Package ‘conformalInference.multi’

September 15, 2025

Type Package

Version 1.1.2

Encoding UTF-8

Title Conformal Inference Tools for Regression with Multivariate Response

Description It computes full conformal, split conformal and multi-split conformal prediction regions when the response variable is multivariate (i.e. dimension is greater than one). Moreover, the package also contains plot functions to visualize the output of the full and split conformal functions. To guarantee consistency, the package structure mimics the univariate package 'conformalInference' by Ryan Tibshirani. See Lei, G’sell, Rinaldo, Tibshirani, & Wasserman (2018) <doi:10.1080/01621459.2017.1307116> for full and split conformal prediction in regression, and Barber, Candès, Ramdas, & Tibshirani (2023) <doi:10.1214/23-AOS2276> for extensions beyond exchangeability.

URL `https://github.com/ryantibs/conformal`

License GPL-2 | file LICENSE

Depends R (>= 4.1.0)

Imports future (>= 1.23.0), future.apply (>= 1.8.1), ggplot2 (>= 3.3.5), glmnet, gridExtra (>= 2.3), stats, utils

Suggests mvtnorm

RoxygenNote 7.3.2

NeedsCompilation no

Author Jacopo Diquigiovanni [aut, ths],
Matteo Fontana [aut, ths],
Aldo Solari [aut, ths],
Simone Vantini [aut, ths],
Paolo Vergottini [aut, cre],
Ryan Tibshirani [ctb]

Maintainer Paolo Vergottini <`paolo.vergottini@gmail.com`>

Repository CRAN

Date/Publication 2025-09-15 21:20:10 UTC

Contents

`computing_s_regression` *Compute modulation function for residuals*

Description

Helper function used internally by `conformal.multidim.split` and `conformal.multidim.msplit`.

Usage

```
computing_s_regression(mat_residual, type, alpha, tau)
```

Arguments

`mat_residual`	A vector of residuals obtained via multivariate modeling.
`type`	A string indicating the type of modulation function. Options are "identity", "st-dev", or "alpha-max".
`alpha`	The confidence level for the interval.
`tau`	A number between 0 and 1 used for the randomized version of the algorithm.

Value

Local scoring values for the residuals.

References

Diquigiovanni, Fontana, Vantini (2021), "Conformal Prediction Bands for Multivariate Functional Data"

See Also

`conformal.multidim.split`, `conformal.multidim.msplit` `conformal.multidim.full` *Full Conformal prediction intervals, Multivariate Response*

Description

Compute prediction intervals using full conformal inference with multivariate response

Usage

```
conformal.multidim.full(
  x,
  y,
  x0,
  train.fun,
  predict.fun,
  alpha = 0.1,
  mad.train.fun = NULL,
  mad.predict.fun = NULL,
  score = c("l2", "mahalanobis", "max", "scaled.max"),
  num.grid.pts.dim = 100,
  grid.factor = 1.25,
  verbose = FALSE
)
```

Arguments

`x`	Matrix of features, of dimension (say) n x p.
`y`	Matrix of responses, of length (say) n X q.
`x0`	Matrix of features, each row being a point at which we want to form a prediction interval, of dimension (say) n0 x p.
`train.fun`	A function to perform model training, i.e., to produce an estimator of E(Y|X), the conditional expectation of the response variable Y given features X. Its input arguments should be x: matrix of features, y: vector of responses, and out: the output produced by a previous call to train.fun, at the *same* features x. The function train.fun may (optionally) leverage this returned output for efficiency purposes. See details below.
`predict.fun`	A function to perform prediction for the (mean of the) responses at new feature values. Its input arguments should be out: output produced by train.fun, and newx: feature values at which we want to make predictions.
`alpha`	Miscoverage level for the prediction intervals, i.e., intervals with coverage 1-alpha are formed. Default for alpha is 0.1.
`mad.train.fun`	A function to perform training on the absolute residuals i.e., to produce an estimator of E(R|X) where R is the absolute residual R = |Y - m(X)|, and m denotes the estimator produced by train.fun. This is used to scale the conformal score,

to produce a prediction interval with varying local width. The input arguments to mad.train.fun should be x: matrix of features, y: vector of absolute residuals, and out: the output produced by a previous call to mad.train.fun, at the *same* features x. The function mad.train.fun may (optionally) leverage this returned output for efficiency purposes. See details below. The default for mad.train.fun is NULL, which means that no training is done on the absolute residuals, and the usual (unscaled) conformal score is used. Note that if mad.train.fun is non-NULL, then so must be mad.predict.fun (next).

`mad.predict.fun`
: A function to perform prediction for the (mean of the) absolute residuals at new feature values. Its input arguments should be out: output produced by mad.train.fun, and newx: feature values at which we want to make predictions. The default for mad.predict.fun is NULL, which means that no local scaling is done for the conformal score, i.e., the usual (unscaled) conformal score is used.

`score`
: Method to compute nonconformity measure in the multivariate regime. The user can choose between squared l^2 norm of the residual, mahalanobis depth of the residual, the max norm of the residual, or a scaled max

`num.grid.pts.dim`
: Number of grid points per dimension used when forming the conformal intervals (each num.grid.pts.dim^q points is a trial point). Default is 100.

`grid.factor`
: Expansion factor used to define the grid for the conformal intervals, i.e., the grid points are taken to be equally spaced in between -grid.factor*max(abs(y)) and grid.factor*max(abs(y)). Default is 1.25. In this case (and with exchangeable data, thus unity weights) the restriction of the trial values to this range costs at most 1/(n+1) in coverage. See details below.

`verbose`
: Should intermediate progress be printed out? Default is FALSE.

Details

Due to eventual computational overload the function is restricted to a bivariate y.

This function is based on the package `future.apply` to perform parallelization.

If the data (training and test) are assumed to be exchangeable, the basic assumption underlying conformal prediction, then the probability that a new response value will lie outside of (-max(abs(y)), max(abs(y))), where y is the vector of training responses, is 1/(n+1). Thus the restriction of the trials values to (-grid.factor*max(abs(y)), grid.factor*max(abs(y))), for all choices grid.factor >= 1, will lead to a loss in coverage of at most 1/(n+1). This was also noted in "Trimmed Conformal Prediction for High-Dimensional Models" by Chen, Wang, Ha, Barber (2016) (who use this basic fact as motivation for proposing more refined trimming methods).

Value

A list with the following components: pred, valid_points. The first is a matrix of dimension n0 x q, while the second is a list of length n0, containing in each position a matrix of varying number of rows (depending on which points where accepted by the method) and with a number of columns equal to q + 1. Indeed, valid_points contains the selected points on the y-grid as well as the p-values.

See Also

conformal.multidim.split

Examples

```
n = 4
n0 = 2
p = 2
mu = rep(0,p)
x = mvtnorm::rmvnorm(n, mu)
my_grid <- seq(from=0,to=1,length.out=2)
y = t(apply(x,1,function(u) u[1] + u[2]*cos(6*pi*my_grid)))
x0 = mvtnorm::rmvnorm(n0, mu)
fun=mean_multi()
#fun=lm_multi()

#################################### FULL CONFORMAL

final.full=conformal.multidim.full(x, y, x0, fun$train.fun,
                                fun$predict.fun, score="l2",
                                num.grid.pts.dim=5, grid.factor=1.25,
                                verbose=FALSE)

ppp<-plot_multidim_full(final.full)
```

---

conformal.multidim.msplit

Multi Split conformal prediction intervals with Multivariate Response

---

Description

Compute prediction intervals using Multi Split conformal inference for a multivariate response.

Usage

```
conformal.multidim.msplit(
  x,
  y,
  x0,
  train.fun,
  predict.fun,
  alpha = 0.1,
  split = NULL,
  seed = FALSE,
  randomized = FALSE,
  seed_beta = FALSE,
  verbose = FALSE,
  training_size = NULL,
  score = "max",
  s_type = "st-dev",
  B = 100,
  lambda = 0,
  tau = 1 - (B + 1)/(2 * B)
)
```

Arguments

`x`	Feature matrix of dimension n x p.
`y`	Response matrix of dimension n x q.
`x0`	New points to evaluate, matrix of dimension n0 x p.
`train.fun`	Function to perform model training, producing an estimator of E(Y|X). Input arguments: x (features), y (responses).
`predict.fun`	Function to predict responses at new feature values. Input arguments: out (output from train.fun), newx (new features).
`alpha`	Miscoverage level for prediction intervals. Default 0.1.
`split`	Indices defining the training split. Default NULL (random split).
`seed`	Integer seed for random split. Ignored if split is provided. Default FALSE.
`randomized`	Logical, whether to use the randomized approach. Default FALSE.
`seed_beta`	Seed for the randomized version. Default FALSE.
`verbose`	Logical, print progress? Default FALSE.
`training_size`	Proportion of data used for training. Default 0.5.
`score`	Nonconformity measure to use for the split conformal function.
`s_type`	Type of modulation function: "identity", "st-dev", or "alpha-max". Default "st-dev".
`B`	Number of repetitions. Default 100.
`lambda`	Smoothing parameter. Default 0.
`tau`	Smoothing parameter for intersection method: **tau = 1 - 1/B** Bonferroni intersection method. **tau = 0** Unadjusted intersection. Default 1 - (B + 1)/(2 * B).

Details

This function extends the univariate Multi Split conformal approach to the multivariate case. Parallelization is performed via the `future_sapply` function.

Value

A list with components x0, lo, and up. lo and up are matrices of dimension n0 x q.

References

Solari, Djordjilovic (2021), "Multi Split Conformal Prediction" (baseline for univariate case)

Examples

```
n = 33
n0 = 2
p = 2
mu = rep(0,p)
x = mvtnorm::rmvnorm(n, mu)
my_grid <- seq(from=0,to=1,length.out=2)
y = t(apply(x,1,function(u) u[1] + u[2]*cos(6*pi*my_grid)))
x0 = mvtnorm::rmvnorm(n0, mu)
fun=mean_multi()
#fun=lm_multi()

B=3

final.multi=conformal.multidim.msplit(x=x,y=y, x0=x0,
                                      fun$train.fun, fun$predict.fun,
                       alpha=0.1,
                       split=NULL, seed=FALSE, randomized=FALSE,seed_beta=FALSE,
                       verbose=FALSE, training_size=NULL,s_type="st-dev",B=B,lambda=0,
                       score="l2")
```

`conformal.multidim.split` *Split conformal prediction intervals with Multivariate Response*

Description

Compute prediction intervals using split conformal inference with multivariate response.

Usage

```
conformal.multidim.split(
  x,
  y,
  x0,
  train.fun,
  predict.fun,
  alpha = 0.1,
  split = NULL,
  seed = FALSE,
  randomized = FALSE,
  seed_tau = FALSE,
  verbose = FALSE,
  training_size = 0.5,
  score = "l2",
  s_type = "st-dev",
  mad.train.fun = NULL,
  mad.predict.fun = NULL
)
```

Arguments

`x`	The feature variables, a matrix n x p.
`y`	The matrix of multivariate responses (dimension n x q)
`x0`	The new points to evaluate, a matrix of dimension n0 x p.
`train.fun`	A function to perform model training, i.e., to produce an estimator of E(Y|X), the conditional expectation of the response variable Y given features X. Its input arguments should be x: matrix of features, and y: matrix of responses.
`predict.fun`	A function to perform prediction for the (mean of the) responses at new feature values. Its input arguments should be out: output produced by train.fun, and newx: feature values at which we want to make predictions.
`alpha`	Miscoverage level for the prediction intervals, i.e., intervals with coverage 1-alpha are formed. Default for alpha is 0.1.
`split`	Indices that define the data-split to be used (i.e., the indices define the first half of the data-split, on which the model is trained). Default is NULL, in which case the split is chosen randomly.
`seed`	Integer to be passed to set.seed before defining the random data-split to be used. Default is FALSE, which effectively sets no seed. If both split and seed are passed, the former takes priority and the latter is ignored.
`randomized`	Should the randomized approach be used? Default is FALSE.
`seed_tau`	The seed for the randomized version. Default is FALSE.
`verbose`	Should intermediate progress be printed out? Default is FALSE.
`training_size`	Split proportion between training and calibration set. Default is 0.5.
`score`	The non-conformity measure. It can either be "max", "l2", "mahalanobis". The default is "l2".
`s_type`	The type of modulation function. Currently we have 3 options: "identity","st-dev","alpha-max". Default is "st-dev"
`mad.train.fun`	A function to perform training on the absolute residuals i.e., to produce an estimator of E(R|X) where R is the absolute residual R = |Y - m(X)|, and m denotes the estimator produced by train.fun. This is used to scale the conformal score, to produce a prediction interval with varying local width. The input arguments to mad.train.fun should be x: matrix of features, y: vector of absolute residuals, and out: the output produced by a previous call to mad.train.fun, at the *same* features x. The function mad.train.fun may (optionally) leverage this returned output for efficiency purposes. See details below. The default for mad.train.fun

is NULL, which means that no training is done on the absolute residuals, and the usual (unscaled) conformal score is used. Note that if mad.train.fun is non-NULL, then so must be mad.predict.fun (next).

`mad.predict.fun`
: A function to perform prediction for the (mean of the) absolute residuals at new feature values. Its input arguments should be out: output produced by mad.train.fun, and newx: feature values at which we want to make predictions. The default for mad.predict.fun is NULL, which means that no local scaling is done for the conformal score, i.e., the usual (unscaled) conformal score is used.

Details

If the two mad functions are provided they take precedence over the s_type parameter, and they force a local scoring via the mad function predicted values.

Value

A list with the following components: x0,pred,k_s,s_type,s,alpha,randomized,tau, average_width,lo,up. In particular pred, lo, up are the matrices of dimension n0 x q, k_s is a scalar, s_type is a string, s is a vector of length q, alpha is a scalar between 0 and 1, randomized is a logical value, tau is a scalar between 0 and 1,and average_width is a positive scalar.

References

The s_regression and the "max" score are taken from "Conformal Prediction Bands for Multivariate Functional Data" by Diquigiovanni, Fontana, Vantini (2021).

See Also

`conformal.multidim.full`

Examples

```
sample_size=98

my_grid <- seq(from=0,to=1,length.out=5)
mu <- c(0,0,0)
sigma <- rbind(c(1,0.6,0.6), c(0.6,1,0.6), c(0.6,0.6,1))
mltvnorm3 <- mvtnorm::rmvnorm(sample_size, mu, sigma)
y=t(apply(mltvnorm3,1,function(x) x[1] + x[2]*cos(6*pi*my_grid) + x[3]*sin(6*pi*my_grid)))
x=mltvnorm3 + mvtnorm::rmvt(sample_size, diag(length(mu)))## add noise

n0=10
x0 = mvtnorm::rmvt(n0, diag(length(mu)))

fun=mean_multi()
fun=lm_multi()
fun=elastic.funs()


############################## SPLIT CONFORMAL
```

```
final.point = conformal.multidim.split(x,y[,1:2],x0[1:10,], fun$train.fun, fun$predict.fun,
                              alpha=0.1,
                                 split=NULL, seed=FALSE, randomized=FALSE,seed_tau=FALSE,
                            verbose=FALSE, training_size=0.5,score ="l2",s_type="st-dev")

ppp2<-plot_multidim(final.point)
```

`glmnet.funs` *Elastic net, lasso, ridge regression training and prediction functions.*

Description

Construct training and prediction functions for the elastic net, the lasso, or ridge regression, based on the `glmnet` package, over a sequence of (given or internally computed) lambda values.

Usage

```
elastic.funs(
  gamma = 0.5,
  standardize = TRUE,
  intercept = TRUE,
  lambda = NULL,
  nlambda = 50,
  lambda.min.ratio = 1e-04,
  cv.rule = c("min", "1se")
)

lasso.funs(
  standardize = TRUE,
  intercept = TRUE,
  lambda = NULL,
  nlambda = 50,
  lambda.min.ratio = 1e-04,
  cv.rule = c("min", "1se")
)

ridge.funs(
  standardize = TRUE,
  intercept = TRUE,
  lambda = NULL,
  nlambda = 50,
  lambda.min.ratio = 1e-04,
  cv.rule = c("min", "1se")
)
```

Arguments

`gamma`	Mixing parameter (between 0 and 1) for the elastic net, where 0 corresponds to ridge regression, and 1 to the lasso. Default is 0.5.
`standardize, intercept`	Should the data be standardized, and should an intercept be included? Default for both is TRUE.
`lambda`	Sequence of lambda values over which training is performed. This must be in decreasing order, and — this argument should be used with caution! When used, it is usually best to grab the sequence constructed by one initial call to glmnet (see examples). Default is NULL, which means that the nlambda, lambda.min.ratio arguments will define the lambda sequence (see next).
`nlambda`	Number of lambda values over which training is performed. In particular, the lambda sequence is defined by nlambda log-spaced values between lambda.max and lambda.min.ratio * lambda.max, where lambda.max is the smallest value of lambda at which the solution has all zero components, and lambda.min.ratio is a small fraction (see next). Default is 50.
`lambda.min.ratio`	Small fraction that gets used in conjunction with nlambda to specify a lambda sequence (see above). Default is 1e-4.
`cv.rule`	If the cv argument is TRUE, then cv.rule determines which rule should be used for the predict function, either "min" (the usual rule) or "1se" (the one-standard-error rule). See the glmnet help files for details. Default is "min".

Details

This function is based on the package `glmnet`. Notice that Cross Validation to select the best lambda value is compulsory! The functions lasso.funs and ridge.funs are convenience functions, they simply call elastic.funs with gamma = 1 and gamma = 0, respectively.

Value

A list with three components: train.fun, predict.fun, active.fun. The third function is designed to take the output of train.fun, and reports which features are active for each fitted model contained in this output.

`lm_multi` *Linear Modeling of Multivariate Response*

Description

This model can be used with conformal prediction functions. It returns a training function and a prediction function.

Usage

```
lm_multi()
```

Details

The training function takes as input:

x Feature matrix of dimension n x p.

y Response matrix of dimension n x q.

The prediction function takes as input:

out Output of a previous call to `train.fun`.

newx New feature matrix to evaluate, dimension n0 x p.

Value

A list with two components:

`train.fun`	Function to train the model. Fits a separate linear model for each dimension of the response.
`predict.fun`	Function to make predictions on new data.

See Also

`conformal.multidim.split`

`mean_multi` *Mean of Multivariate Response*

Description

This model can be used with conformal prediction functions. It returns a training function and a prediction function.

Usage

```
mean_multi()
```

Details

The training function takes as input:

x Feature matrix of dimension n x p.

y Response matrix of dimension n x q.

The prediction function takes as input:

out Output of a previous call to `train.fun`.

newx New feature matrix to evaluate, dimension n0 x p.

Value

A list with two components:

`train.fun`	Function to train the model.
`predict.fun`	Function to make predictions on new data.

See Also

`conformal.multidim.split`

`plot_multidim` *Plot Confidence Regions obtained with Split Conformal*

Description

Generate plots for the confidence regions produced by a split multivariate conformal prediction function.

Usage

```
plot_multidim(split, same.scale = FALSE)
```

Arguments

`split`	Output of a split multivariate conformal prediction function.
`same.scale`	Logical. Should all plots use the same y-axis scale? Default is FALSE.

Details

This function uses the `ggplot2` and `gridExtra` packages for visualization.

Value

A list of `ggplot` objects, one for each observation (n0 = length(x0)).

Examples

```
sample_size=98

my_grid <- seq(from=0,to=1,length.out=5)
mu <- c(0,0,0)
sigma <- rbind(c(1,0.6,0.6), c(0.6,1,0.6), c(0.6,0.6,1))
mltvnorm3 <- mvtnorm::rmvnorm(sample_size, mu, sigma)
y=t(apply(mltvnorm3,1,function(x) x[1] + x[2]*cos(6*pi*my_grid) + x[3]*sin(6*pi*my_grid)))
x=mltvnorm3 + mvtnorm::rmvt(sample_size, diag(length(mu)))## add noise

n0=10
x0 = mvtnorm::rmvt(n0, diag(length(mu)))
```

```
fun=mean_multi()
fun=lm_multi()
fun=elastic.funs()


############################## SPLIT CONFORMAL


final.point = conformal.multidim.split(x,y[,1:2],x0[1:10,], fun$train.fun, fun$predict.fun,
                             alpha=0.1,
                                split=NULL, seed=FALSE, randomized=FALSE,seed_tau=FALSE,
                           verbose=FALSE, training_size=0.5,score ="l2",s_type="st-dev")

ppp2<-plot_multidim(final.point)
```

`plot_multidim_full` *Plot Confidence Regions obtained from Full Conformal*

Description

Plot Confidence Regions obtained from Full Conformal

Usage

```
plot_multidim_full(full)
```

Arguments

`full`	It's the output of the multivariate full conformal prediction function

Details

It exploits the package `ggplot2` to better visualize the results.

Value

A list of ggplots (output[[i]] is the i-th observation confidence region).

Examples

```
n = 4
n0 = 2
p = 2
mu = rep(0,p)
x = mvtnorm::rmvnorm(n, mu)
my_grid <- seq(from=0,to=1,length.out=2)
y = t(apply(x,1,function(u) u[1] + u[2]*cos(6*pi*my_grid)))
x0 = mvtnorm::rmvnorm(n0, mu)
fun=mean_multi()
```

```
#fun=lm_multi()

#################################### FULL CONFORMAL

final.full=conformal.multidim.full(x, y, x0, fun$train.fun,
                                fun$predict.fun, score="l2",
                                num.grid.pts.dim=5, grid.factor=1.25,
                                verbose=FALSE)

ppp<-plot_multidim_full(final.full)
```

Index