Package ‘ConformalSmallest’

October 12, 2022

Title Efficient Tuning-Free Conformal Prediction
Version 1.0
Description An implementation of efficiency first conformal prediction (EFCP) and validity first con-
formal prediction (VFCP) that demonstrates both validity (coverage guarantee) and effi-
ciency (width guarantee). To learn how to use it, check the vignettes for a quick tutor-
rial. The package is based on the work by Yang Y., Kuchibhotla A.,(2021) <arxiv:2104.13871>.

URL https://github.com/Elsa-Yang98/ConformalSmallest
Imports glmnet, mvtnorm, stats, MASS, quantregForest
License GPL (>= 3)
Encoding UTF-8
RoxygenNote 7.1.1
LazyData true
Suggests testthat (>= 3.0.0), knitr, rmarkdown, ggplot2, repr
Config/testthat/edition 3
Depends R (>= 3.5.0)
VignetteBuilder knitr
NeedsCompilation no
Author Yachong Yang [aut, cre]
Maintainer Yachong Yang <yachong@wharton.upenn.edu>
Repository CRAN
Date/Publication 2021-08-09 14:10:06 UTC

R topics documented:

conf_CQR ............................................................... 2
conf_CQR_conditional ............................................. 3
conf_CQR_prelim .................................................... 3
conf_CQR_reg ........................................................ 4
conf_CQR_reg_conditional ....................................... 5
cv.fun ................................................................. 6
Conditional width and coverage for CQR, internal function used inside conf_CQR_conditional

Description

Conditional width and coverage for CQR, internal function used inside conf_CQR_conditional

Usage

conf_CQR(X1, Y1, X2, Y2, beta, mtry, ntree, alpha = 0.1)

Arguments

- X1: training matrix to fit the quantile regression forest
- Y1: training vector
- X2: training matrix to compute the conformal scores
- Y2: training vector to compute the conformal scores
- beta: nominal quantile level
- mtry: random forest parameter
- ntree: random forest parameter
- alpha: miscoverage level

Value

a function for computing conditional width and coverage
**conf_CQR_conditional**  
*Conditional width and coverage for CQR*

### Description
Conditional width and coverage for CQR

### Usage
conf_CQR_conditional(x, y, beta, mtry, ntree, alpha = 0.1)

### Arguments
- **x**: A $N \times d$ training matrix
- **y**: A $N \times 1$ training vector
- **beta**: nominal quantile level
- **mtry**: random forest parameter
- **ntree**: random forest parameter
- **alpha**: miscoverage level

### Value
a function for computing conditional width and coverage

---

**conf_CQR_prelim**  
*preliminary function for CQR*

### Description
preliminary function for CQR

### Usage
conf_CQR_prelim(X1, Y1, X2, Y2, beta_grid, mtry, ntree, alpha = 0.1)

### Arguments
- **X1**: A $n_1 \times d$ matrix for training
- **Y1**: A $n_1 \times 1$ vector for training
- **X2**: A $n_2 \times d$ matrix for calibration
- **Y2**: A $n_2 \times 1$ vector for calibration
- **beta_grid**: a grid of beta’s
- **mtry**: mtry parameter in random forest
- **ntree**: number of trees parameter in random forest
- **alpha**: miscoverage level
Value

the smallest width and its corresponding beta

---

**conf_CQR_reg**

*EFCP and VFCP for CQR, CQR-m, CQR-r*

Description

EFCP and VFCP for CQR, CQR-m, CQR-r

Usage

```r
conf_CQR_reg(
  x,
  y,
  split,
  beta_grid,
  mtry_grid,
  ntree_grid,
  method = "efficient",
  alpha = 0.1
)
```

Arguments

- `x` A N*d training matrix
- `y` A N*1 training vector
- `split` a vector of length 1 for efcp, length 2 for vfcp
- `beta_grid` a grid of beta’s
- `mtry_grid` a grid of mtry
- `ntree_grid` a grid of ntree
- `method` "efficient" for efcp; "valid" for vfcp
- `alpha` miscoverage level

Value

the selected cqr method
conf_CQR_reg_conditional

Conditional width and coverage for EFCP, VFCP between CQR, CQR-m, CQR-r

Description

Conditional width and coverage for EFCP, VFCP between CQR, CQR-m, CQR-r

Usage

conf_CQR_reg_conditional(
  x,
  y,
  split,
  beta_grid,
  mtry_grid,
  ntree_grid,
  method = "efficient",
  alpha = 0.1
)

Arguments

x A N*d training matrix
y A N*1 training vector
split a vector of length 1 for efcp, length 2 for vfcp
beta_grid a grid of beta's
mtry_grid a grid of mtry
ntree_grid a grid of ntree
method "efficient" for efcp; "valid" for vfcp
alpha miscoverage level

Value

the selected cqr method
cv.fun  

Cross validation conformal prediction for ridge regression

Description

Cross validation conformal prediction for ridge regression

Usage

cv.fun(X, Y, X0, lambda = seq(0, 100, length = 100), nfolds = 10, alpha = 0.1)

Arguments

X  A N*d training matrix
Y  A N*1 training vector
X0  A N0*d testing vector
lambda  a sequence of penalty parameters for ridge regression
nfolds  number of folds
alpha  misscovracy level

Value

upper and lower prediction intervals for X0

efcp.fun  

Efficiency first conformal prediction for ridge regression

Description

Efficiency first conformal prediction for ridge regression

Usage

efcp.fun(X, Y, X0, lambda = seq(0, 100, length = 100), alpha = 0.1)

Arguments

X  A N*d training matrix
Y  A N*1 training vector
X0  A N0*d testing vector
lambda  a sequence of penalty parameters for ridge regression
alpha  miscoverage level
efcp_cqr

Value

upper and lower prediction intervals for X0.

Examples

df=3
d = 5
n=50  #number of training samples
n0=10  #number of prediction points
rho=0.5
Sigma=matrix(rho,d,d)
diag(Sigma)=rep(1,d)
beta=rep(1:5,d/5)
X0=mvtnorm::rmvt(n0,Sigma,df)
X=mvtnorm::rmvt(n,Sigma,df)  #multivariate t distribution
eps=rt(n,df)*(1+sqrt(X[,1]^2+X[,2]^2))
Y=X%*%beta+eps
out.efcp=efcp.fun(X,Y,X0)
out.efcp$up
out.efcp$lo

Description

Efficiency first conformal prediction for Conformal Quantile Regression

Usage

efcp_cqr(x, y, split, beta_grid, params_grid, alpha = 0.1)

Arguments

x A N*d training matrix
y A N*1 training vector
split a number between 0 and 1
beta_grid a grid of beta’s
params_grid a grid of mtry and ntree
alpha miscoverage level

Value

average prediction width and a function for coverage on some testing points
**Description**

Efficiency first conformal prediction for ridge regression

**Usage**

```r
efcp_ridge(X, Y, X0, lambda = seq(0, 100, length = 100), alpha = 0.1)
```

**Arguments**

- `X`: A N*d training matrix
- `Y`: A N*1 training vector
- `X0`: A N0*d testing vector
- `lambda`: a sequence of penalty parameters for ridge regression
- `alpha`: miscoverage level

**Value**

upper and lower prediction intervals for X0.

**Examples**

```r
df=3
d = 5
n=50  #number of training samples
n0=10  #number of prediction points
rho=0.5
Sigma=matrix(rho,d,d)
diag(Sigma)=rep(1,d)
beta=rep(1:5,d/5)
X0=rmvt(n0,df)
X=rmvt(n,df)  #multivariate t distribution
eps=rtn(n,df)*((1+sqrt(X[,1]^2+X[,2]^2)+X[,2]^2))
Y=X%*%beta+eps
out.efcp=efcp.fun(X,Y,X0)
out.efcp$up
out.efcp$lo
```
ginverse.fun

**Conformal prediction for linear regression**

### Description
Conformal prediction for linear regression

### Usage
```r
ginverse.fun(x, y, x0, alpha = 0.1)
```

### Arguments
- **x**: A N*d training matrix
- **y**: A N*1 training vector
- **x0**: A N0*d testing vector
- **alpha**: miscoverage level

### Value
upper and lower prediction intervals for X0

---

ginverselm.funs

**Internal function used for** ginverse.fun

### Description
Internal function used for ginverse.fun

### Usage
```r
ginverselm.funs(intercept = TRUE, lambda = 0)
```

### Arguments
- **intercept**: default is TRUE
- **lambda**: a vector
my.ginverselm.funs  
Internal function used for ginverse.fun

Description
Internal function used for ginverse.fun

Usage
my.ginverselm.funs

Format
An object of class list of length 4.

naive.fun  
Conformal prediction for linear regression

Description
Conformal prediction for linear regression

Usage
naive.fun(X, Y, X0, alpha = 0.1)

Arguments
X  A N*d training matrix
Y  A N*1 training vector
X0  A N0*d testing vector
alpha  miscoverage level

Value
upper and lower prediction intervals for X0
Outcomes of an example for tuning-free conformalized quantile regression (CQR).

Description

A dataset containing the experiment results used in the vignettes.

Usage

pois_n400_reps100

Format

A list with 10 elements: x_test, n, nrep, width_mat, cov_mat, beta_mat, ntree_mat, cqr_method_mat, evaluations, alpha

x_test test points of x

n number of training samples

nrep number of replications

width_mat a data frame with the first column being the width of the prediction regions

cov_mat a data frame with the first column being the coverage of the prediction regions

beta_mat a data frame with the first column being the beta for CQR used in the final prediction

ntree_mat a data frame with the first column being the number of trees for CQR used in the final prediction

ntree_mat a data frame with the first column being the CQR method (among CQR, CQR-m, CQR-r) used in the final prediction

alpha desired miscoverage level

Source

For details please see the "Example-tuning_free_CQR" vignette: vignette("Example-tuning_free_CQR", package = "ConformalSmallest")
ridge_linear_cov100_t3

Description

A dataset containing the experiment results used in the vignettes.

Usage

ridge_linear_cov100_t3

Format

A list with 7 elements: dim_linear_t3, cov.param_linear_fm_t3, cov.naive_linear_fm_t3, cov.vfcp_linear_fm_t3, cov.star_linear_fm_t3, cov.cv5_linear_fm_t3, cov.efcp_linear_fm_t3

- **dim** dimensions used in the experiment
- **len.param** a matrix with coverages for the prediction regions produced by the parametric method
- **len.naive** a matrix with coverages for the prediction regions produced by naive linear regression method
- **len.vfcp** a matrix with coverages for the prediction regions produced by VFCP
- **len.star** a matrix with coverages for the prediction regions produced by cross validation with the errors
- **len.cv5** a matrix with coverages for the prediction regions produced by cross-validation with 5 splits
- **len.efcp** a matrix with coverages for the prediction regions produced by efcp

Source

For details please see the "Example-tuning_free_ridge_regression" vignette: vignette("Example-tuning_free_ridge_regression", package = "ConformalSmallest")
Outcomes of an example for tuning-free conformal prediction with ridge regression.
len.vfcp  na matrix with widths for the prediction regions produced by VFCP
len.star  a matrix with widths for the prediction regions produced by cross validation with the errors
len.cv5  a matrix with widths for the prediction regions produced by cross-validation with 5 splits
len.efcp  a matrix with widths for the prediction regions produced by efcp

Source
For details please see the "Example-tuning_free_ridge_regression" vignette:

vignette("Example-tuning_free_ridge_regression",
package = "ConformalSmallest")

\begin{verbatim}
ridge_linear_len100_t5
  # Outcomes of an example for tuning-free conformal prediction with
  # ridge regression.
\end{verbatim}

Description
A dataset containing the experiment results used in the vignettes.

Usage
ridge_linear_len100_t5

Format
A list with 6 elements: len.param_linear_fm_t5, len.naive_linear_fm_t5, len.vfcp_linear_fm_t5,
len.star_linear_fm_t5, len.cv5_linear_fm_t5, len.efcp_linear_fm_t5

len.param  a matrix with widths for the prediction regions produced by the parametric method
len.naive  a matrix with widths for the prediction regions produced by naive linear regression method
len.vfcp  na matrix with widths for the prediction regions produced by VFCP
len.star  a matrix with widths for the prediction regions produced by cross validation with the errors
len.cv5  a matrix with widths for the prediction regions produced by cross-validation with 5 splits
len.efcp  a matrix with widths for the prediction regions produced by efcp

Source
For details please see the "Example-tuning_free_ridge_regression" vignette:

vignette("Example-tuning_free_ridge_regression",
package = "ConformalSmallest")
star.fun

Conformal prediction for ridge regression, tuning parameter by minimizing the mean of the residuals

Description
Conformal prediction for ridge regression, tuning parameter by minimizing the mean of the residuals

Usage
star.fun(X, Y, X0, lambda = seq(0, 100, length = 100), alpha = 0.1)

Arguments
- X: A N*d training matrix
- Y: A N*1 training vector
- X0: A N0*d testing vector
- lambda: a sequence of penalty parameters for ridge regression
- alpha: miscoverage level

Value
upper and lower prediction intervals for X0

vfcp.fun

Validity first conformal prediction for ridge regression

Description
Validity first conformal prediction for ridge regression

Usage
vfcp.fun(X, Y, X0, lambda = seq(0, 100, length = 100), alpha = 0.1)

Arguments
- X: A N*d training matrix
- Y: A N*1 training vector
- X0: A N0*d testing vector
- lambda: a sequence of penalty parameters for ridge regression
- alpha: miscoverage level
Value

upper and lower prediction intervals for X0.

Examples

df=3
d = 5
n=50  # number of training samples
n0=10  # number of prediction points
rho=0.5
Sigma=matrix(rho,d,d)
diag(Sigma)=rep(1,d)
beta=rep(1:5,d/5)
X0=mvtnorm::rmvt(n0,Sigma,df)
X=mvtnorm::rmvt(n,Sigma,df)  # multivariate t distribution
eps=rt(n,df)*(1+sqrt(X[,1]^2+X[,2]^2))
Y=X%*%beta+eps
out.vfcp=vfcp.fun(X,Y,X0)
out.vfcp$up
out.vfcp$lo

vfcp_cqr  Validity first conformal prediction for Conformal Quantile Regression

Description

Validity first conformal prediction for Conformal Quantile Regression

Usage

vfcp_cqr(x, y, split, beta_grid, params_grid, alpha = 0.1)

Arguments

x  A N*d training matrix
y  A N*1 training vector
split  a number between 0 and 1
beta_grid  a grid of beta’s
params_grid  a grid of mtry and ntree
alpha  miscoverage level

Value

average prediction width and a function for coverage on some testing points
Validity first conformal prediction for ridge regression

Usage

```
vfcp_ridge(X, Y, X0, lambda = seq(0, 100, length = 100), alpha = 0.1)
```

Arguments

- **X**: A N*d training matrix
- **Y**: A N*1 training vector
- **X0**: A N0*d testing vector
- **lambda**: a sequence of penalty parameters for ridge regression
- **alpha**: miscoverage level

Value

upper and lower prediction intervals for X0.

Examples

```
df=3
d = 5
n=50  #number of training samples
n0=10  #number of prediction points
rho=0.5
Sigma=matrix(rho,d,d)
diag(Sigma)=rep(1,d)
beta=rep(1:5,d/5)
X0=mvtnorm::rmvt(n0,Sigma,df)
X=mvtnorm::rmvt(n,Sigma,df)  #multivariate t distribution
eps=rt(n,df)*(1+sqrt(X[,1]^2+X[,2]^2))
Y=X%*%beta+eps
out.vfcp=vfcp.fun(X,Y,X0)
out.vfcp$up
out.vfcp$lo
```
Index

* datasets
  my.ginverselm.funs, 10
  pois_n400_reps100, 11
  ridge_linear_cov100_t3, 12
  ridge_linear_cov100_t5, 12
  ridge_linear_len100_t3, 13
  ridge_linear_len100_t5, 14
conf_CQR, 2
conf_CQR_conditional, 3
conf_CQR_prelim, 3
conf_CQR_reg, 4
conf_CQR_reg_conditional, 5
cv.fun, 6
efcp.fun, 6
efcp_cqr, 7
efcp_ridge, 8
ginverse.fun, 9
ginverselm.funs, 9
my.ginverselm.funs, 10
naive.fun, 10
pois_n400_reps100, 11
ridge_linear_cov100_t3, 12
ridge_linear_cov100_t5, 12
ridge_linear_len100_t3, 13
ridge_linear_len100_t5, 14
star.fun, 15
vfcp.fun, 15
vfcp_cqr, 16
vfcp_ridge, 17