

Package ‘profoc’

March 2, 2023

Type Package

Title Probabilistic Forecast Combination Using CRPS Learning

Version 1.1.1

Date 2023-03-02

Description Combine probabilistic forecasts using CRPS learning algorithms proposed in Berrisch, Ziel (2021) <[arXiv:2102.00968](https://arxiv.org/abs/2102.00968)> <[doi:10.1016/j.jeconom.2021.11.008](https://doi.org/10.1016/j.jeconom.2021.11.008)>. The package implements multiple online learning algorithms like Bernstein online aggregation; see Wintemberger (2014) <[arXiv:1404.1356](https://arxiv.org/abs/1404.1356)>. Quantile regression is also implemented for comparison purposes. Model parameters can be tuned automatically with respect to the loss of the forecast combination. Methods like `predict()`, `update()`, `plot()` and `print()` are available for convenience. This package utilizes the `optim` C++ library for numeric optimization <<https://github.com/kthohr/optim>>.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 3.0.2)

Imports Rcpp (>= 1.0.5), Matrix, abind, methods, lifecycle

LinkingTo Rcpp, RcppArmadillo (>= 0.10.7.5.0), RcppProgress, splines2 (>= 0.4.4)

URL <https://profoc.berrisch.biz/>

BugReports <https://github.com/BerriJ/profoc/issues>

RoxygenNote 7.2.3

Language en-US

Suggests testthat (>= 3.0.0), gamlss.dist, ggplot2

Config/testthat/edition 3

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>

Repository CRAN

Date/Publication 2023-03-02 07:10:02 UTC

R topics documented:

profoc-package	2
autoplot	3
autoplot.batch	4
autoplot.online	4
batch	5
conline	8
online	9
oracle	13
penalty	14
plot.batch	15
plot.online	15
predict.online	16
print.batch	17
print.online	17
summary.online	18
update.online	18
Index	19

profoc-package	<i>Package Info</i>
----------------	---------------------

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](https://doi.org/10.1016/j.jeconom.2021.11.008)

Details

Index of help topics:

<code>autoplot</code>	Create a complete ggplot appropriate to a particular data type
<code>autoplot.batch</code>	Autoplot method for batch models
<code>autoplot.online</code>	Autoplot method for online models
<code>batch</code>	Probabilistic Forecast Combination - Batch
<code>conline</code>	Create an conline Object from the conline C++ Class
<code>online</code>	Probabilistic Forecast Combination - Online
<code>oracle</code>	Probabilistic Forecast Combination - Oracle
<code>penalty</code>	B-Spline penalty
<code>plot.batch</code>	Plot method for batch models
<code>plot.online</code>	Plot method for online models
<code>predict.online</code>	Predict method for online models

print.batch	Print method for batch models
print.online	Print method for online models
profoc-package	Package Info
summary.online	Summary method for online models
update.online	Update method for online models

Author(s)

Maintainer: Jonathan Berrisch <mailto:Jonathan@Berrisch.biz>
Co-Author: Florian Ziel

References

Berrisch, Ziel: "CRPS Learning", 2021
[doi:10.1016/j.jeconom.2021.11.008](https://doi.org/10.1016/j.jeconom.2021.11.008)
[doi:10.48550/arXiv.2102.00968](https://doi.org/10.48550/arXiv.2102.00968)

See Also

Source Code: <https://github.com/BerriJ/profoc>
BugReports: <https://github.com/BerriJ/profoc/issues>

autoplot

Create a complete ggplot appropriate to a particular data type

Description

autoplot() uses ggplot2 to draw a particular plot for an object of a particular class in a single command. This defines the S3 generic that other classes and packages can extend.

Usage

```
autoplot(object, ...)
```

Arguments

object	an object, whose class will determine the behavior of autoplot
...	other arguments passed to specific methods

Value

a ggplot object

See Also

[ggplot2::autolayer\(\)](#), [ggplot2::ggplot\(\)](#) and [ggplot2::fortify\(\)](#)

autoplot.batch *Autoplot method for batch models*

Description

Plots the most recent weights in each quantile using ggplot2.

Usage

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

Arguments

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

autoplot.online *Autoplot method for online models*

Description

Plots the most recent weights in each quantile using ggplot2.

Usage

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

Arguments

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

`batch`*Probabilistic Forecast Combination - Batch*

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,  
  basis_knot_distance = 1/(dim(experts)[2] + 1),  
  basis_knot_distance_power = 1,  
  basis_deg = 1,  
  forget = 0,  
  soft_threshold = -Inf,  
  hard_threshold = -Inf,  
  fixed_share = 0,  
  p_smooth_lambda = -Inf,  
  p_smooth_knot_distance = basis_knot_distance,  
  p_smooth_knot_distance_power = basis_knot_distance_power,  
  p_smooth_deg = basis_deg,  
  p_smooth_ndiff = 1.5,  
  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 $T \times 1$, in multivariate settings a $T \times P$ 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)
basis_knot_distance	determines the distance of the knots in the probability basis. Defaults to $1 / (\dim(\text{experts})[2] + 1)$.
basis_knot_distance_power	Parameter which defines the symmetry of the basis reducing the probability space. Defaults to 1, which corresponds to equidistant knots. Values less than 1 create more knots in the center, while values above 1 concentrate more knots in the tails.
basis_deg	Degree of the basis reducing the probability space. Defaults to 1.
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: $w = \text{sgn}(w) * \max(\text{abs}(w) - t, 0)$ where t is the soft_threshold parameter. Defaults to $-\infty$, 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: $w = w * (\text{abs}(w) > t)$ 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.
p_smooth_lambda	Penalization parameter used in the smoothing step. -Inf causes the smoothing step to be skipped (default).
p_smooth_knot_distance	determines the distance of the knots. Defaults to the value of basis_knot_distance. Corresponds to the grid steps when knot_distance_power = 1 (the default).
p_smooth_knot_distance_power	Parameter which defines the symmetry of the P-Spline basis. Takes the value of basis_knot_distance_power if unspecified.
p_smooth_deg	Degree of the B-Spline basis functions. Defaults to the value of basis_deg.
p_smooth_ndiff	Degree of the differencing operator in the smoothing equation. 1.5 (default) leads to shrinkage towards a constant. Can take values from 1 to 2. If a value in between is used, a weighted sum of the first and second differentiation matrix is calculated.
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.

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_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

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


Description

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

[Stable]

Usage

```

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),
  p_smooth_pr = list(knots = P, mu = 0.5, sigma = 1, nonc = 0, tailweight = 1, deg = 1,
    ndiff = 1.5, lambda = -Inf),
  b_smooth_mv = list(knots = D, mu = 0.5, sigma = 1, nonc = 0, tailweight = 1, deg = 1),
  p_smooth_mv = list(knots = D, mu = 0.5, sigma = 1, nonc = 0, tailweight = 1, deg = 1,
    ndiff = 1.5, lambda = -Inf),
  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
)

```

Arguments

y A numeric matrix of realizations. In probabilistic settings a matrix of dimension $T \times 1$, in multivariate settings a $T \times P$ 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 $T \times D \times P \times K$ (Observations x Variables x Quantiles x Experts) or $T \times D \times K$ or $T \times P \times K$.
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.
b_smooth_mv	A named list determining how the B-Spline matrices for multivariate smoothing are created. Default corresponds to no probabilistic smoothing. See details.
p_smooth_mv	A named list determining how the hat matrices for probabilistic P-Spline smoothing are created. Default corresponds to no smoothing. See details.
forget_regret	Share of past regret not to be considered, resp. to be forgotten in every iteration of the algorithm. Defaults to 0.
soft_threshold	If specified, the following soft threshold will be applied to the weights: $w = \text{sgn}(w) \cdot \max(\text{abs}(w) - t, 0)$ where t is the soft_threshold parameter. Defaults to $-\infty$, 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: $w = w \cdot (\text{abs}(w) > t)$ where t is the threshold_hard parameter. Defaults to $-\infty$, 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.
gamma	Scaling parameter for the learning rate.
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.

parametergrids	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: "general", "b_smooth_pr", "b_smooth_mv", "p_smooth_pr", "p_smooth_mv". The "general" matrix must contain 11 named columns: "forget_regret", "soft_threshold", "hard_threshold", "fixed_share", "basis_pr_idx", "basis_mv_idx", "hat_pr_idx", "hat_mv_idx", "gamma", "loss_share", "regret_share". The matrices determining the basis smoothing (b_smooth_pr, b_smooth_mv) must contain the following named columns: n, mu, sigma, nonc, tailw, deg. In addition to the columns of the basis smoothing matrices, the matrices determining the penalized smoothing (p_smooth_pr, p_smooth_mv) must contain the following columns: diff, lambda. The *_idx columns in the general matrix determine which row of the corresponding smoothing matrix is used.
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.
save_past_performance	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.
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 $D \times P \times K$ used as starting weights. "R0" a matrix of dimension $P \times K$ or $1 \times K$ 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.

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 include are: `knots` which determines the number of knots to be created, `mu`, `sigma`, `sigma`, `nonc`, `tailweight` correspond to parameters of the beta distribution, which defines how the knots are #distributed (see `?make_knots2` 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. 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)

new_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 $T \times 1$, in multivariate settings a $T \times P$ 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

```
## 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)
```

penalty

B-Spline penalty

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](https://doi.org/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.

Usage

```
penalty(knots, order, max_diff = 999L)
```

Arguments

knots	Vector of knots.
order	Order of the Basis (degree + 1).
max_diff	Maximum difference order to calculate.

Value

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

Examples

```
## Not run:
# Equidisan 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, 0, 1, 3, 4, 4, 4, 4)

P <- penalty(knots, order = 4)

print(P[[1]]) # First differences
print(P[[2]]) # Second differences
print(P[[3]]) # Third differences

## End(Not run)
```

plot.batch

Plot method for batch models

Description

Plots the most recent weights in each quantile.

Usage

```
## 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

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

Arguments

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

predict.online	<i>Predict method for online models</i>
----------------	---

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

```
## 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	<i>Print method for batch models</i>
-------------	--------------------------------------

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'
...	further arguments are ignored

print.online	<i>Print method for online models</i>
--------------	---------------------------------------

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

summary.online	<i>Summary method for online models</i>
----------------	---

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

update.online	<i>Update method for online models</i>
---------------	--

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.
...	further arguments are ignored

Value

update.online produces an updated model object.

Index

* package

profoc-package, 2

autoplot, 3

autoplot.batch, 4

autoplot.online, 4

batch, 5

conline, 8

ggplot2::autolayer(), 3

ggplot2::fortify(), 3

ggplot2::ggplot(), 3

online, 9

oracle, 13

penalty, 14

plot.batch, 15

plot.online, 15

predict.online, 16

print.batch, 17

print.online, 17

profoc-package, 2

summary.online, 18

update.online, 18