

# Package ‘ConformalSmallest’

May 7, 2026

**Title** Efficient Tuning-Free Conformal Prediction

**Version** 1.0

**Description** An implementation of efficiency first conformal prediction (EFCP) and validity first conformal prediction (VFCP) that demonstrates both validity (coverage guarantee) and efficiency (width guarantee). To learn how to use it, check the vignettes for a quick tutorial. The package is based on the work by Yang Y., Kuchibhotla A.,(2021) <doi:10.48550/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

## Contents

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
vfcf.fun . . . . .	15
vfcf_cqr . . . . .	16
vfcf_ridge . . . . .	17

## Index 18

---

conf_CQR	<i>Conditional width and coverage for CQR, internal function used inside conf_CQR_conditional</i>
----------	---

---

### 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*d training matrix
y	A N*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 n1*d matrix for training
Y1	A n1*1 vector for training
X2	A n2*d matrix for calibration
Y2	A n2*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	<i>EFCP and VFCP for CQR, CQR-m, CQR-r</i>
--------------	--

---

**Description**

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

**Usage**

```
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 efc, length 2 for vfc
beta_grid	a grid of beta's
mtry_grid	a grid of mtry
ntree_grid	a grid of ntree
method	"efficient" for efc; "valid" for vfc
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 efcf, length 2 for vfcf
beta_grid	a grid of beta's
mtry_grid	a grid of mtry
ntree_grid	a grid of ntree
method	"efficient" for efcf; "valid" for vfcf
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), n folds = 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
n folds	number of folds
alpha	miscoverage 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

**Value**

upper and lower prediction intervals for  $X_0$ .

**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
```

---

 efcp\_cqr

*Efficiency first conformal prediction for Conformal Quantile Regression*

---

**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 $N \times d$ training matrix
y	A $N \times 1$ 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

---

 efcp\_ridge

*Efficiency first conformal prediction for ridge regression*


---

## Description

Efficiency first conformal prediction for ridge regression

## Usage

```
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

```
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
```

---

`ginverse.fun`*Conformal prediction for linear regression*

---

**Description**

Conformal prediction for linear regression

**Usage**

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

**Arguments**

<code>x</code>	A $N \times d$ training matrix
<code>y</code>	A $N \times 1$ training vector
<code>x0</code>	A $N_0 \times d$ testing vector
<code>alpha</code>	miscoverage level

**Value**

upper and lower prediction intervals for  $X_0$

---

`ginverselm.funs`*Internal function used for ginverse.fun*

---

**Description**

Internal function used for ginverse.fun

**Usage**

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

**Arguments**

<code>intercept</code>	default is TRUE
<code>lambda</code>	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

---

pois_n400_reps100	<i>Outcomes of an example for tuning-free conformalized quantile regression(CQR).</i>
-------------------	---

---

### 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

*Outcomes of an example for tuning-free conformal prediction with ridge regression.*

---

### 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** na matrix with coverages for the prediction regions produced by VFPCP

**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_reg", package = "ConformalSmallest")`

---

ridge\_linear\_cov100\_t5

*Outcomes of an example for tuning-free conformal prediction with ridge regression.*

---

### Description

A dataset containing the experiment results used in the vignettes.

**Usage**

```
ridge_linear_cov100_t5
```

**Format**

A list with 7 elements: `dim_linear_t5`, `cov.param_linear_fm_t5`, `cov.naive_linear_fm_t5`, `cov.vfcp_linear_fm_t5`, `cov.star_linear_fm_t5`, `cov.cv5_linear_fm_t5`, `cov.efcp_linear_fm_t5`

**dim** dimensions used in the experiment

**cov.param** a matrix with coverages for the prediction regions produced by the parametric method

**cov.naive** a matrix with coverages for the prediction regions produced by naive linear regression method

**cov.vfcp** na matrix with coverages for the prediction regions produced by VFPCP

**cov.star** a matrix with coverages for the prediction regions produced by cross validation with the errors

**cov.cv5** a matrix with coverages for the prediction regions produced by cross-validation with 5 splits

**cov.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")`

---

```
ridge_linear_len100_t3
```

*Outcomes of an example for tuning-free conformal prediction with ridge regression.*

---

**Description**

A dataset containing the experiment results used in the vignettes.

**Usage**

```
ridge_linear_len100_t3
```

**Format**

A list with 6 elements: `len.param_linear_fm_t3`, `len.naive_linear_fm_t3`, `len.vfcp_linear_fm_t3`, `len.star_linear_fm_t3`, `len.cv5_linear_fm_t3`, `len.efcp_linear_fm_t3`

**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_reg`  
`package = "ConformalSmallest")`

---

ridge\_linear\_len100\_t5

*Outcomes of an example for tuning-free conformal prediction with  
 ridge regression.*

---

### 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_reg`  
`package = "ConformalSmallest")`

---

star.fun	<i>Conformal prediction for ridge regression, tuning parameter by minimizing the mean of the residuals</i>
----------	--

---

**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

---

vfcf.fun	<i>Validity first conformal prediction for ridge regression</i>
----------	---

---

**Description**

Validity first conformal prediction for ridge regression

**Usage**

```
vfcf.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  $X_0$ .

**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=rmvtnorm::rmvt(n0,Sigma,df)
X=rmvtnorm::rmvt(n,Sigma,df) #multivariate t distribution
eps=rt(n,df)*(1+sqrt(X[,1]^2+X[,2]^2))
Y=X%%beta+eps
out.vfcq=vfcq.fun(X,Y,X0)
out.vfcq$up
out.vfcq$lo
```

---

vfcq\_cqr

*Validity first conformal prediction for Conformal Quantile Regression*


---

**Description**

Validity first conformal prediction for Conformal Quantile Regression

**Usage**

```
vfcq_cqr(x, y, split, beta_grid, params_grid, alpha = 0.1)
```

**Arguments**

x	A $N \times d$ training matrix
y	A $N \times 1$ 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

vfcpridge

*Validity first conformal prediction for ridge regression***Description**

Validity first conformal prediction for ridge regression

**Usage**

```
vfcpridge(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.vfcpr=vfcpr.fun(X,Y,X0)
out.vfcpr$up
out.vfcpr$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](#)

  

- vfcf.fun, [15](#)
- vfcf\_cqr, [16](#)
- vfcf\_ridge, [17](#)