Package: RFpredInterval (via r-universe)

September 6, 2024

Type Package

Title Prediction Intervals with Random Forests and Boosted Forests

Version 1.0.6

Description Implements various prediction interval methods with random forests and boosted forests. The package has two main functions: pibf() produces prediction intervals with boosted forests (PIBF) as described in Alakus et al. (2021) <arXiv:2106.08217> and rfpi() builds 15 distinct variations of prediction intervals with random forests (RFPI) proposed by Roy and Larocque (2020) <doi:10.1177/0962280219829885>.

Depends R (>= 3.6.0)

License GPL (>= 3)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.0

Imports ranger, data.table, hdrcde, parallel, data.tree, DiagrammeR

Suggests knitr, rmarkdown, testthat

URL https://github.com/calakus/RFpredInterval

BugReports https://github.com/calakus/RFpredInterval/issues

Repository https://calakus.r-universe.dev

RemoteUrl https://github.com/calakus/rfpredinterval

RemoteRef HEAD

RemoteSha 6170733a407edd4fff3ad5fdda118e05050634c9

Contents

`RFpredInterval-package`

RFpredInterval: A package for building prediction intervals with random forests and boosted forests

---

Description

`RFpredInterval` provides methods to build prediction intervals with random forests. The methods provided in the package are Prediction Intervals with Boosted Forests (PIBF) proposed by Alakus et al. (2021) and 15 distinct variations to build PIs proposed by Roy and Larocque (2020). `RFpredInterval` includes two main functions: `pibf()` and `rfpi()`. `pibf()` applies the PIBF method and it uses the `ranger` package (Wright and Ziegler, 2017) to fit random forests. `rfpi()` applies the 15 variations proposed by Roy and Larocque (2020). For `rfpi()`, `RFpredInterval` uses `randomForestSRC` package. For the least-squares splitting rule, both `randomForestSRC` and `ranger` packages are applicable.

Details

Among 16 methods, ten of them have specialized splitting rules in the random forest growing process. These methods are the ones with L1 and shortest prediction interval (SPI) splitting rules proposed by Roy and Larocque (2020). To implement these methods, the custom split feature of the `randomForestSRC` package (Ishwaran and Kogalur, 2021) have been utilised.

The `randomForestSRC` package allows users to define a custom splitting rule for the tree growing process. The user needs to define the customized splitting rule in the `splitCustom.c` file. After modifying the `splitCustom.c` file, all C source code files under the `src` folder of the package must be recompiled. Finally, the package must be re-installed for the custom split rule to become available. `RFpredInterval` uses `randomForestSRC` package by freezing at the version 2.11.0.

RFpredInterval functions

`pibf` `rfpi` `piall` `plot.rfpredinterval` `print.rfpredinterval`

References

Alakus, C., Larocque, D., and Labbe, A. (2021). RFpredInterval: An R Package for Prediction Intervals with Random Forests and Boosted Forests. arXiv preprint arXiv:2106.08217.

Ishwaran H, Kogalur U (2021). Fast Unified Random Forests for Survival, Regression, and Classification (RF-SRC). R package version 2.11.0, `https://cran.r-project.org/package=randomForestSRC`.

Roy, M. H., & Larocque, D. (2020). Prediction intervals with random forests. Statistical methods in medical research, 29(1), 205-229. doi:10.1177/0962280219829885.

Wright MN, Ziegler A (2017). "ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R." Journal of Statistical Software, 77(1), 1–17. doi:10.18637/jss.v077.i01.

`BostonHousing` *Boston housing data set*

Description

Housing data for 506 census tracts of Boston from the 1970 census. The data set contains the original data by Harrison and Rubinfeld (1979).

Usage

```
BostonHousing
```

Format

A data frame with three 506 rows observations on 14 variables. `medv` is the target variable. The variables are as follows:

- `crim`: per capita crime rate by town
- `zn`: proportion of residential land zoned for lots over 25,000 sq.ft
- `indus`: proportion of non-retail business acres per town
- `chas`: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- `nox`: nitric oxides concentration (parts per 10 million)
- `rm`: average number of rooms per dwelling
- `age`: proportion of owner-occupied units built prior to 1940
- `dis`: weighted distances to five Boston employment centres
- `rad`: index of accessibility to radial highways
- `tax`: full-value property-tax rate per USD 10,000
- `ptratio`: pupil-teacher ratio by town
- `b`: 1000(B - 0.63)^2 where B is the proportion of blacks by town
- `lstat`: percentage of lower status of the population
- `medv`: median value of owner-occupied homes in USD 1000's

Examples

```
## load data
data(BostonHousing, package = "RFpredInterval")
```

piall *Prediction intervals with all methods*

Description

Constructs prediction intervals with the 16 methods (PIBF method implemented in `pibf()` and 15 method variations implemented in `rfpi()`).

Usage

```
piall(
  formula,
  traindata,
  testdata,
  alpha = 0.05,
  num.trees = 2000,
  mtry = ceiling(px/3)
)
```

Arguments

`formula`	Object of class `formula` or `character` describing the model to fit.
`traindata`	Training data of class `data.frame`.
`testdata`	Test data of class `data.frame`.
`alpha`	Confidence level. (1 - `alpha`) is the desired coverage level. The default is `alpha` = 0.05 for the 95% prediction interval.
`num.trees`	Number of trees. The default is `num.trees` = 2000
`mtry`	Number of variables randomly selected as candidates for splitting a node. The default is rounded up $px/3$ where px is the number of variables.

Value

A list with the following components:

`PIBF`	Prediction intervals for test data with PIBF method. A list containing lower and upper bounds.
`LS_LM`	Prediction intervals for test data with least-squares (LS) splitting rule and classical method (LM). A list containing lower and upper bounds.
`LS_SPI`	Prediction intervals for test data with least-squares (LS) splitting rule and shortest PI (SPI) method. A list containing lower and upper bounds.
`LS_Quant`	Prediction intervals for test data with least-squares (LS) splitting rule and quantiles method. A list containing lower and upper bounds.
`LS_HDR`	Prediction intervals for test data with least-squares (LS) splitting rule and highest density region (HDR) method. A list containing lower and upper bounds of prediction interval for each test observation. There may be multiple PIs for a single observation.

LS_CHDR	Prediction intervals for test data with least-squares (LS) splitting rule and contiguous HDR method. A list containing lower and upper bounds.
L1_LM	Prediction intervals for test data with L_1 splitting rule and classical method (LM). A list containing lower and upper bounds.
L1_SPI	Prediction intervals for test data with L_1 splitting rule and shortest PI (SPI) method. A list containing lower and upper bounds.
L1_Quant	Prediction intervals for test data with L_1 splitting rule and quantiles method. A list containing lower and upper bounds.
L1_HDR	Prediction intervals for test data with L_1 splitting rule and highest density region (HDR) method. A list containing lower and upper bounds of prediction interval for each test observation. There may be multiple PIs for a single observation.
L1_CHDR	Prediction intervals for test data with L_1 splitting rule and contiguous HDR method. A list containing lower and upper bounds.
SPI_LM	Prediction intervals for test data with shortest PI (SPI) splitting rule and classical method (LM). A list containing lower and upper bounds.
SPI_SPI	Prediction intervals for test data with shortest PI (SPI) splitting rule and shortest PI (SPI) method. A list containing lower and upper bounds.
SPI_Quant	Prediction intervals for test data with shortest PI (SPI) splitting rule and quantiles method. A list containing lower and upper bounds.
SPI_HDR	Prediction intervals for test data with shortest PI (SPI) splitting rule and highest density region (HDR) method. A list containing lower and upper bounds of prediction interval for each test observation. There may be multiple PIs for a single observation.
SPI_CHDR	Prediction intervals for test data with shortest PI (SPI) splitting rule and contiguous HDR method. A list containing lower and upper bounds.
pred_pibf	Bias-corrected random forest predictions for test data.
pred_ls	Random forest predictions for test data with least-squares (LS) splitting rule.
pred_l1	Random forest predictions for test data with L_1 splitting rule.
pred_spi	Random forest predictions for test data with shortest PI (SPI) splitting rule.
test_response	If available, true response values of the test data. Otherwise, NULL.

See Also

pibf rfpi plot.rfpredinterval print.rfpredinterval

Examples

```
## load example data
data(BostonHousing, package = "RFpredInterval")
set.seed(2345)

## define train/test split
testindex <- 1
trainindex <- sample(2:nrow(BostonHousing), size = 50, replace = FALSE)
traindata <- BostonHousing[trainindex, ]
```

```
testdata <- BostonHousing[testindex, ]

## construct 95% PI with 16 methods for the first observation in testdata
out <- piall(formula = medv ~ ., traindata = traindata,
  testdata = testdata, num.trees = 50)
```

---

pibf *Prediction intervals with boosted forests*

Description

Constructs prediction intervals with boosted forests.

Usage

```
pibf(
  formula,
  traindata,
  testdata,
  alpha = 0.05,
  calibration = c("cv", "oob", FALSE),
  coverage_range = c(1 - alpha - 0.005, 1 - alpha + 0.005),
  numfolds = 5,
  params_ranger = list(num.trees = 2000, mtry = ceiling(px/3), min.node.size = 5,
    replace = TRUE),
  oob = FALSE
)
```

Arguments

`formula`	Object of class `formula` or `character` describing the model to fit.
`traindata`	Training data of class `data.frame`.
`testdata`	Test data of class `data.frame`.
`alpha`	Confidence level. (1 - `alpha`) is the desired coverage level. The default is `alpha` = 0.05 for the 95% prediction interval.
`calibration`	Calibration method for finding working level of `alpha`, i.e. α_w. Options are `"cv"`, `"oob"`, and `FALSE` standing for calibration with cross-validation, OOB calibration, and no calibration, respectively. See below for details. The default is `"cv"`.
`coverage_range`	The allowed target calibration range for coverage level. α_w is selected such that the `"cv"` or `"oob"` coverage is within `coverage_range`.
`numfolds`	Number of folds for calibration with cross-validation. The default is 5 folds.

`params_ranger` List of parameters that should be passed to `ranger`. In the default parameter set, `num.trees` = 2000, `mtry` = $px/3$ (rounded up), `min.node.size` = 5, `replace` = TRUE. See `ranger` for possible parameters.

`oob` Should out-of-bag (OOB) predictions and prediction intervals for the training observations be returned?

Value

A list with the following components:

`pred_interval` Prediction intervals for test data. A list containing lower and upper bounds.

`test_pred` Bias-corrected random forest predictions for test data.

`alphaw` Working level of `alpha`, i.e. α_w. If `calibration = FALSE`, it returns `NULL`.

`test_response` If available, test response.

`oob_pred_interval` Out-of-bag (OOB) prediction intervals for train data. Prediction intervals are built with `alpha`. If `oob = FALSE`, it returns `NULL`.

`oob_pred` Bias-corrected out-of-bag (OOB) predictions for train data. If `oob = FALSE`, it returns `NULL`.

`train_response` Train response.

Details

Calibration process

Let $(1 - \alpha)$ be the target coverage level. The goal of the calibration is to find the value of α_w, which is the working level of α called by Roy and Larocque (2020), such that the coverage level of the PIs for the training observations is closest to the target coverage level. Two calibration procedures are provided: calibration with cross-validation and out-of-bag (OOB) calibration.

1. In calibration with CV, we apply k-fold cross-validation to form prediction intervals for the training observations. In each fold, we split the original training data set into training and testing sets. For the training set, we train a one-step boosted random forest and compute the OOB residuals. Then, for each observation in the testing set, we build a PI. After completing CV, we compute the coverage level with the constructed PIs and if the coverage is not within the acceptable coverage range (`coverage_range`), then we apply a grid search to find the α_w such that α_w is the closest to the target α among the set of α_w's that ensures the target coverage level for the constructed PIs. Once we find the α_w, we use this level to build the PI for the new observations.
2. The OOB calibration procedure is proposed by Roy and Larocque (2020) and it is the default calibration procedure of `rfpi()`. See details section of `rfpi()` for the detailed explanation of this calibration procedure.

In terms of computational time, OOB calibration is faster than calibration with CV. However, empirical results show that OOB calibration may result in conservative prediction intervals. Therefore, the recommended calibration procedure for the PIBF method is calibration with CV.

References

Alakus, C., Larocque, D., and Labbe, A. (2021). RFpredInterval: An R Package for Prediction Intervals with Random Forests and Boosted Forests. arXiv preprint arXiv:2106.08217.

Roy, M. H., & Larocque, D. (2020). Prediction intervals with random forests. Statistical methods in medical research, 29(1), 205-229. doi:10.1177/0962280219829885.

See Also

piall rfpi print.rfpredinterval

Examples

```
## load example data
data(BostonHousing, package = "RFpredInterval")
set.seed(2345)

## define train/test split
testindex <- 1:10
trainindex <- sample(11:nrow(BostonHousing), size = 100, replace = FALSE)
traindata <- BostonHousing[trainindex, ]
testdata <- BostonHousing[testindex, ]
px <- ncol(BostonHousing) - 1

## construct 95% PI with "cv" calibration using 5-folds
out <- pibf(formula = medv ~ ., traindata = traindata,
  testdata = testdata, calibration = "cv", numfolds = 5,
  params_ranger = list(num.trees = 40))

## get the PI for the first observation in the testdata
c(out$pred_interval$lower[1], out$pred_interval$upper[1])

## get the bias-corrected random forest predictions for testdata
out$test_pred

## construct 90% PI with "oob" calibration
out2 <- pibf(formula = medv ~ ., traindata = traindata,
  testdata = testdata, alpha = 0.1, calibration = "oob",
  coverage_range = c(0.89,91), params_ranger = list(num.trees = 40))

## get the PI for the testdata
out2$pred_interval

## get the working level of alpha (alphaw)
out2$alphaw
```

plot.rfpredinterval *Plot constructed prediction intervals for* ('rfpredinterval', 'piall') *objects*

Description

Plots the 16 constructed PIs obtained with piall() for a test observation. For each method, the red point presents the point prediction and blue line shows the constructed prediction interval for the test observation. If the true response of the test observation is known, it is demonstrated with a dashed vertical line. Note that we may have multiple prediction intervals with the HDR PI method.

Usage

```
## S3 method for class 'rfpredinterval'
plot(x, test_id = 1, sort = TRUE, show_response = TRUE, ...)
```

Arguments

x	An object of class ('rfpredinterval', 'piall').
test_id	Integer value specifying the test observation to be plotted. The default is 1.
sort	Should the prediction intervals be sorted according to their lengths in the plot? The default is TRUE.
show_response	Should the true response value of the test observation (if available) be displayed in the plot?
...	Optional arguments to be passed to other methods.

Value

Invisibly, the prediction intervals and point predictions that were plotted for the test observation.

See Also

piall

Examples

```
## load example data
data(BostonHousing, package = "RFpredInterval")
set.seed(2345)

## define train/test split
testindex <- 1
trainindex <- sample(2:nrow(BostonHousing), size = 50, replace = FALSE)
traindata <- BostonHousing[trainindex, ]
testdata <- BostonHousing[testindex, ]

## build 95% PIs with all 16 methods for the first observation in testdata
```

```
out <- piall(formula = medv ~ ., traindata = traindata,
             testdata = testdata, num.trees = 50)

## plot the constructed PIs for test_id = 1 with all methods
plot(out, test_id = 1)
```

`print.rfpredinterval` *Print summary output*

Description

Print summary output from `pibf()`, `rfpi()`, or `piall()` functions. This is the default print method for the package.

Usage

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

Arguments

`x`	An object of class `('rfpredinterval', 'piall')`, `('rfpredinterval', 'pibf')`, or `('rfpredinterval', 'rfpi')`.
`...`	Optional arguments to be passed to other methods.

See Also

`pibf` `piall` `rfpi`

Examples

```
## load example data
data(BostonHousing, package = "RFpredInterval")
set.seed(2345)

## define train/test split
testindex <- 1:10
trainindex <- sample(11:nrow(BostonHousing), size = 100, replace = FALSE)
traindata <- BostonHousing[trainindex, ]
testdata <- BostonHousing[testindex, ]
px <- ncol(BostonHousing) - 1

## construct 95% PI with "cv" calibration using 5-folds
out <- pibf(formula = medv ~ ., traindata = traindata,
  testdata = testdata, calibration =  "oob",
  params_ranger = list(num.trees = 40))
```

```
## print summary output
print(out)

## contruct 95% PI with "ls" split rule, "lm", "quant" and "spi" PI methods
## with calibration and use "ranger" package for RF training
out2 <- rfpi(formula = medv ~ ., traindata = traindata,
  testdata = testdata, split_rule = "ls", pi_method = c("lm", "quant", "spi"),
  rf_package = "ranger", params_ranger = list(num.trees = 50))

## print summary output
print(out2)
```

---

rfpi *Prediction intervals with random forests*

Description

Constructs prediction intervals with 15 distinct variations proposed by Roy and Larocque (2020). The variations include two aspects: The method used to build the forest and the method used to build the prediction interval. There are three methods to build the forest, (i) least-squares (LS), (ii) L1 and (iii) shortest prediction interval (SPI) from the CART paradigm. There are five methods for constructing prediction intervals, classical method, shortest prediction interval, quantile method, highest density region, and contiguous HDR.

Usage

```
rfpi(
  formula,
  traindata,
  testdata,
  alpha = 0.05,
  split_rule = c("ls", "l1", "spi"),
  pi_method = c("lm", "spi", "quant", "hdr", "chdr"),
  calibration = TRUE,
  rf_package = c("rfsrc", "ranger"),
  params_rfsrc = list(ntree = 2000, mtry = ceiling(px/3), nodesize = 5, samptype =
    "swr"),
  params_ranger = list(num.trees = 2000, mtry = ceiling(px/3), min.node.size = 5,
    replace = TRUE),
  params_calib = list(range = c(1 - alpha - 0.005, 1 - alpha + 0.005), start = (1 -
    alpha), step = 0.01, refine = TRUE),
  oob = FALSE
)
```

Arguments

formula
: Object of class formula or character describing the model to fit.

traindata
: Training data of class data.frame.

testdata
: Test data of class data.frame.

alpha
: Confidence level. (1 - alpha) is the desired coverage level. The default is alpha = 0.05 for the 95% prediction interval.

split_rule
: Split rule for building a forest. Options are "ls" for CART with least-squares (LS) splitting rule, "l1" for CART with L1 splitting rule, "spi" for CART with shortest prediction interval (SPI) splitting rule. The default is "ls".

pi_method
: Methods for building a prediction interval. Options are "lm" for classical method, "spi" for shortest prediction interval, "quant" for quantile method, "hdr" for highest density region, and "chdr" for contiguous HDR. The default is to use all methods for PI construction. Single method or a subset of methods can be applied.

calibration
: Apply OOB calibration for finding working level of alpha, i.e. α_w. See below for details. The default is TRUE.

rf_package
: Random forest package that can be used for RF training. Options are "rfsrc" for randomForestSRC and "ranger" for ranger packages. Split rule "ls" can be used with both packages. However, "l1" and "spi" split rules can only be used with "rfsrc". The default is "rfsrc".

params_rfsrc
: List of parameters that should be passed to randomForestSRC. In the default parameter set, ntree = 2000, mtry = $px/3$ (rounded up), nodesize = 5, samptype = "swr". See randomForestSRC for possible parameters.

params_ranger
: List of parameters that should be passed to ranger. In the default parameter set, num.trees = 2000, mtry = $px/3$ (rounded up), min.node.size = 5, replace = TRUE. See ranger for possible parameters.

params_calib
: List of parameters for calibration procedure. range is the allowed target calibration range for coverage level. The value that provides a coverage level within the range is chosen as α_w. start is the initial coverage level to start calibration procedure. step is the coverage step size for each calibration iteration. refine is the gradual decrease in step value when close to target coverage level, the default is TRUE which allows gradual decrease.

oob
: Should out-of-bag (OOB) predictions and prediction intervals for the training observations be returned?

Value

A list with the following components:

lm_interval
: Prediction intervals for test data with the classical method. A list containing lower and upper bounds.

spi_interval
: Prediction intervals for test data with SPI method. A list containing lower and upper bounds.

hdr_interval — Prediction intervals for test data with HDR method. A list containing lower and upper bounds of prediction interval for each test observation. There may be multiple PIs for a single observation.

chdr_interval — Prediction intervals for test data with contiguous HDR method. A list containing lower and upper bounds.

quant_interval — Prediction intervals for test data with quantiles method. A list containing lower and upper bounds.

test_pred — Random forest predictions for test data.

test_response — If available, test response.

alphaw — Working level of alpha, i.e. α_w. A numeric array for the PI methods entered with pi_method. If calibration = FALSE, it returns NULL.

split_rule — Split rule used for building the random forest.

rf_package — Random forest package that was used for RF training.

oob_pred_interval — Out-of-bag (OOB) prediction intervals for train data. Prediction intervals are built with alpha. If oob = FALSE, it returns NULL.

oob_pred — Out-of-bag (OOB) predictions for train data. If oob = FALSE, it returns NULL.

train_response — Train response.

Details

Calibration process

The calibration procedure uses the "Bag of Observations for Prediction" (BOP) idea. BOP for a new observation is built with the set inbag observations that are in the same terminal nodes as the new observation. The calibration procedure uses the BOPs constructed for the training observations. BOP for a training observation is built using only the trees where this training observation is out-of-bag (OOB).

Let $(1-\alpha)$ be the target coverage level. The goal of the calibration is to find the value of α_w, which is the working level of α called by Roy and Larocque (2020), such that the coverage level of the prediction intervals for the training observations is closest to the target coverage level. The idea is to find the value of α_w using the OOB-BOPs. Once found, $(1-\alpha_w)$ becomes the level used to build the prediction intervals for the new observations.

References

Roy, M. H., & Larocque, D. (2020). Prediction intervals with random forests. Statistical methods in medical research, 29(1), 205-229. doi:10.1177/0962280219829885.

See Also

piall pibf print.rfpredinterval

Examples

```
## load example data
data(BostonHousing, package = "RFpredInterval")
set.seed(2345)

## define train/test split
testindex <- 1:10
trainindex <- sample(11:nrow(BostonHousing), size = 100, replace = FALSE)
traindata <- BostonHousing[trainindex, ]
testdata <- BostonHousing[testindex, ]
px <- ncol(BostonHousing) - 1

## contruct 90% PI with "l1" split rule and "spi" PI method with calibration
out <- rfpi(formula = medv ~ ., traindata = traindata,
  testdata = testdata, alpha = 0.1, calibration = TRUE,
  split_rule = "l1", pi_method = "spi", params_rfsrc = list(ntree = 50),
  params_calib = list(range = c(0.89, 0.91), start = 0.9, step = 0.01,
  refine = TRUE))

## get the PI with "spi" method for first observation in the testdata
c(out$spi_interval$lower[1], out$spi_interval$upper[1])

## get the random forest predictions for testdata
out$test_pred

## get the working level of alpha (alphaw)
out$alphaw

## contruct 95% PI with "ls" split rule, "lm" and "quant" PI methods
## with calibration and use "ranger" package for RF training
out2 <- rfpi(formula = medv ~ ., traindata = traindata,
  testdata = testdata, split_rule = "ls", pi_method = c("lm", "quant"),
  rf_package = "ranger", params_ranger = list(num.trees = 50))

## get the PI with "quant" method for the testdata
cbind(out2$quant_interval$lower, out2$quant_interval$upper)
```

Index