Package ‘profoc’

March 25, 2024

Type Package
Title Probabilistic Forecast Combination Using CRPS Learning
Version 1.3.2
Date 2024-03-25

License GPL (>= 3)
Encoding UTF-8
Depends R (>= 4.3.0)
Imports Rcpp (>= 1.0.5), Matrix, abind, methods, lifecycle, generics, tibble, ggplot2
LinkingTo Rcpp, RcppArmadillo (>= 0.10.7.5.0), RcppProgress, splines2 (>= 0.4.4), rcppTimer (>= 1.1.0)
BugReports https://github.com/BerriJ/profoc/issues
RoxygenNote 7.3.1
Language en-US
Suggests testthat (>= 3.0.0), gamlss.dist, knitr, rmarkdown, dplyr
Config/testthat/edition 3
VignetteBuilder knitr
NeedsCompilation yes
Author Jonathan Berrisch [aut, cre] (<https://orcid.org/0000-0002-4944-9074>), Florian Ziel [aut] (<https://orcid.org/0000-0002-2974-2660>)
Maintainer Jonathan Berrisch <Jonathan@Berrisch.biz>
Description

Use multiple online-aggregation algorithms to combine probabilistic forecasts using CRPS Learning as described in Berrisch, Ziel: "CRPS Learning", 2021. The primary function of this package is called online.

doi:10.1016/j.jeconom.2021.11.008
Details

Index of help topics:

- autoplot.batch: Autoplot method for batch models
- autoplot.online: Autoplot method for online models
- batch: Probabilistic Forecast Combination - Batch
- conline: Create an conline Object from the conline C++ Class
- init_experts_list: Create experts list to be used in conline class
- make_basis_mats: Create a List of Basis Matrices
- make_hat_mats: Create a List of Hat Matrices
- make_knots: Create a vector of knots for splines
- online: Probabilistic Forecast Combination - Online
- oracle: Probabilistic Forecast Combination - Oracle
- penalty: B-Spline penalty
- plot.batch: Plot method for batch models
- plot.online: Plot method for online models
- post_process_model: Post Process Data from conline Class
- predict.online: Predict method for online models
- print.batch: Print method for batch models
- print.online: Print method for online models
- profoc-package: Package Info
- splines2_basis: Create B-Spline basis
- summary.online: Summary method for online models
- tidy.online.experts_loss: Tidy the Experts' losses of an Online object
- tidy.online.forecaster_loss: Tidy the Experts' losses of an Online object
- tidy.online.predictions: Tidy the Predictions of an Online object
- tidy.online.weights: Tidy the Weights of an Online object
- update.online: Update method for online models

Author(s)

Maintainer: Jonathan Berrisch
Co-Author: Florian Ziel

References

Berrisch, Ziel: "CRPS Learning", 2021
doi:10.1016/j.jeconom.2021.11.008
doi:10.48550/arXiv.2102.00968

See Also

Source Code: https://github.com/BerriJ/profoc
BugReports: https://github.com/BerriJ/profoc/issues
autplot.batch  
*Autoplot method for batch models*

**Description**

Plots the most recent weights in each quantile using ggplot2.

**Usage**

```r
## S3 method for class 'batch'
autplot(object, ...)
```

**Arguments**

- `object`  
  Object of class inheriting from 'batch'
- `...`  
  Further arguments are ignored

autplot.online  
*Autoplot method for online models*

**Description**

Plots the most recent weights in each quantile using ggplot2.

**Usage**

```r
## S3 method for class 'online'
autplot(object, ...)
```

**Arguments**

- `object`  
  Object of class inheriting from 'online'
- `...`  
  Further arguments are ignored
Description

Returns predictions and weights calculated by sequential numeric optimization. The optimization is done stepwise, always calculating a one-step-ahead forecast.

[Experimental]

Usage

batch(
  y,
  experts,
  tau = 1:dim(experts)[2]/(dim(experts)[2] + 1),
  affine = FALSE,
  positive = FALSE,
  intercept = FALSE,
  debias = TRUE,
  lead_time = 0,
  initial_window = 30,
  rolling_window = initial_window,
  loss_function = "quantile",
  loss_parameter = 1,
  qw_crps = FALSE,
  b_smooth = list(knots = length(tau), mu = 0.5, sigma = 1, nonc = 0, tailweight = 1, deg = 1, periodic = FALSE),
  p_smooth = list(knots = length(tau), mu = 0.5, sigma = 1, nonc = 0, tailweight = 1, deg = 1, ndiff = 1.5, lambda = -Inf, periodic = FALSE),
  forget = 0,
  soft_threshold = -Inf,
  hard_threshold = -Inf,
  fixed_share = 0,
  parametergrid_max_combinations = 100,
  parametergrid = NULL,
  forget_past_performance = 0,
  allow_quantile_crossing = FALSE,
  trace = TRUE
)

Arguments

y A numeric matrix of realizations. In probabilistic settings a matrix of dimension Tx1, in multivariate settings a TxD matrix. In the latter case, each slice of the expert’s array gets evaluated using the corresponding column of the y matrix.

experts An array of predictions with dimension (Observations, Quantiles, Experts).
tau
A numeric vector of probabilities.
affine
Defines whether weights are summing to 1 or not. Defaults to FALSE.
positive
Defines if a positivity constraint is applied to the weights. Defaults to FALSE.
intercept
Determines if an intercept is added, defaults to FALSE. If true, a new first expert is added, always predicting 1.
debias
Defines whether the intercepts weight is constrained or not. If TRUE (the default), the intercept weight is unconstrained. Only affects the results if affine and or positive is set to TRUE. If FALSE, the intercept is treated as an expert.
lead_time
offset for expert forecasts. Defaults to 0, which means that experts forecast t+1 at t. Setting this to h means experts predictions refer to t+1+h at time t. The weight updates delay accordingly.
initial_window
Defines the size of the initial estimation window.
rolling_window
Defines the size of the rolling window. Defaults to the value of initial_window. Set it to the number of observations to receive an expanding window.
loss_function
Either "quantile", "expectile" or "percentage".
loss_parameter
Optional parameter scaling the power of the loss function.
qw_crps
Decides whether the sum of quantile scores (FALSE) or the quantile weighted CRPS (TRUE) should be minimized. Defaults to FALSE. Which corresponds to Berrisch & Ziel (2021)
b_smooth
A named list determining how the B-Spline matrices for probabilistic smoothing are created. Default corresponds to no probabilistic smoothing. See details.
p_smooth
A named list determining how the hat matrices for probabilistic P-Spline smoothing are created. Default corresponds to no smoothing. See details.
forget
Adds an exponential forgetting to the optimization. Past observations will get less influence on the optimization. Defaults to 0, which corresponds to no forgetting.
soft_threshold
If specified, the following soft threshold will be applied to the weights: w = sgn(w)*max(abs(w)-t,0) where t is the soft_threshold parameter. Defaults to -inf, which means that no threshold will be applied. If all expert weights are thresholded to 0, a weight of 1 will be assigned to the expert with the highest weights prior to thresholding. Thus soft_threshold = 1 leads to the 'follow the leader' strategy if method is set to "ewa".
hard_threshold
If specified, the following hard thresholding will be applied to the weights: w = w*(abs(w)>t) where t is the threshold_hard parameter. Defaults to -inf, which means that no threshold will be applied. If all expert weights are thresholded to 0, a weight of 1 will be assigned to the expert with the highest weight prior to thresholding. Thus hard_threshold = 1 leads to the 'follow the leader' strategy if method is set to "ewa".
fixed_share
Amount of fixed share to be added to the weights. Defaults to 0. 1 leads to uniform weights.
parametergrid_max_combinations
Integer specifying the maximum number of parameter combinations that should be considered. If the number of possible combinations exceeds this threshold, the maximum allowed number is randomly sampled. Defaults to 100.
parametergrid  User supplied grid of parameters. Can be used if not all combinations of the input vectors should be considered. Must be a matrix with 13 columns (online) or 12 columns batch with the following order: basis_knot_distance, basis_knot_distance_power, basis_deg, forget_regret, soft_threshold, hard_threshold, fixed_share, p_smooth_lambda, p_smooth_knot_distance, p_smooth_knot_distance_power, p_smooth_deg, p_smooth_ndiff, gamma.

forget_past_performance  Share of past performance not to be considered, resp. to be forgotten in every iteration of the algorithm when selecting the best parameter combination. Defaults to 0.

allow_quantile_crossing  Shall quantile crossing be allowed? Defaults to false, which means that predictions are sorted in ascending order.

trace  Print a progress bar to the console? Defaults to TRUE.

Details

batch selects various parameters automatically based on the past loss. For this, the parameters smoothing parameters (see below) can be specified as numeric vectors containing values to consider.

This package offers two options for smoothing (Basis Smoothing and P-Splines). Parameters b_smooth and p_smooth take named lists to create the corresponding basis and hat matrices. The arguments are: knots which determines the number of knots to be created, mu, sigma, sigma, nonc, tailweight correspond to to parameters of the beta distribution, which defines how the knots are distributed (see ?make_knots for details) the defaults will create an equidistant knot sequence, deg sets the degree of the spline function and also influences how many outer knots will be used and periodic which determines whether the spline basis will be periodic. It’s possible to provide vectors of values for each of these parameters. In that case, all parameter combinations will be used to create the respective matrices and all candidates will be considered during online-learning. In addition to the inputs mentioned before p_smooth requires ndiff which determines the degree of differentiation applied to the basis-matrix (can take any value between and including 1 and 2), lambda which determines the degree of penalization applied to the smoothing, higher values will give smoother weight functions. As for the other parameters, it is possible to provide multiple values.

Value

Returns weights and corresponding predictions. It is possible to impose a convexity constraint to the weights by setting affine and positive to TRUE.

Examples

## Not run:
T <- 50  # Observations
N <- 2  # Experts
P <- 9  # Quantiles
prob_grid <- 1:P / (P + 1)

y <- rnorm(n = T)  # Realized
experts <- array(dim = c(T, P, N))  # Predictions
for (t in 1:T) {
    experts[t, , 1] <- qnorm(prob_grid, mean = -1, sd = 1)
    experts[t, , 2] <- qnorm(prob_grid, mean = 3, sd = sqrt(4))
}

model <- batch(
    y = matrix(y),
    experts = experts,
    p_smooth = list(lambda = 10)
)

print(model)
plot(model)
autoplot(model)

## End(Not run)

---

**conline**

Create an conline Object from the conline C++ Class

**Description**

Allows for the creation of a Online Object in C++ from R using the C++ conline class.

**Value**

A conline object from the C++ conline Class.

**Examples**

```r
conline_obj <- new(conline)
```

---

**init_experts_list**

Create experts list to be used in conline class

**Description**

This function works in conjunction with the conline class. It takes a matrix of experts and a matrix of outcomes and returns a list of experts which fulfills all properties that are needed for passing it to the an instance of conline.

**Usage**

`init_experts_list(experts, y, output_with_names = FALSE)`
Arguments

experts array of predictions with dimension T x D x P x K (Observations x Variables x Quantiles x Experts) or T x D x K or T x P x K.
y A matrix of outcomes with dimension T x D.
output_with_names Defaults to FALSE. If TRUE, the function returns a list with the experts list, the names of the variables (dnames) and the names of the experts (enames).

make\_basis\_mats Create a List of Basis Matrices

Description

This function creates a list of basis matrices and the corresponding parameters. It is used in online() to create the basis matrices for basis smoothing.

Usage

make\_basis\_mats(
x, 
n = length(x), 
mu = 0.5, 
sigma = 1, 
nonc = 0, 
tailw = 1, 
deg = 1, 
periodic = FALSE, 
idx = NULL, 
params = NULL
)

Arguments

x The predictor variable
n Number of knots
mu Beta distribution location parameter
sigma Beta distribution scale parameter
nonc Beta distribution noncentrality parameter
tailw Tailweight
deg Degree of splines
periodic Create periodic basis
idx make\_basis\_mats() will create a grid containing all combinations of the parameters. If idx is set, this grid will be subbed to the rows specified by idx.
params Instead of the arguments above, a grid (data.frame or named matrix) of parameters can be passed directly.
make_hat_mats  Create a List of Hat Matrices

Description

This function creates a list of hat matrices and the corresponding parameters. It is used in online() to create the hat matrices for penalized smoothing.

Usage

```
make_hat_mats(
  x,
  n = length(x),
  mu = 0.5,
  sigma = 1,
  nonc = 0,
  tailw = 1,
  deg = 1,
  ndiff = 1.5,
  lambda = -Inf,
  periodic = FALSE,
  idx = NULL,
  params = NULL
)
```

Arguments

- **x**: The predictor variable
- **n**: Number of knots
- **mu**: Beta distribution location parameter
- **sigma**: Beta distribution scale parameter
- **nonc**: Beta distribution noncentrality parameter
- **tailw**: Tailweight
- **deg**: Degree of splines
- **ndiff**: Sets the degree of the differencing matrix for creating the penalty
- **lambda**: Penalty parameter (higher values lead to higher penalty)
- **periodic**: Create periodic penalty
- **idx**: `make_hat_mats()` will create a grid containing all combinations of the parameters. If `idx` is set, this grid will be subsetted to the rows specified by `idx`.
- **params**: Instead of the arguments above, a grid (data.frame or named matrix) of parameters can be passed directly.
**make_knots**

*Create a vector of knots for splines*

**Description**

This function creates a knot vector for splines. The knots are distributed according to a beta distribution. The first input defines the number of inner knots. The total number of knots is \( n + 2 \times \text{order} \).

**Usage**

```r
make_knots(n, mu = 0.5, sig = 1, nonc = 0, tailw = 1, deg = 1)
```

**Arguments**

- `n`: Number of knots
- `mu`: Beta distribution location parameter
- `sig`: Beta distribution scale parameter
- `nonc`: Beta distribution noncentrality parameter
- `tailw`: Tailweight
- `deg`: Degree of splines

**online**

*Probabilistic Forecast Combination - Online*

**Description**

Returns predictions and weights calculated by online-learning algorithms using CRPS Learning.

**[Stable]**

**Usage**

```r
online(
  y,
  experts,
  tau,
  lead_time = 0,
  loss_function = "quantile",
  loss_parameter = 1,
  loss_gradient = TRUE,
  method = "bewa",
  b_smooth_pr = list(knots = P, mu = 0.5, sigma = 1, nonc = 0, tailweight = 1, deg = 1, periodic = FALSE),
  p_smooth_pr = list(knots = P, mu = 0.5, sigma = 1, nonc = 0, tailweight = 1, deg = 1, periodic = FALSE)
)
```
ndiff = 1.5, lambda = -Inf, periodic = FALSE),
b_smooth_mv = list(knots = D, mu = 0.5, sigma = 1, nonc = 0, tailweight = 1, deg = 1,
periodic = FALSE),
p_smooth_mv = list(knots = D, mu = 0.5, sigma = 1, nonc = 0, tailweight = 1, deg = 1,
ndiff = 1.5, lambda = -Inf, periodic = FALSE),
forget_regret = 0,
soft_threshold = -Inf,
hard_threshold = -Inf,
fixed_share = 0,
gamma = 1,
parametergrid_max_combinations = 100,
parametergrids = list(general = NULL, b_smooth_pr = NULL, p_smooth_pr = NULL,
  b_smooth_mv = NULL, p_smooth_mv = NULL),
forget_past_performance = 0,
save_past_performance = FALSE,
save_predictions_grid = FALSE,
allow_quantile_crossing = FALSE,
init = NULL,
loss = NULL,
regret = NULL,
trace = TRUE,
get_timings = FALSE
)

Arguments

y    A numeric matrix of realizations. In probabilistic settings a matrix of dimension 
      Tx1, in multivariate settings a TxD matrix. In the latter case, each slice of the 
      expert’s array gets evaluated using the corresponding column of the y matrix.
experts  An array of predictions with dimension T x D x P x K (Observations x Variables 
      x Quantiles x Experts) or T x D x K or T x P x K.
tau     A numeric vector of probabilities.
lead_time offset for expert forecasts. Defaults to 0, which means that experts forecast t+1 
         at t. Setting this to h means experts predictions refer to t+1+h at time t. The 
         weight updates delay accordingly.
loss_function  Either "quantile", "expectile" or "percentage".
loss_parameter Optional parameter scaling the power of the loss function.
loss_gradient  Determines if a linearized version of the loss is used.
method      One of "boa", "bewa", "ml_poly" or "ewa". Where "bewa" refers to a mixture of 
             boa and ewa, including the second order refinement of boa, but updating weights 
             with the simple exponential weighting.
b_smooth_pr A named list determining how the B-Spline matrices for probabilistic smoothing 
             are created. Default corresponds to no probabilistic smoothing. See details.
p_smooth_pr A named list determining how the hat matrices for probabilistic P-Spline smoothing 
             are created. Default corresponds to no smoothing. See details.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>b_smooth_mv</code></td>
<td>A named list determining how the B-Spline matrices for multivariate smoothing are created. Default corresponds to no probabilistic smoothing. See details.</td>
</tr>
<tr>
<td><code>p_smooth_mv</code></td>
<td>A named list determining how the hat matrices for probabilistic P-Spline smoothing are created. Default corresponds to no smoothing. See details.</td>
</tr>
<tr>
<td><code>forget_regret</code></td>
<td>Share of past regret not to be considered, resp. to be forgotten in every iteration of the algorithm. Defaults to 0.</td>
</tr>
<tr>
<td><code>soft_threshold</code></td>
<td>If specified, the following soft threshold will be applied to the weights: ( w = \text{sgn}(w) \times \max(\text{abs}(w) - t, 0) ) where ( t ) is the soft_threshold parameter. Defaults to -inf, which means that no threshold will be applied. If all expert weights are thresholded to 0, a weight of 1 will be assigned to the expert with the highest weights prior to thresholding. Thus soft_threshold = 1 leads to the 'follow the leader' strategy if method is set to &quot;ewa&quot;.</td>
</tr>
<tr>
<td><code>hard_threshold</code></td>
<td>If specified, the following hard thresholding will be applied to the weights: ( w = w \times (\text{abs}(w) &gt; t) ) where ( t ) is the threshold_hard parameter. Defaults to -inf, which means that no threshold will be applied. If all expert weights are thresholded to 0, a weight of 1 will be assigned to the expert with the highest weight prior to thresholding. Thus hard_threshold = 1 leads to the 'follow the leader' strategy if method is set to &quot;ewa&quot;.</td>
</tr>
<tr>
<td><code>fixed_share</code></td>
<td>Amount of fixed share to be added to the weights. Defaults to 0. 1 leads to uniform weights.</td>
</tr>
<tr>
<td><code>gamma</code></td>
<td>Scaling parameter for the learning rate.</td>
</tr>
<tr>
<td><code>parametergrid_max_combinations</code></td>
<td>Integer specifying the maximum number of parameter combinations that should be considered. If the number of possible combinations exceeds this threshold, the maximum allowed number is randomly sampled. Defaults to 100.</td>
</tr>
<tr>
<td><code>parametergrids</code></td>
<td>User supplied grids of parameters. Can be used if not all combinations of the input vectors should be considered. Must be a named list of five matrices. The matrices in list must be named as: &quot;general&quot;, &quot;b_smooth_pr&quot;, &quot;b_smooth_mv&quot;, &quot;p_smooth_pr&quot;, &quot;p_smooth_mv&quot;. The &quot;general&quot; matrix must contain 11 named columns: &quot;forget_regret&quot;, &quot;soft_threshold&quot;, &quot;hard_threshold&quot;, &quot;fixed_share&quot;, &quot;basis_pr_idx&quot;, &quot;basis_mv_idx&quot;, &quot;hat_pr_idx&quot;, &quot;hat_mv_idx&quot;, &quot;gamma&quot;, &quot;loss_share&quot;, &quot;regret_share&quot;. The matrices determining the basis smoothing (b_smooth_pr, b_smooth_mv) must contain the following named columns: n, mu, sigma, nonc, tailw, deg, periodic. In addition to the columns of the basis smoothing matrices, the matrices determining the penalized smoothing (p_smooth_pr, p_smoothMV) must contain the following columns: diff, lambda. The *_idx columns in the general matrix determine which row of the corresponding smoothing matrix is used.</td>
</tr>
<tr>
<td><code>forget_past_performance</code></td>
<td>Share of past performance not to be considered, resp. to be forgotten in every iteration of the algorithm when selecting the best parameter combination. Defaults to 0.</td>
</tr>
<tr>
<td><code>save_past_performance</code></td>
<td>Whether or not the past performance w.r.t to the considered parameter grid should be reported or not. Defaults to FALSE to save memory. Setting it to TRUE can be memory intensive depending on the data and the considered grid.</td>
</tr>
</tbody>
</table>
save_predictions_grid
Whether or not all predictions w.r.t to the considered parameter grid should be reported or not. Defaults to FALSE. Setting it to TRUE can be memory intensive depending on the data and the considered grid.

allow_quantile_crossing
Shall quantile crossing be allowed? Defaults to false, which means that predictions are sorted in ascending order.

init
A named list containing "init_weights": Array of dimension DxPxK used as starting weights. "R0" a matrix of dimension PxK or 1xK used as starting regret.

loss
User specified loss array. Can also be a list with elements "loss_array" and "share", share mixes the provided loss with the loss calculated by profoc. 1 means, only the provided loss will be used. share can also be vector of shares to consider.

regret
User specified regret array. If specific, the regret will not be calculated by profoc. Can also be a list with elements "regret_array" and "share", share mixes the provided regret with the regret calculated by profoc. 1 means, only the provided regret will be used. share can also be vector of shares to consider.

trace
Print a progress bar to the console? Defaults to TRUE.

get_timings
Whether or not to return timings. Defaults to FALSE. If set to true a dataframe times will be written to your global environment.

Details
online selects various parameters automatically based on the past loss. For this, lambda, forget, fixed_share, gamma, and the smoothing parameters (see below) can be specified as numeric vectors containing values to consider.

This package offers two options for smoothing (Basis Smoothing and P-Splines). Both options can be used to smooth the weights over dimension D (covariates) or P (quantiles) or both. Parameters b_smooth_pr and b_smooth_mv take named lists to create the corresponding basis matrices. The arguments are: knots which determines the number of knots to be created, mu, sigma, sigma, nonc, tailweight correspond to to parameters of the beta distribution, which defines how the knots are distributed (see ?make_knots for details) the defaults will create an equidistant knot sequence, deg sets the degree of the spline function and also influences how many outer knots will be used and periodic which determines whether the spline basis will be periodic. It’s possible to provide vectors of values for each of these parameters. In that case, all parameter combinations will be used to create the respective matrices and all candidates will be considered during online-learning. Parameters p_smooth_pr and p_smooth_mv determine the hat-matrix creation for P-Spline smoothing. In addition to the inputs mentioned before, they require to provide ndiff which determines the degree of differentiation applied to the basis-matrix (can take any value between and including 1 and 2), lambda which determines the degree of penalization applied to the smoothing, higher values will give smoother weight functions. As for the other parameters, it is possible to provide multiple values.

Value
Returns weights and corresponding predictions.
Examples

## Not run:
T <- 50 # Observations
N <- 2 # Experts
P <- 9 # Quantiles
prob_grid <- 1:P / (P + 1)

y <- rnorm(n = T) # Realized
experts <- array(dim = c(T, P, N)) # Predictions
for (t in 1:T) {
  experts[t, , 1] <- qnorm(prob_grid, mean = -1, sd = 1)
  experts[t, , 2] <- qnorm(prob_grid, mean = 3, sd = sqrt(4))
}

model <- online(
  y = matrix(y),
  experts = experts,
  tau = prob_grid,
  p_smooth_pr = list(lambda = 10)
)

print(model)
plot(model)

ew_y <- matrix(rnorm(1)) # Realized
new_experts <- experts[T, , drop = FALSE]
# Update will update the models weights etc if you provide new realizations
model <- update(model, new_y = new_y, new_experts = new_experts)

# Predict will expand `model$predictions` by default
model <- predict(model, new_experts = new_experts, update_model = TRUE)

## End(Not run)

oracle

Probabilistic Forecast Combination - Oracle

Description

Returns predictions and weights calculated by numeric optimization. The optimization is done in hindsight. This means all observations are used.

Usage

oracle(y, experts, tau, affine = FALSE,
positive = FALSE, intercept = FALSE, debias = TRUE,
loss_function = "quantile", loss_parameter = 1, forget = 0)
Arguments

- **y**: A numeric matrix of realizations. In probabilistic settings a matrix of dimension Tx1, in multivariate settings a TxD matrix. In the latter case, each slice of the expert’s array gets evaluated using the corresponding column of the y matrix.

- **experts**: An array of predictions with dimension (Observations, Quantiles, Experts).

- **tau**: A numeric vector of probabilities.

- **affine**: Defines whether weights are summing to 1 or not. Defaults to FALSE.

- **positive**: Defines if a positivity constraint is applied to the weights. Defaults to FALSE.

- **intercept**: Determines if an intercept is added, defaults to FALSE. If true, a new first expert is added, always predicting 1.

- **debias**: Defines whether the intercepts weight is constrained or not. If TRUE (the default), the intercept weight is unconstrained. Only affects the results if affine and or positive is set to TRUE. If FALSE, the intercept is treated as an expert.

- **loss_function**: Either "quantile", "expectile" or "percentage".

- **loss_parameter**: Optional parameter scaling the power of the loss function.

- **forget**: Adds an exponential forgetting to the optimization. Past observations will get less influence on the optimization. Defaults to 0, which corresponds to no forgetting.

Value

Returns weights and corresponding predictions. It is possible to calculate the best convex combination of weights by setting affine and positive to TRUE.

Examples

```r
## Not run:
T <- 50 # Observations
N <- 2 # Experts
P <- 9 # Quantiles
prob_grid <- 1:P / (P + 1)
y <- rnorm(n = T) # Realized
experts <- array(dim = c(T, P, N)) # Predictions
for (t in 1:T) {
  experts[t, , 1] <- qnorm(prob_grid, mean = -1, sd = 1)
  experts[t, , 2] <- qnorm(prob_grid, mean = 3, sd = sqrt(4))
}
model <- oracle(
  y = matrix(y),
  experts = experts
)
## End(Not run)
```
Description
This function calculates the B-Spline basis penalty. It follows the procedure outlined in the paper by Zheyuan Li, Jiguo Cao, 2022 "General P-Splines for Non-Uniform B-Splines" doi:10.48550/arXiv.2201.06808. For equidistant knots it coincides with the usual penalty based on the identity. For non-equidistant knots it is a weighted penalty with respect to the knot distances. In addition to the above, we added the possibility to calculate periodic penalties which are based on the periodic differencing matrices.

Usage
penalty(knots, order, periodic = FALSE, max_diff = 999L)

Arguments
knots Vector of knots.
order Order of the Basis (degree + 1).
periodic Whether the penalties should be periodic or not.
max_diff Maximum difference order to calculate.

Value
Returns a list of (order - 1) penalty matrices.

Examples
## Not run:
# Equidistant knots with order 2
knots <- 1:10
P <- penalty(knots, order = 2)
print(P[[1]]) # First differences

# Non equidistant knots
knots <- c(0, 0, 0, 1, 3, 4, 4, 4)
P <- penalty(knots, order = 4)
print(P[[1]]) # First differences
print(P[[2]]) # Second differences
print(P[[3]]) # Third differences

# Periodic penalty for equidistant knots
order <- 4
plot.online

deg <- order - 1
knots <- 1:15

penalty(knots, order = order, periodic = TRUE)[[1]]
penalty(knots, order = order, periodic = TRUE)[[2]]
penalty(knots, order = order, periodic = TRUE)[[3]]

## End(Not run)

---

plot.batch  
*Plot method for batch models*

**Description**

Plots the most recent weights in each quantile.

**Usage**

```r
# S3 method for class 'batch'
plot(x, ...)
```

**Arguments**

- `x`  
  Object of class inheriting from 'batch'
- `...`  
  further arguments are ignored

---

plot.online  
*Plot method for online models*

**Description**

Plots the most recent weights in each quantile.

**Usage**

```r
# S3 method for class 'online'
plot(x, ...)
```

**Arguments**

- `x`  
  Object of class inheriting from 'online'
- `...`  
  further arguments are ignored
post_process_model  

**Post Process Data from conline Class**

**Description**

This function works in conjunction with the conline class. After the main learning task, it takes the output of the conline class and returns an object suitable for visualization, further, and deployment analysis.

**Usage**

```r
post_process_model(model_instance, names)
```

**Arguments**

- `model_instance`  
  An instance of conline.
- `names`  
  A named list with dimnames of y and experts.

**predict.online**  

**Predict method for online models**

**Description**

Calculates predictions based on new expert advice. This does not update weights. If new observations are available use update instead. The latter updates and weights and computes predictions.

**Usage**

```r
## S3 method for class 'online'
predict(object, new_experts, update_model = TRUE, ...)
```

**Arguments**

- `object`  
  Object of class inheriting from 'online'
- `new_experts`  
  new expert predictions
- `update_model`  
  Defines whether the model object should be updated or not. If TRUE, new forecaster and expert predictions are appended onto the respective object items. Defaults to TRUE.
- `...`  
  further arguments are ignored

**Value**

predict.online produces an updated model object.
print.batch

Description

Prints the average loss of all and the forecast combination.

Usage

## S3 method for class 'batch'
print(x, ...)

Arguments

x Object of class inheriting from 'batch'
...

print.online

Description

Prints the average loss of all experts and the forecast combination.

Usage

## S3 method for class 'online'
print(x, ...)

Arguments

x Object of class inheriting from 'online'
...

further arguments are ignored
splines2_basis

Create B-Spline basis

Description

This function creates a B-Spline matrix.

Usage

splines2_basis(x, knots, deg, periodic = FALSE, intercept = TRUE)

Arguments

x
   Vector of values.

knots
   Vector of knots.

deg
   Degree of the Spline functions.

periodic
   Whether the basis should be periodic or not.

intercept
   Whether the first column should be kept.

Value

Returns a matrix of B-Spline basis functions.

Examples

n <- 9
deg <- 3
mu <- 0.35
x <- 0:1000 / 1000

knots <- make_knots(n, mu = mu, deg = deg)

B <- splines2_basis(x, knots, deg)
ts.plot(B, col = 1:dim(B)[2])

# Periodic Case
B <- splines2_basis(x, knots, deg, periodic = TRUE)
ts.plot(B, col = 1:dim(B)[2])
summary.online  Summary method for online models

Description
Calculates parameters chosen during optimization and aggregates losses.

Usage
## S3 method for class 'online'
summary(object, ...)

Arguments
object Object of class inheriting from 'online'
... further arguments are ignored

tidy.online.experts_loss  Tidy the Experts’ losses of an Online object

Description
tidy will transform the experts_loss array of an online object into a tibble that is better suited for plotting and analysis.

Usage
## S3 method for class 'online.experts_loss'
tidy(x, ...)

Arguments
x The experts_loss of an online object.
... Not currently used.

Value
A tibble with columns t d p k and w corresponding to the time, marginals, probabilities, and experts_loss of the online-learning computation.
tidy.online.forecaster_loss

*Tidy the Experts’ losses of an Online object*

**Description**

`tidy` will transform the `forecaster_loss` array of an online object into a tibble that is better suited for plotting and analysis.

**Usage**

```r
## S3 method for class 'online.forecaster_loss'
tidy(x, ...)
```

**Arguments**

- `x` The `forecaster_loss` of an online object.
- `...` Not currently used.

**Value**

A tibble with columns t, d, p, k and w corresponding to the time, marginals, probabilities, and forecaster_loss of the online-learning computation.

---

tidy.online.predictions

*Tidy the Predictions of an Online object*

**Description**

`tidy` will transform the `predictions` array of an online object into a tibble that is better suited for plotting and analysis.

**Usage**

```r
## S3 method for class 'online.predictions'
tidy(x, ...)
```

**Arguments**

- `x` The `predictions` of an online object.
- `...` Not currently used.

**Value**

A tibble with columns t, d, p, k and w corresponding to the time, marginals, probabilities, and predictions of the online-learning computation.
tidy.online.weights  Tidy the Weights of an Online object

Description

tidy will transform the weights array of an online object into a tibble that is better suited for plotting and analysis.

Usage

## S3 method for class 'online.weights'
tidy(x, ...)

Arguments

x  The weights of an online object.
...
Not currently used.

Value

A tibble with columns \( t \ d \ p \ k \) and \( w \) corresponding to the time, marginals, probabilities, experts, and weights of the online-learning computation.

update.online  Update method for online models

Description

Continues learning using new observations and new expert advice.

Usage

## S3 method for class 'online'
update(object, new_y, new_experts = NULL, trace = FALSE, ...)

Arguments

object  Object of class inheriting from 'online'
new_y  new observations
new_experts  new expert predictions. This must be left unspecified
trace  If a progress bar shall be shown. Defaults to FALSE if the model already contains the expert predictions corresponding to new_y.
...
not further arguments are ignored

Value

update.online produces an updated model object.
Index

* package
  profoc-package, 2

autoplot.batch, 4
autoplot.online, 4

batch, 5

conline, 8

init_experts_list, 8

make_basis_mats, 9
make_hat_mats, 10
make_knots, 11

online, 11

oracle, 15

penalty, 17

plot.batch, 18
plot.online, 18
post_process_model, 19
predict.online, 19
print.batch, 20
print.online, 20
profoc-package, 2

splines2_basis, 21
summary.online, 22

tidy.online.experts_loss, 22
tidy.online.forecaster_loss, 23
tidy.online.predictions, 23
tidy.online.weights, 24

update.online, 24