Package: conformalInference.multi (via r-universe)

October 21, 2024

Type Package

Title Conformal Inference Tools for Regression with Multivariate Response

Version 1.1.1

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 contain plot functions to visualize the output of the full and split conformal functions. To guarantee consistency, the package structure mimics the univariate 'conformalInference' package of professor Ryan Tibshirani. The main references for the code are: Lei et al. (2016) <arXiv:1604.04173>, 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.multi`

License GPL-2

Depends R (>= 4.1.0)

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

Suggests mvtnorm, 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>

Date/Publication 2022-03-16 16:40:06 UTC

Repository https://paolo-vergo.r-universe.dev

RemoteUrl https://github.com/cran/conformalInference.multi

RemoteRef HEAD

RemoteSha 12671e6e379631a2e819181063a25535c7898c18

Contents

`bikeMi` *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, as well as meteorological data.

Usage

```
bikeMi
```

Format

A data frame with 41 rows and 6 variables:

start number of trips started in Duomo a given day

end number of trips ended in Duomo a given day

we is weekend?If true, than we is 1

rain mean amount of rain during the day

dtemp difference between average temperature of the day and of the period

we_rain interaction between weekend and rain

Source

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

`computing_s_regression` *Computing modulation function for residuals.*

Description

It computes values for local scoring.

Usage

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

Arguments

`mat_residual`	A vector of the residuals obtained via multivariate 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

Details

It is an helper function for `conformal.multidim.split` and `conformal.multidim.msplit`

Value

It returns local scoring values for the residuals.

References

"Conformal Prediction Bands for Multivariate Functional Data" by Diquigiovanni, Fontana, Vantini (2021) <arXiv:2106.01792>.

conformal.multidim.full *Full Conformal Prediction Regions, 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 = "l2",
  s.type = "st-dev",
  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.
`s.type`	The type of modulation function. Currently we have 3 options: "identity","st-dev". Default is "st-dev"
`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 x max(abs(y)) and grid.factor x 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 parallelisation.

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 x max(abs(y)), grid.factor x 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) <arXiv:1611.09933> (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=25
p=4
q=2


mu=rep(0,p)
x = mvtnorm::rmvnorm(n, mu)
beta<-sapply(1:q, function(k) c(mvtnorm::rmvnorm(1,mu)))
y = x%*%beta + t(mvtnorm::rmvnorm(q,1:n))
x0=x[ceiling(0.9*n):n,]
y0=y[ceiling(0.9*n):n,]

n0<-nrow(y0)
q<-ncol(y)


fun=mean_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(final.full)
```

---

conformal.multidim.jackplus

Multivariate Response Jackknife + Prediction Regions

Description

Compute prediction regions using multivariate Jackknife + inference.

Usage

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

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.

Details

The work is an extension of the univariate approach to jackknife + inference to a multivariate 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 with length n0, giving the lower and upper bounds for each observation.

Examples

```
## One instance

n=50
p=3
q=2
mu=rep(0,p)
x = mvtnorm::rmvnorm(n, mu)
beta<-sapply(1:q, function(k) c(mvtnorm::rmvnorm(1,mu)))
y = x%*%beta + t(mvtnorm::rmvnorm(q,1:n))
x0=matrix(x[n,],nrow=1)
y0=matrix(y[n,],nrow=1)
n0<-nrow(y0)
funs=lm_multi()

sol<-conformal.multidim.jackplus(x,y,x,train.fun = funs$train.fun,
                                       predict.fun = funs$predict.fun,alpha=0.05)

sol
```

`conformal.multidim.msplit`
Multi Split Conformal Prediction Regions with Multivariate Response

Description

Compute prediction intervals using Multi Split conformal inference with multivariate response.

Usage

```
conformal.multidim.msplit(
  x,
  y,
  x0,
  train.fun,
  predict.fun,
  alpha = 0.1,
  split = NULL,
  seed = FALSE,
  randomized = FALSE,
  seed.rand = FALSE,
  verbose = FALSE,
  rho = NULL,
  score = "max",
  s.type = "st-dev",
  B = 100,
  lambda = 0,
  tau = 0.1,
  mad.train.fun = NULL,
  mad.predict.fun = NULL
)
```

Arguments

`x`	The feature variables, a matrix nxp.
`y`	The matrix of multivariate responses (dimension nxq)
`x0`	The new points to evaluate, a matrix of dimension n0xp.
`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.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.
`score`	The chosen score for the split conformal function.
`s.type`	The type of modulation function. Currently we have 3 options: "identity","st-dev","alpha-max". Default is "std-dev"
`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 1-(B+1)/(2*B).
`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

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

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

Value

A list with length n0, giving the lower and upper bounds for each observation.

References

"Multi Split Conformal Prediction" by Solari, Djordjilovic (2021) <arXiv:2103 .00627> is the baseline for the univariate case.

Examples

```
set.seed(12345)

n=200
p=4
q=2
mu=rep(0,p)
x = mvtnorm::rmvnorm(n, mu)
beta<-sapply(1:q, function(k) c(mvtnorm::rmvnorm(1,mu)))
y = x%*%beta + t(mvtnorm::rmvnorm(q,1:n))
x0=matrix(x[n,],nrow=1)
y0=matrix(y[n,],nrow=1)
n0<-nrow(y0)
q<-ncol(y)
B=100
funs=lm_multi()


sol<-conformal.multidim.msplit(x,y, x0, train.fun = funs$train.fun,
                                          predict.fun = funs$predict.fun, alpha=0.05,
                                          split=NULL, seed=FALSE, randomized=FALSE,
                               seed.rand=FALSE,
                                          verbose=FALSE, rho=NULL,score = "max",
                                          s.type = "st-dev",B=B,lambda=0,
                                          tau = 0.1,mad.train.fun = NULL,
                                          mad.predict.fun = NULL)

sol
```

---

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.rand = FALSE,
  verbose = FALSE,
  rho = 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.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.
`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

```
n=50
p=4
q=2

mu=rep(0,p)
x = mvtnorm::rmvnorm(n, mu)
beta<-sapply(1:q, function(k) c(mvtnorm::rmvnorm(1,mu)))
y = x%*%beta + t(mvtnorm::rmvnorm(q,1:n))
x0=x[ceiling(0.9*n):n,]
y0=y[ceiling(0.9*n):n,]

n0<-nrow(y0)
```

```
q<-ncol(y)

fun=mean_multi()

final.point = conformal.multidim.split(x,y,x0, fun$train.fun, fun$predict.fun,
                             alpha=0.1,
                               split=NULL, seed=FALSE, randomized=FALSE,seed.rand=FALSE,
                                verbose=FALSE, rho=0.5,score ="l2",s.type="st-dev")

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

elastic.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 packages `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 is fed to `conformal.multidim.full`, `conformal.multidim.split`, and `conformal.multidim.msplit`. It outputs a training function and a prediction function.

Usage

```
lm_multi()
```

Details

The training function takes as input:

x The feature matrix (dim n x p) y The response matrix (dim n x q)

The predict function, instead, takes as input:

out The output of a previous call to train.fun newx The new features to evaluate (i.e. an n0 x p matrix) Here I defined an lm model for every dimension of the responses (q).

Value

A list with the training function and the prediction function.

`mean_multi` *Mean of Multivariate Response*

Description

This model is fed to `conformal.multidim.full`, `conformal.multidim.split`, and `conformal.multidim.msplit`. It outputs a training function and a prediction function.

Usage

```
mean_multi()
```

Details

The training function takes as input:

x The feature matrix (dim n x p) y The response matrix (dim n x q)

The predict function, instead, takes as input:

out The output of a previous call to train.fun newx The new features to evaluate (i.e. an n0 x p matrix)

Value

A list with the training function and the prediction function.

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

Description

Plot Confidence Regions obtained with Split Conformal

Usage

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

Arguments

`out`	The output of a prediction function.
`same.scale`	Should I force the same scale for all the y-axis ? Default is FALSE.

Details

It exploits the package `ggplot2`, `gridExtra` and `hrbrthemes` to better visualize the results.

Value

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

Examples

```
n=50
p=4
q=2

mu=rep(0,p)
x = mvtnorm::rmvnorm(n, mu)
beta<-sapply(1:q, function(k) c(mvtnorm::rmvnorm(1,mu)))
y = x%*%beta + t(mvtnorm::rmvnorm(q,1:n))
x0=x[ceiling(0.9*n):n,]
y0=y[ceiling(0.9*n):n,]

n0<-nrow(y0)
q<-ncol(y)

fun=mean_multi()

final.point = conformal.multidim.split(x,y,x0, fun$train.fun, fun$predict.fun,
                             alpha=0.1,
                               split=NULL, seed=FALSE, randomized=FALSE,seed.rand=FALSE,
                                verbose=FALSE, rho=0.5,score ="l2",s.type="st-dev")

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

```
n=25
p=4
q=2


mu=rep(0,p)
x = mvtnorm::rmvnorm(n, mu)
beta<-sapply(1:q, function(k) c(mvtnorm::rmvnorm(1,mu)))
y = x%*%beta + t(mvtnorm::rmvnorm(q,1:n))
x0=x[ceiling(0.9*n):n,]
y0=y[ceiling(0.9*n):n,]

n0<-nrow(y0)
q<-ncol(y)


fun=mean_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(final.full)
```

Index