection: Lasso and BIC. For Lasso, cross-validation is used to find the best lambda that fits the model. For BIC, since it's not possible to test all models, a backward or forward elimination method is applied. The selection is done on one part of the data, followed by calibration on the other part, and this process is repeated multiple times.

Details

This package provides two main functions:

- **lmps** : This function provides the model selection matrices for the different data splits, as well as the matrix of coefficient estimates for the selected models. Its 'summary' method gives important information about the appropriate voting type to use with the CIps function.

- **CIps** : This function takes an 'lmps' object as a argument, along with other parameters that specify the type of vote and the confidence level for the confidence intervals (calculated empirically).

Package Information

Package: CIpostSelect
Version: 0.1.0
Date: 2024-09-26
License: MIT

Author and Maintainer

Author: Boubacar DIALLO
Maintainer: Boubacar Diallo <boubacar.diallo0@yahoo.com>

Examples

```
library(mlbench)
data("BostonHousing")
# Create lmps object
model = lmps(medv ~ ., data = BostonHousing, method = "Lasso", N = 100)
# Summary of lmps
summary(model) # helps choose the appropriate vote type
# Create CIps object
cips = CIps(model, vote = "coef", alpha = 0.05, s.vote_coef = 0.5)
# Results
print(cips)
# Summary plot
```

```
plot(cips)
```

`CIps` *Creates an object of class CIps based on the provided parameters.*

Description

Creates an object of class CIps based on the provided parameters.

Usage

```
CIps(x, vote, alpha, s.vote_coef = 0.5)
```

Arguments

`x`	An object of class lmps, which contains the selection and coefficient estimation matrices.
`vote`	The type of vote to perform: "model" for selection based on the most frequent model, or "coef" for variable selection (e.g., if a variable is selected more than 50 percent of the time).
`alpha`	Specifies the confidence level for the confidence intervals.
`s.vote_coef`	A parameter between 0 and 1 that, when using "coef" voting, indicates the frequency threshold for selecting a variable.

Details

After obtaining the lmps object, which provides the selection matrices (models and coefficients), this function allows us to compute confidence intervals that are calculated empirically based on the chosen voting method and the desired level of certainty. The confidence intervals are obtained through empirical calculation on each vector of estimates for the corresponding coefficients.

CIps also provides an intercept (test version) estimated as follows: in the case of a vote on models, it takes the average of the intercept vector for the rows where the most frequently selected model in the N splits is chosen. For the vote on coefficients, the idea is to select the coefficient that has been chosen the least number of times among those retained and then average the intercept only for the rows where this coefficient is selected.

Value

An object of class CIps.

Examples

```
library(mlbench)
data("BostonHousing")
# lmps object
model = lmps(medv~., data = BostonHousing, method = "Lasso", N = 50)
# CIps object
cips = CIps(model, vote = "coef", alpha = 0.05, s.vote_coef = 0.5)
```

```
# lmps object
model = lmps(medv~., data = BostonHousing, method = "Lasso", N = 50, cores = 2)
# CIps object
cips = CIps(model, vote = "coef", alpha = 0.05, s.vote_coef = 0.5)
```

---

lmps *Function that handles storing our estimation and variable selection matrices during the different splits.*

Description

Function that handles storing our estimation and variable selection matrices during the different splits.

Usage

```
lmps(
  formula,
  data,
  method,
  N,
  p_split = 0.5,
  cores = NULL,
  direction = "backward",
  forced_var = NULL
)
```

Arguments

formula	Regression model to use, specified as a formula.
data	Data set to be used for regression modeling.
method	Method for variable selection. Should be one of "Lasso" or "BIC".
N	Number of splits.
p_split	Probabilities associated with the splits.

cores	Number of cores for parallel processing.
direction	It can take two values: "backward" and "forward". In the case of BIC, it specifies the direction in which the selection will be made.
forced_var	A character string specifying a predictor variable to be forced into selection. By default, it is NULL, allowing for no forced selection. If provided, this variable will be consistently selected during the N splits.

Details

We have data that we will split several times while shuffling it each time. Then, we will divide the data into two parts based on a specific probability for splitting. In the first half, we will perform model selection, followed by calibration on the second half. At the end of these steps, we will obtain matrices of dimensions N*p that represent the selected models and the estimated coefficients associated with these models.

Value

An object of class lmps

Examples

```
library(mlbench)
data("BostonHousing")
# lmps object
model = lmps(medv ~ ., data = BostonHousing, method = "Lasso", N = 50)


# A parallelized example
# lmps object
model = lmps(medv ~ ., data = BostonHousing, method = "Lasso", N = 50, cores = 2)
```

---

plot.CIps *Plot method for the CIps class*

Description

It provides a ggplot graphic where the x-axis displays all the explanatory variables, with the confidence intervals of the selected variables shown in green and the coefficient estimates represented as red points.

Usage

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

Arguments

x	An object of class CIps.
...	Additional arguments to be passed to the plot function.

Value

No return value, called for its side effects, which is plotting a graph.

Examples

```
library(mlbench)
data("BostonHousing")
# lmps object
model = lmps(medv~., data = BostonHousing, method = "Lasso", N = 50)
# CIps object
cips = CIps(model, vote = "coef", alpha = 0.05, s.vote_coef = 0.5)
# plot
plot(cips)


# lmps object
model = lmps(medv~., data = BostonHousing, method = "Lasso", N = 50, cores = 2)
# CIps object
cips = CIps(model, vote = "coef", alpha = 0.05, s.vote_coef = 0.5)
# plot
plot(cips)
```

---

predict.CIps *A predict function for cips*

Description

This function generates predictions based on a cips object.

Usage

```
## S3 method for class 'CIps'
predict(object, newdata, X = NULL, y = NULL, ...)
```

Arguments

object	An object of class 'cips'.
newdata	A dataframe containing new data to make predictions.
X	Explanatory variables, default is NULL.
y	Target corresponding to the explanatory variables, default is NULL.
...	Additional arguments for future use.

Details

When X and y are not NULL, they represent the data used to select the model. These data are then reused by recalibrating on the selected subset to obtain the beta estimates (hybrid approach).

Value

A numeric vector of predicted values.

---

print.CIps *Print method for the CIps class*

Description

It provides information on the selected variables, the estimated confidence intervals, and the coefficients of these selected variables.

Usage

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

Arguments

x	An object of class CIps.
...	Additional arguments to be passed to the print function.

Value

No return value, called for its side effects, which is printing the object to the console.

Examples

```
library(mlbench)
data("BostonHousing")
# lmps object
model = lmps(medv~., data = BostonHousing, method = "Lasso", N = 50)
# CIps object
cips = CIps(model, vote = "coef", alpha = 0.05, s.vote_coef = 0.5)
# print
print(cips)


# lmps object
model = lmps(medv~., data = BostonHousing, method = "Lasso", N = 50, cores = 2)
# CIps object
cips = CIps(model, vote = "coef", alpha = 0.05, s.vote_coef = 0.5)
# print
print(cips)
```

summary.lmps *Summary function for our lmps object*

Description

Summary function for our lmps object

Usage

```
## S3 method for class 'lmps'
summary(object, ...)
```

Arguments

object	Our lmps object
...	Other arguments ignored (for compatibility with generic)

Details

This function provides a summary of the data collected during the application of the lmps function. It summarizes how many times the most frequently selected model was chosen across our N divisions, as well as the selection frequency of variables in the different divisions. It can also provide the execution time of the lmps function, which may vary significantly depending on the chosen post-selection method and the dimensionality of our data.

Value

A summary of our lmps object

Examples

```
library(mlbench)
data("BostonHousing")
# lmps object
model = lmps(medv~., data = BostonHousing, method = "Lasso", N = 50)
summary(model)



# lmps object
model = lmps(medv~., data = BostonHousing, method = "Lasso", N = 50, cores = 2)
summary(model)
```

Index