Package 'conformalInference.fd'

October 12, 2022

Type Package

Title Tools for Conformal Inference for Regression in Multivariate Functional Setting

Version 1.1.1

Description It computes full conformal, split conformal and multi split conformal prediction regions when the response has functional nature. Moreover, the package also contain a plot function to visualize the output of the split conformal.
To guarantee consistency, the package structure mimics the univariate 'conformalInference' package of professor Ryan Tibshirani.
The main references for the code are:
Diquigiovanni, Fontana, and Vantini (2021) <arXiv:2102.06746>,
Diquigiovanni, Fontana, and Vantini (2021) <arXiv:2106.01792>,
Solari, and Djordjilovic (2021) <arXiv:2103.00627>.

URL https://github.com/ryantibs/conformal ,
https://github.com/paolo-vergo/conformalInference.fd

License GPL-2

Depends R (>= 4.1.0)

Imports fda (>= 5.5.1), future (>= 1.23.0), future.apply (>= 1.8.1), ggplot2 (>= 3.3.5), stats, utils, methods, ggnewscale, ggpubr, scales,

Suggests roahd, pbapply

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

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 2022-03-23 11:00:02 UTC

R topics documented:

`bike_log` *Log of all bike rentals in Milan in 2016 form January to March.*

Description

A dataset containing the log of all the bike trips in Milan (using the BikeMi service), in the period from 25th of January to the 6th of March from Duomo to Duomo.

Usage

```
bike_log
```

Format

A list of 41 observed days, each containing a list of 2 components: one which indicates the number of bike trips starting from Duomo at hour t and the other about the number of trips ending in Duomo at time t. Therefore each component is made up by 90 time steps, ranging from 7.00 A.M. to 1.00 A.M. Therefore each component is made up by 90 time steps, ranging from 7.00 A.M. to 1.00 A.M.

start number of departing trips from Duomo

end number of ending trips in Duomo

Source

https://www.mate.polimi.it/biblioteca/add/qmox/19-2019.pdf

bike_regressors *Regressors to model the log of all bike rentals in Milan in 2016.*

Description

A dataset containing temperature and humidity data to model the bike flows from Milano's Duomo district to itself.

Usage

```
bike_regressors
```

Format

A list of 41 observed days, each containing a list of 4 components: a flag indicating whether the day is part of the weekend or not, the amount of rain at a given time t of the day (in mm), the difference between the mean temperature in the last few days and the actual temperature at time t and an interaction term between weekend and rain.

weekend flag for weekend

rain amount of rain (in mm)

dtemp different in temperature w.r.t. the last days

weekend_rain interaction term among rain and weekend

Source

https://www.mate.polimi.it/biblioteca/add/qmox/19-2019.pdf

computing_s_regression *COMPUTING THE MODULATION FUNCTION S*

Description

It computes modulation functions which allows local scaling of the prediction bands .

Usage

```
computing_s_regression(vec_residual, type, alpha, tau, grid_size)
```

Arguments

vec_residual	A vector of the residuals obtained via functional modeling.
type	A string indicating the type of modulation function chosen. The alternatives are "identity","st-dev","alpha-max".
alpha	The value of the confidence interval.
tau	A number between 0 and 1 used for the randomized version of the algorithm.
grid_size	A vector containing the number of grid points in each dimension.

Details

More details can be found in the help of `conformal.fun.split` function.

Value

It returns a the values of a modulation function in each dimension of the response.

---

concurrent *Concurrent Model for Functional Regression*

Description

It is a concurrent model, which may be fed to `conformal.fun.split`.

Usage

```
concurrent()
```

Details

For more details about the structure of the inputs go to split.R

Value

A training and a prediction function.

conformal.fun.jackplus

Functional Jackknife + Prediction Regions

Description

Compute prediction regions using functional Jackknife + inference.

Usage

```
conformal.fun.jackplus(x, t_x, y, t_y, x0, train.fun, predict.fun, alpha = 0.1)
```

Arguments

x	The input variable, a list of n elements. Each element is composed by a list of p vectors(with variable length, since the evaluation grid may change). If x is NULL, the function will sample it from a gaussian.
t_x	The grid points for the evaluation of function x. It is a list of vectors. If the x data type is "fData" or "mfData" is must be NULL.
y	The response variable. It is either, as with x, a list of list of vectors or an fda object (of type fd, fData, mfData).
t_y	The grid points for the evaluation of function y_val. It is a list of vectors. If the y_val data type is "fData" or "mfData" is must be NULL.
x0	The new points to evaluate, a list of n0 elements. Each element is composed by a list of p vectors(with variable length).
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: list of features, and y: list 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.

Details

The work is an extension of the univariate approach to jackknife + inference to a multivariate functional context, exploiting the concept of depth measures.

This function is based on the package future.apply to perform parallelisation. If this package is not installed, then the function will abort.

Value

A list containing lo, up, tn. lo and up are lists of length n0, containing lists of length p, with vectors of lower and upper bounds. tn is the list of the grid evaluations.#'

Examples

```
library(roahd)

N = 3
P= 3
grid = seq( 0, 1, length.out = P )
C = exp_cov_function( grid, alpha = 0.3, beta = 0.4 )
values = generate_gauss_fdata( N,
                                    centerline = sin( 2 * pi * grid ),
                                    Cov = C )
fD = fData( grid, values )
x0=list(as.list(grid))
fun=mean_lists()
x0=list(as.list(grid))
fun=mean_lists()
true.jack = conformal.fun.jackplus (x=NULL,t_x=NULL, y=fD,t_y=NULL,
                                    x0=list(x0[[1]]), fun$train.fun,
                                    fun$predict.fun,alpha=0.1)
```

`conformal.fun.msplit` *Functional Multi Split Conformal Prediction Regions*

Description

Compute prediction regions using functional multi split conformal inference.

Usage

```
conformal.fun.msplit(
  x,
  t_x,
  y,
  t_y,
  x0,
  train.fun,
  predict.fun,
  alpha = 0.1,
  split = NULL,
  seed = FALSE,
  randomized = FALSE,
  seed.rand = FALSE,
  verbose = FALSE,
  rho = NULL,
  s.type = "alpha-max",
  B = 50,
  lambda = 0,
  tau = 0.08
)
```

Arguments

x	The input variable, a list of n elements. Each element is composed by a list of p vectors(with variable length, since the evaluation grid may change). If x is NULL, the function will sample it from a gaussian.
t_x	The grid points for the evaluation of function x. It is a list of vectors. If the x data type is "fData" or "mfData" is must be NULL.
y	The response variable. It is either, as with x, a list of list of vectors or an fda object (of type fd, fData, mfData).
t_y	The grid points for the evaluation of function y_val. It is a list of vectors. If the y_val data type is "fData" or "mfData" is must be NULL.
x0	The new points to evaluate, a list of n0 elements. Each element is composed by a list of p vectors(with variable length).
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: list of features, and y: list 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.rand	The seed for the randomized version of the conformal.split.fun. Default is FALSE.
verbose	Should intermediate progress be printed out? Default is FALSE.
rho	Vector containing the split proportion between training and calibration set. It has B components. Default is 0.5.
s.type	The type of modulation function. Currently we have 3 options: "identity","st-dev","alpha-max".
B	Number of repetitions. Default is 100.
lambda	Smoothing parameter. Default is 0.
tau	It is a smoothing parameter: tau=1-1/B Bonferroni intersection method tau=0 unadjusted intersection Default is 0.05, a value selected through sensitivity analysis .

Details

The work is an extension of the univariate approach to Multi Split conformal inference to a multivariate functional context, exploiting the concept of depth measures.

This function is based on the package future.apply to perform parallelisation. If this package is not installed, then the function will abort.

Value

A list containing lo, up, tn. lo and up are lists of length n0, containing lists of length p, with vectors of lower and upper bounds. tn is the list of the grid evaluations.

References

"Multi Split Conformal Prediction" by Solari, Djordjilovic (2021) is the baseline for the univariate case.

Examples

```
library(roahd)

N = 10
P= 5
grid = seq( 0, 1, length.out = P )
C = exp_cov_function( grid, alpha = 0.3, beta = 0.4 )
values = generate_gauss_fdata( N,
                                    centerline = sin( 2 * pi * grid ),
                                    Cov = C )
fD = fData( grid, values )
x0=list(as.list(grid))
fun=mean_lists()
rrr<-conformal.fun.msplit(x=NULL,t_x=NULL, y=fD,t_y=NULL, x0=list(x0[[1]]),
                          fun$train.fun, fun$predict.fun,alpha=0.2,
                          split=NULL, seed=FALSE, randomized=FALSE,seed.rand=FALSE,
                          verbose=FALSE, rho=NULL,B=2,lambda=0)
```

---

conformal.fun.split *Functional Split Conformal Prediction Intervals*

Description

Compute prediction intervals using split conformal inference.

Usage

```
conformal.fun.split(
  x,
  t_x,
  y,
  t_y,
  x0,
  train.fun,
  predict.fun,
  alpha = 0.1,
  split = NULL,
  seed = FALSE,
  randomized = FALSE,
  seed.rand = FALSE,
  verbose = FALSE,
  rho = 0.5,
  s.type = "st-dev"
)
```

Arguments

x	The input variable, a list of n elements. Each element is composed by a list of p vectors(with variable length, since the evaluation grid may change). If x is NULL, the function will sample it from a gaussian.
t_x	The grid points for the evaluation of function x. It is a list of vectors. If the x data type is "fData" or "mfData" is must be NULL.
y	The response variable. It is either, as with x, a list of list of vectors or an fda object (of type fd, fData, mfData).
t_y	The grid points for the evaluation of function y_val. It is a list of vectors. If the y_val data type is "fData" or "mfData" is must be NULL.
x0	The new points to evaluate, a list of n0 elements. Each element is composed by a list of p vectors(with variable length).
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: list of features, and y: list 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.rand	The seed for the randomized version.Default is FALSE.
verbose	Should intermediate progress be printed out? Default is FALSE.
rho	Split proportion between training and calibration set. Default is 0.5.
s.type	The type of modulation function. Currently we have 3 options: "identity","st-dev","alpha-max". Default is "std-dev".

Value

A list with the following components: t,pred,average_width,lo, up. t is a list of vectors, pred has the same interval structure of y_val, but the outside list is of length n0, lo and up are lists of length n0 of lists of length p, each containing a vector of lower and upper bounds respectively.

Examples

```
###  mfData #

library(roahd)

N = 10
P= 5
grid = seq( 0, 1, length.out = P )
C = exp_cov_function( grid, alpha = 0.3, beta = 0.4 )
Data_1 = generate_gauss_fdata( N, centerline = sin( 2 * pi * grid ), Cov = C )
Data_2 = generate_gauss_fdata( N, centerline = log(1+ 2 * pi * grid ), Cov = C )
mfD=mfData( grid, list( Data_1, Data_2 ) )
x0=list(as.list(grid))
fun=mean_lists()
final.mfData = conformal.fun.split(NULL,NULL, mfD,NULL, x0, fun$train.fun, fun$predict.fun,
                              alpha=0.2,
                              split=NULL, seed=FALSE, randomized=FALSE,seed.rand=FALSE,
                              verbose=TRUE, rho=0.5,s.type="identity")
```

---

mean_lists *Mean of Functional Data*

Description

This model, which averages functional data, is a fed to a Functional Conformal Prediction function.

Usage

```
mean_lists()
```

Details

For more details about the structure of the inputs go to the help of `conformal.fun.split`

Value

It outputs a training function and a prediction function.

`plot_fun` *Plot Functional Split Conformal Confidence Bands*

Description

The function plots the confidence bands provided by the `conformal.fun.split` #'function, `conformal.fun.msplit` and `conformal.fun.jackplus`.

Usage

```
plot_fun(
  out,
  y0 = NULL,
  ylab = NULL,
  titles = NULL,
  date = NULL,
  ylim = NULL,
  fillc = "red"
)
```

Arguments

`out`	The output of the split/msplit/jackknife+ function.
`y0`	The true values at x0.
`ylab`	The label for the y-axes.
`titles`	The title for the plot.
`date`	A vector of dates.
`ylim`	A vector containing the extremes for the y-axes.
`fillc`	A string of color.

Details

It exploits the package `ggplot`, `ggarrange` and `annotate_figure`. to better visualize the results. It outputs n0=length(x0) plots.

It plots, for each value in x0, the predicted functional value and bands in all the dimensions of the multivariate functional response.

Value

None

Index