Package 'conformalForecast'

January 15, 2026

Title Conformal Prediction Methods for Multistep-Ahead Time Series Forecasting

Version 0.1.1

Description Methods and tools for performing multistep-ahead time series forecasting using conformal prediction methods including classical conformal prediction, adaptive conformal prediction, conformal PID (Proportional-Integral-Derivative) control, and autocorrelated multistep-ahead conformal prediction.
The methods were described by Wang and Hyndman (2024) <doi:10.48550/arXiv.2410.13115>.

License GPL-3

URL https://github.com/xqnwang/conformalForecast, https://xqnwang.github.io/conformalForecast/

BugReports https://github.com/xqnwang/conformalForecast/issues

Imports forecast, ggdist, rlang, stats, zoo

Suggests dplyr, ggplot2, knitr, rmarkdown, testthat (>= 3.0.0), tibble, tsibble

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder knitr

Depends R (>= 4.1.0)

NeedsCompilation no

Author Xiaoqian Wang [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-4827-496X>), Rob Hyndman [aut] (ORCID: <https://orcid.org/0000-0002-2140-5352>)

Maintainer Xiaoqian Wang <Xiaoqian.Wang@amss.ac.cn>

Repository CRAN

Date/Publication 2026-01-15 08:10:10 UTC

Contents

`conformalForecast-package`

conformalForecast: Conformal Prediction Methods for Multistep-Ahead Time Series Forecasting

---

Description

Methods and tools for performing multistep-ahead time series forecasting using conformal prediction methods including classical conformal prediction, adaptive conformal prediction, conformal PID (Proportional-Integral-Derivative) control, and autocorrelated multistep-ahead conformal prediction. The methods were described by Wang and Hyndman (2024) doi:10.48550/arXiv.2410.13115.

Author(s)

Maintainer: Xiaoqian Wang <Xiaoqian.Wang@amss.ac.cn> (ORCID) [copyright holder]

Authors:

- Rob Hyndman <Rob.Hyndman@monash.edu> (ORCID)

See Also

Useful links:

- https://github.com/xqnwang/conformalForecast
- https://xqnwang.github.io/conformalForecast/
- Report bugs at https://github.com/xqnwang/conformalForecast/issues

`accuracy.cvforecast` *Accuracy measures for a cross-validation model and a conformal prediction model*

Description

Return range of summary measures of the out-of-sample forecast accuracy. If `x` is given, the function also measures test set forecast accuracy. If `x` is not given, the function only produces accuracy measures on validation set.

Usage

```
## S3 method for class 'cvforecast'
accuracy(
  object,
  x,
  CV = TRUE,
  period = NULL,
  measures = interval_measures,
  byhorizon = FALSE,
  ...
)

## S3 method for class 'cpforecast'
accuracy(object, ...)
```

Arguments

`object`	An object of class `"cvforecast"` or `"cpforecast"`.
`x`	An optional numerical vector containing actual values of the same length as `mean` in `object`.
`CV`	If `TRUE`, the cross-validation forecast accuracy will be returned.
`period`	The seasonal period of the data.
`measures`	A list of accuracy measure functions to compute (such as point_measures or interval_measures).
`byhorizon`	If `TRUE`, accuracy measures will be calculated for each individual forecast horizon `h` separately.
`...`	Additional arguments depending on the specific measure.

Details

The measures calculated are:

- ME: Mean Error
- MAE: Mean Absolute Error

- MSE: Mean Squared Error
- RMSE: Root Mean Squared Error
- MPE: Mean Percentage Error
- MAPE: Mean Absolute Percentage Error
- MASE: Mean Absolute Scaled Error
- RMSSE: Root Mean Squared Scaled Error
- winkler_score: Winkler Score
- MSIS: Mean Scaled Interval Score

Value

A matrix giving mean out-of-sample forecast accuracy measures.

See Also

point_measures, interval_measures

Examples

```
# Simulate time series from an AR(2) model
library(forecast)
series <- arima.sim(n = 200, list(ar = c(0.8, -0.5)), sd = sqrt(1))

# Cross-validation forecasting with a rolling window
far2 <- function(x, h, level) {
  Arima(x, order = c(2, 0, 0)) |>
    forecast(h = h, level)
}
fc <- cvforecast(series, forecastfun = far2, h = 3, level = 95,
                 forward = TRUE, initial = 1, window = 50)

# Out-of-sample forecast accuracy on validation set
accuracy(fc, measures = point_measures, byhorizon = TRUE)
accuracy(fc, measures = interval_measures, level = 95, byhorizon = TRUE)

# Out-of-sample forecast accuracy on test set
accuracy(fc, x = c(1, 0.5, 0), measures = interval_measures,
         level = 95, byhorizon = TRUE)
```

---

acmcp *Autocorrelated multistep-ahead conformal prediction method*

Description

Compute prediction intervals and other information by applying the Autocorrelated Multistep-ahead Conformal Prediction (AcMCP) method. The method can only deal with asymmetric nonconformity scores, i.e., forecast errors.

Usage

```
acmcp(
  object,
  alpha = 1 - 0.01 * object$level,
  ncal = 10,
  rolling = FALSE,
  integrate = TRUE,
  scorecast = TRUE,
  lr = 0.1,
  Tg = NULL,
  delta = NULL,
  Csat = 2/pi * (ceiling(log(Tg) * delta) - 1/log(Tg)),
  KI = max(abs(object$errors), na.rm = TRUE),
  ...
)
```

Arguments

object	An object of class "cvforecast". It must have an argument x for original univariate time series, an argument MEAN for point forecasts and ERROR for forecast errors on validation set. See the results of a call to cvforecast.
alpha	A numeric vector of significance levels to achieve a desired coverage level $1-\alpha$.
ncal	Length of the burn-in period for training the scorecaster. If rolling = TRUE, it is also used as the length of the trailing windows for learning rate calculation and the windows for the calibration set. If rolling = FALSE, it is used as the initial period of calibration sets and trailing windows for learning rate calculation.
rolling	If TRUE, a rolling window strategy will be adopted to form the trailing window for learning rate calculation and the calibration set for scorecaster if applicable. Otherwise, expanding window strategy will be used.
integrate	If TRUE, error integration will be included in the update process.
scorecast	If TRUE, scorecasting will be included in the update process.
lr	Initial learning rate used for quantile tracking.
Tg	The time that is set to achieve the target absolute coverage guarantee before this.
delta	The target absolute coverage guarantee is set to $1-\alpha-\delta$.
Csat	A positive constant ensuring that by time Tg, an absolute guarantee is of at least $1-\alpha-\delta$ coverage.
KI	A positive constant to place the integrator on the same scale as the scores.
...	Other arguments are passed to the function.

Details

Similar to the PID method, the AcMCP method also integrates three modules (P, I, and D) to form the final iteration. However, instead of performing conformal prediction for each individual forecast horizon h separately, AcMCP employs a combination of an MA$(h-1)$ model and a linear regression model of $e_{t+h|t}$ on $e_{t+h-1|t}, \ldots, e_{t+1|t}$ as the scorecaster. This allows the AcMCP method to capture the relationship between the h-step ahead forecast error and past errors.

Value

A list of class `c("acmcp", "cpforecast", "forecast")` with the following components:

`x`	The original time series.
`series`	The name of the series `x`.
`method`	A character string "acmcp".
`cp_times`	The number of times the conformal prediction is performed in cross-validation.
`MEAN`	Point forecasts as a multivariate time series, where the hth column holds the point forecasts for forecast horizon h. The time index corresponds to the period for which the forecast is produced.
`ERROR`	Forecast errors given by $e_{t+h\|t} = y_{t+h} - \hat{y}_{t+h\|t}$.
`LOWER`	A list containing lower bounds for prediction intervals for each `level`. Each element within the list will be a multivariate time series with the same dimensional characteristics as `MEAN`.
`UPPER`	A list containing upper bounds for prediction intervals for each `level`. Each element within the list will be a multivariate time series with the same dimensional characteristics as `MEAN`.
`level`	The confidence values associated with the prediction intervals.
`call`	The matched call.
`model`	A list containing information abouth the conformal prediction model.

If `mean` is included in the `object`, the components `mean`, `lower`, and `upper` will also be returned, showing the information about the test set forecasts generated using all available observations.

References

Wang, X., and Hyndman, R. J. (2024). "Online conformal inference for multi-step time series forecasting", arXiv preprint arXiv:2410.13115.

See Also

`pid`

Examples

```
# Simulate time series from an AR(2) model
library(forecast)
series <- arima.sim(n = 200, list(ar = c(0.8, -0.5)), sd = sqrt(1))

# Cross-validation forecasting
far2 <- function(x, h, level) {
  Arima(x, order = c(2, 0, 0)) |>
    forecast(h = h, level)
}
fc <- cvforecast(series, forecastfun = far2, h = 3, level = 95,
                 forward = TRUE, initial = 1, window = 50)
```

```
# AcMCP setup
Tg <- 200; delta <- 0.01
Csat <- 2 / pi * (ceiling(log(Tg) * delta) - 1 / log(Tg))
KI <- 2
lr <- 0.1

# AcMCP with integrator and scorecaster
acmcpfc <- acmcp(fc, ncal = 50, rolling = TRUE,
             integrate = TRUE, scorecast = TRUE,
             lr = lr, KI = KI, Csat = Csat)
print(acmcpfc)
summary(acmcpfc)
```

---

acp *Adaptive conformal prediction method*

Description

Compute prediction intervals and other information by applying the adaptive conformal prediction (ACP) method.

Usage

```
acp(
  object,
  alpha = 1 - 0.01 * object$level,
  gamma = 0.005,
  symmetric = FALSE,
  ncal = 10,
  rolling = FALSE,
  quantiletype = 1,
  update = FALSE,
  na.rm = TRUE,
  ...
)
```

Arguments

object	An object of class "cvforecast". It must have an argument x for original univariate time series, an argument MEAN for point forecasts and ERROR for forecast errors on validation set. See the results of a call to cvforecast.
alpha	A numeric vector of significance levels to achieve a desired coverage level $1-\alpha$.
gamma	The step size parameter $\gamma > 0$ for α updating.
symmetric	If TRUE, symmetric nonconformity scores (i.e. $\|e_{t+h\|t}\|$) are used. If FALSE, asymmetric nonconformity scores (i.e. $e_{t+h\|t}$) are used, and then upper bounds and lower bounds are produced separately.

ncal	Length of the calibration set. If rolling = FALSE, it denotes the initial period of calibration sets. Otherwise, it indicates the period of every rolling calibration set.
rolling	If TRUE, a rolling window strategy will be adopted to form the calibration set. Otherwise, expanding window strategy will be used.
quantiletype	An integer between 1 and 9 determining the type of quantile estimator to be used. Types 1 to 3 are for discontinuous quantiles, types 4 to 9 are for continuous quantiles. See the weighted_quantile function in the ggdist package.
update	If TRUE, the function will be compatible with the update(update.cpforecast) function, allowing for easy updates of conformal prediction.
na.rm	If TRUE, corresponding entries in sample values are removed if it is NA when calculating sample quantile.
...	Other arguments are passed to the weighted_quantile function for quantile computation.

Details

The ACP method considers the online update:

$$\alpha_{t+h|t} := \alpha_{t+h-1|t-1} + \gamma(\alpha - \mathrm{err}_{t|t-h}),$$

for each individual forecast horizon h, respectively, where $\mathrm{err}_{t|t-h} = 1$ if $s_{t|t-h} > q_{t|t-h}$, and $\mathrm{err}_{t|t-h} = 0$ if $s_{t|t-h} \leq q_{t|t-h}$.

Value

A list of class c("acp", "cpforecast", "forecast") with the following components:

x	The original time series.
series	The name of the series x.
method	A character string "acp".
cp_times	The number of times the conformal prediction is performed in cross-validation.
MEAN	Point forecasts as a multivariate time series, where the hth column holds the point forecasts for forecast horizon h. The time index corresponds to the period for which the forecast is produced.
ERROR	Forecast errors given by $e_{t+h|t} = y_{t+h} - \hat{y}_{t+h|t}$.
LOWER	A list containing lower bounds for prediction intervals for each level. Each element within the list will be a multivariate time series with the same dimensional characteristics as MEAN.
UPPER	A list containing upper bounds for prediction intervals for each level. Each element within the list will be a multivariate time series with the same dimensional characteristics as MEAN.
level	The confidence values associated with the prediction intervals.
call	The matched call.
model	A list containing information abouth the conformal prediction model.

If mean is included in the object, the components mean, lower, and upper will also be returned, showing the information about the forecasts generated using all available observations.

References

Gibbs, I., and Candes, E. (2021). "Adaptive conformal inference under distribution shift", *Advances in Neural Information Processing Systems*, **34**, 1660–1672.

Examples

```
# Simulate time series from an AR(2) model
library(forecast)
series <- arima.sim(n = 200, list(ar = c(0.8, -0.5)), sd = sqrt(1))

# Cross-validation forecasting
far2 <- function(x, h, level) {
  Arima(x, order = c(2, 0, 0)) |>
    forecast(h = h, level)
}
fc <- cvforecast(series, forecastfun = far2, h = 3, level = 95,
                 forward = TRUE, initial = 1, window = 50)

# ACP with asymmetric nonconformity scores and rolling calibration sets
acpfc <- acp(fc, symmetric = FALSE, gamma = 0.005, ncal = 50, rolling = TRUE)
print(acpfc)
summary(acpfc)
```

---

conformal *Conformal prediction*

Description

This function allows you to specify the method used to perform conformal prediction.

Usage

```
conformal(object, ...)

## S3 method for class 'cvforecast'
conformal(object, method = c("scp", "acp", "pid", "acmcp"), ...)
```

Arguments

`object`	An object of class `"cvforecast"`. It must have an argument `x` for original univariate time series, an argument `MEAN` for point forecasts and `ERROR` for forecast errors on validation set. See the results of a call to `cvforecast`.
`...`	Additional arguments to be passed to the selected conformal method.
`method`	A character string specifying the conformal method to be applied. Possible options include `"scp"` (scp), `"acp"`(acp), `"pid"`(pid), and `"acmcp"`(acmcp).

Value

An object whose class depends on the method invoked.

See Also

scp, acp, pid, and acmcp

Examples

```
# Simulate time series from an AR(2) model
library(forecast)
series <- arima.sim(n = 200, list(ar = c(0.8, -0.5)), sd = sqrt(1))

# Cross-validation forecasting
far2 <- function(x, h, level) {
  Arima(x, order = c(2, 0, 0)) |>
    forecast(h = h, level)
}
fc <- cvforecast(series, forecastfun = far2, h = 3, level = 95,
                 forward = TRUE, initial = 1, window = 50)

# Classical conformal prediction with equal weights
scpfc <- conformal(fc, method = "scp", symmetric = FALSE, ncal = 50, rolling = TRUE)
summary(scpfc)

# ACP with asymmetric nonconformity scores and rolling calibration sets
acpfc <- conformal(fc, method = "acp", symmetric = FALSE, gamma = 0.005,
                   ncal = 50, rolling = TRUE)
summary(acpfc)
```

---

coverage *Calculate interval forecast coverage*

Description

Calculate the mean coverage and the ifinn matrix for prediction intervals on validation set. If `window` is not `NULL`, a matrix of the rolling means of interval forecast coverage is also returned.

Usage

```
coverage(object, ..., level = 95, window = NULL, na.rm = FALSE)
```

Arguments

`object`	An object of class `"cvforecast"` or `"cpforecast"`.
`...`	Additional inputs if `object` is missing.
`level`	Target confidence level for prediction intervals.

window	If not NULL, the rolling mean matrix for coverage is also returned.
na.rm	A logical indicating whether NA values should be stripped before the rolling mean computation proceeds.

Value

A list of class "coverage" with the following components:

mean	Mean coverage across the validation set.
ifinn	A indicator matrix as a multivariate time series, where the hth column holds the coverage for forecast horizon h. The time index corresponds to the period for which the forecast is produced.
rollmean	If window is not NULL, a matrix of the rolling means of interval forecast coverage will be returned.

Examples

```
# Simulate time series from an AR(2) model
library(forecast)
series <- arima.sim(n = 200, list(ar = c(0.8, -0.5)), sd = sqrt(1))

# Cross-validation forecasting with a rolling window
far2 <- function(x, h, level) {
  Arima(x, order = c(2, 0, 0)) |>
    forecast(h = h, level)
}
fc <- cvforecast(series, forecastfun = far2, h = 3, level = 95,
                 forward = TRUE, initial = 1, window = 50)

# Mean and rolling mean coverage for interval forecasts on validation set
cov_fc <- coverage(fc, level = 95, window = 50)
str(cov_fc)
```

---

cvforecast *Time series cross-validation forecasting*

Description

Compute forecasts and other information by applying forecastfun to subsets of the time series y using a rolling forecast origin.

Usage

```
cvforecast(
  y,
  forecastfun,
  h = 1,
```

```
  level = c(80, 95),
  forward = TRUE,
  xreg = NULL,
  initial = 1,
  window = NULL,
  ...
)
```

Arguments

y	Univariate time series.
forecastfun	Function to return an object of class "forecast". Its first argument must be a univariate time series, and it must have an argument h for the forecast horizon and an argument level for the confidence level for prediction intervals. If exogenous predictors are used, then it must also have xreg and newxreg arguments corresponding to the training and test periods, respectively.
h	Forecast horizon.
level	Confidence level for prediction intervals. If NULL, prediction intervals will not be generated.
forward	If TRUE, the final forecast origin for forecasting is y_T. Otherwise, the final forecast origin is y_{T-1}.
xreg	Exogenous predictor variables passed to forecastfun if required. It should be of the same size as y+forward*h, otherwise, NA padding or subsetting will be applied.
initial	Initial period of the time series where no cross-validation forecasting is performed.
window	Length of the rolling window. If NULL, a rolling window will not be used.
...	Other arguments are passed to forecastfun.

Details

Let y denote the time series $y_1, \dots, y_T$ and let t_0 denote the initial period.

Suppose forward = TRUE. If window is NULL, forecastfun is applied successively to the subset time series $y_1, \dots, y_t$, for $t = t_0, \dots, T$, generating forecasts $\hat{y}_{t+1|t}, \dots, \hat{y}_{t+h|t}$. If window is not NULL and has a length of t_w, then forecastfun is applied successively to the subset time series $y_{t-t_w+1}, \dots, y_t$, for $t = \max(t_0, t_w), \dots, T$.

If forward is FALSE, the last observation used for training will be y_{T-1}.

Value

A list of class c("cvforecast", "forecast") with components:

x	The original time series.
series	The name of the series x.
xreg	Exogenous predictor variables used in the model, if applicable.
method	A character string "cvforecast".

`fit_times`	The number of times the model is fitted in cross-validation.
`MEAN`	Point forecasts as a multivariate time series, where the hth column holds the point forecasts for forecast horizon h. The time index corresponds to the period for which the forecast is produced.
`ERROR`	Forecast errors given by $e_{t+h\|t} = y_{t+h} - \hat{y}_{t+h\|t}$.
`LOWER`	A list containing lower bounds for prediction intervals for each `level`. Each element within the list will be a multivariate time series with the same dimensional characteristics as `MEAN`.
`UPPER`	A list containing upper bounds for prediction intervals for each `level`. Each element within the list will be a multivariate time series with the same dimensional characteristics as `MEAN`.
`level`	The confidence values associated with the prediction intervals.
`call`	The matched call.
`forward`	Whether `forward` is applied.

If `forward` is `TRUE`, the components `mean`, `lower`, `upper`, and `model` will also be returned, showing the information about the final fitted model and forecasts using all available observations, see e.g. `forecast.ets` for more details.

Examples

```
# Simulate time series from an AR(2) model
library(forecast)
series <- arima.sim(n = 200, list(ar = c(0.8, -0.5)), sd = sqrt(1))

# Example with a rolling window
far2 <- function(x, h, level) {
  Arima(x, order = c(2, 0, 0)) |>
    forecast(h = h, level)
}
fc <- cvforecast(series, forecastfun = far2, h = 3, level = 95,
                 forward = TRUE, initial = 1, window = 50)
print(fc)
summary(fc)

# Example with exogenous predictors
far2_xreg <- function(x, h, level, xreg, newxreg) {
  Arima(x, order=c(2, 0, 0), xreg = xreg) |>
    forecast(h = h, level = level, xreg = newxreg)
}
fc_xreg <- cvforecast(series, forecastfun = far2_xreg, h = 3, level = 95,
                      forward = TRUE, xreg = matrix(rnorm(406), ncol = 2, nrow = 203),
                      initial = 1, window = 50)
```

lagmatrix *Create lags or leads of a matrix*

Description

Find a shifted version of a matrix, adjusting the time base backward (lagged) or forward (leading) by a specified number of observations for each column.

Usage

```
lagmatrix(x, lag)
```

Arguments

x	A matrix or multivariate time series.
lag	A vector of lags (positive values) or leads (negative values) with a length equal to the number of columns of x.

Value

A matrix with the same class and size as x.

Examples

```
x <- matrix(rnorm(20), nrow = 5, ncol = 4)

# Create lags of a matrix
lagmatrix(x, c(0, 1, 2, 3))

# Create leads of a matrix
lagmatrix(x, c(0, -1, -2, -3))
```

ME *Point estimate accuracy measures*

Description

Accuracy measures for point forecast residuals.

Usage

```
ME(resid, na.rm = TRUE)

MAE(resid, na.rm = TRUE, ...)

MSE(resid, na.rm = TRUE, ...)

RMSE(resid, na.rm = TRUE, ...)

MPE(resid, actual, na.rm = TRUE, ...)

MAPE(resid, actual, na.rm = TRUE, ...)

MASE(
  resid,
  train,
  demean = FALSE,
  na.rm = TRUE,
  period,
  d = period == 1,
  D = period > 1,
  ...
)

RMSSE(
  resid,
  train,
  demean = FALSE,
  na.rm = TRUE,
  period,
  d = period == 1,
  D = period > 1,
  ...
)

point_measures
```

Arguments

resid	A numeric vector of residuals from either the validation or test data.
na.rm	If TRUE, remove missing values before calculating the measure.
...	Additional arguments for each measure.
actual	A numeric vector of responses matching the forecasts (for percentage measures).
train	A numeric vector of responses used to train the model (for scaled measures).
demean	Should the response be demeaned (for MASE and RMSSE)?
period	The seasonal period of the data.

d	Should the response model include a first difference?
D	Should the response model include a seasonal difference?

Format

An object of class list of length 8.

Value

For the individual functions (ME, MAE, MSE, RMSE, MPE, MAPE, MASE, RMSSE), returns a single numeric scalar giving the requested accuracy measure.

For the exported object point_measures, returns a **named list of functions** that can be supplied to higher-level accuracy routines.

Examples

```
# Toy residuals and data
set.seed(1)
y_train <- rnorm(50)
y_test  <- rnorm(10)
fcast   <- y_test + rnorm(10, sd = 0.2)
resid   <- y_test - fcast

# Basic measures
ME(resid)
MAE(resid)
RMSE(resid)

# Percentage measures require 'actual'
MPE(resid, actual = y_test)
MAPE(resid, actual = y_test)

# Scaled measures require training data (and seasonal period if applicable)
MASE(resid, train = y_train, period = 1)
RMSSE(resid, train = y_train, period = 1)
```

MSIS *Interval estimate accuracy measures*

Description

Accuracy measures for interval forecasts.

Usage

```
MSIS(
  lower,
  upper,
  actual,
  train,
  level = 95,
  period,
  d = period == 1,
  D = period > 1,
  na.rm = TRUE,
  ...
)

winkler_score(lower, upper, actual, level = 95, na.rm = TRUE, ...)

interval_measures
```

Arguments

`lower`	A numeric vector of lower bounds of interval forecasts.
`upper`	A numeric vector of upper bounds of interval forecasts.
`actual`	A numeric vector of realised values.
`train`	A numeric vector of responses used to train the model (for scaled scores).
`level`	The nominal level of the forecast interval (e.g., 95 or 0.95).
`period`	The seasonal period of the data.
`d`	Should the response model include a first difference?
`D`	Should the response model include a seasonal difference?
`na.rm`	If `TRUE`, remove missing values before calculating the measure.
`...`	Additional arguments for each measure.

Format

An object of class `list` of length 2.

Value

For `winkler_score` and `MSIS`, returns a single numeric scalar giving the average interval score (Winkler or mean scaled interval score).

For the exported object `interval_measures`, returns a **named list of functions** that can be supplied to higher-level accuracy routines.

Examples

```
set.seed(1)
actual <- rnorm(10)
lower  <- actual - runif(10, 0.5, 1)
upper  <- actual + runif(10, 0.5, 1)
train  <- rnorm(50)

# Winkler score at 95%
winkler_score(lower, upper, actual, level = 95)

# Mean scaled interval score (needs training data and period)
MSIS(lower, upper, actual, train, level = 95, period = 1)
```

---

pid *Conformal PID control method*

Description

Compute prediction intervals and other information by applying the conformal PID (Proportional-Integral-Derivative) control method.

Usage

```
pid(
  object,
  alpha = 1 - 0.01 * object$level,
  symmetric = FALSE,
  ncal = 10,
  rolling = FALSE,
  integrate = TRUE,
  scorecast = !symmetric,
  scorecastfun = NULL,
  lr = 0.1,
  Tg = NULL,
  delta = NULL,
  Csat = 2/pi * (ceiling(log(Tg) * delta) - 1/log(Tg)),
  KI = max(abs(object$errors), na.rm = TRUE),
  ...
)
```

Arguments

object	An object of class "cvforecast". It must have an argument x for original univariate time series, an argument MEAN for point forecasts and ERROR for forecast errors on validation set. See the results of a call to cvforecast.
alpha	A numeric vector of significance levels to achieve a desired coverage level $1-\alpha$.

symmetric	If TRUE, symmetric nonconformity scores (i.e. $\|e_{t+h\|t}\|$) are used. If FALSE, asymmetric nonconformity scores (i.e. $e_{t+h\|t}$) are used, and then upper bounds and lower bounds are produced separately.
ncal	Length of the burn-in period for training the scorecaster. If rolling = TRUE, it is also used as the length of the trailing windows for learning rate calculation and the windows for the calibration set. If rolling = FALSE, it is used as initial period of calibration sets and trailing windows for learning rate calculation.
rolling	If TRUE, a rolling window strategy will be adopted to form the trailing window for learning rate calculation and the calibration set for scorecaster if applicable. Otherwise, expanding window strategy will be used.
integrate	If TRUE, error integration will be included in the update process.
scorecast	If TRUE, scorecasting will be included in the update process, and scorecastfun should be given.
scorecastfun	A scorecaster function to return an object of class forecast. Its first argument must be a univariate time series, and it must have an argument h for the forecast horizon.
lr	Initial learning rate used for quantile tracking.
Tg	The time that is set to achieve the target absolute coverage guarantee before this.
delta	The target absolute coverage guarantee is set to $1-\alpha-\delta$.
Csat	A positive constant ensuring that by time Tg, an absolute guarantee is of at least $1-\alpha-\delta$ coverage.
KI	A positive constant to place the integrator on the same scale as the scores.
...	Other arguments are passed to the scorecastfun function.

Details

The PID method combines three modules to make the final iteration:

$$q_{t+h|t} = \underbrace{q_{t+h-1|t-1} + \eta(\mathrm{err}_{t|t-h} - \alpha)}_{\mathrm{P}} + \underbrace{r_t\left(\sum_{i=1}^{t}\left(\mathrm{err}_{i|i-h} - \alpha\right)\right)}_{\mathrm{I}} + \underbrace{\hat{s}_{t+h|t}}_{\mathrm{D}}$$

for each individual forecast horizon h, respectively, where

- Quantile tracking part (P) is $q_{t+h-1|t-1} + \eta(\mathrm{err}_{t|t-h} - \alpha)$, where $q_{1+h|1}$ is set to 0 without a loss of generality, $\mathrm{err}_{t|t-h} = 1$ if $s_{t|t-h} > q_{t|t-h}$, and $\mathrm{err}_{t|t-h} = 0$ if $s_{t|t-h} \leq q_{t|t-h}$.
- Error integration part (I) is $r_t\left(\sum_{i=1}^{t}\left(\mathrm{err}_{i|i-h} - \alpha\right)\right)$. Here we use a nonlinear saturation function $r_t(x) = K_{\mathrm{I}} \tan\left(x \log(t)/\left(tC_{\mathrm{sat}}\right)\right)$, where we set $\tan(x) = \mathrm{sign}(x) \cdot \infty$ for $x \notin [-\pi/2, \pi/2]$, and C_{sat} , $K_{\mathrm{I}} > 0$ are constants that we choose heuristically.
- Scorecasting part (D) is $\hat{s}_{t+h|t}$ is forecast generated by training a scorecaster based on nonconformity scores available at time t.

Value

A list of class `c("pid", "cpforecast", "forecast")` with the following components:

`x`	The original time series.
`series`	The name of the series `x`.
`method`	A character string "pid".
`cp_times`	The number of times the conformal prediction is performed in cross-validation.
`MEAN`	Point forecasts as a multivariate time series, where the hth column holds the point forecasts for forecast horizon h. The time index corresponds to the period for which the forecast is produced.
`ERROR`	Forecast errors given by $e_{t+h\|t} = y_{t+h} - \hat{y}_{t+h\|t}$.
`LOWER`	A list containing lower bounds for prediction intervals for each `level`. Each element within the list will be a multivariate time series with the same dimensional characteristics as `MEAN`.
`UPPER`	A list containing upper bounds for prediction intervals for each `level`. Each element within the list will be a multivariate time series with the same dimensional characteristics as `MEAN`.
`level`	The confidence values associated with the prediction intervals.
`call`	The matched call.
`model`	A list containing information abouth the conformal prediction model.

If `mean` is included in the `object`, the components `mean`, `lower`, and `upper` will also be returned, showing the information about the forecasts generated using all available observations.

References

Angelopoulos, A., Candes, E., and Tibshirani, R. J. (2024). "Conformal PID control for time series prediction", *Advances in Neural Information Processing Systems*, **36**, 23047–23074.

Examples

```
# Simulate time series from an AR(2) model
library(forecast)
series <- arima.sim(n = 200, list(ar = c(0.8, -0.5)), sd = sqrt(1))
# Cross-validation forecasting
far2 <- function(x, h, level) {
  Arima(x, order = c(2, 0, 0)) |>
    forecast(h = h, level)
}
fc <- cvforecast(series, forecastfun = far2, h = 3, level = 95,
                 forward = TRUE, initial = 1, window = 50)
# PID setup
Tg <- 200; delta <- 0.01
Csat <- 2 / pi * (ceiling(log(Tg) * delta) - 1 / log(Tg))
KI <- 2
lr <- 0.1
# PID without scorecaster
```

```
pidfc_nsf <- pid(fc, symmetric = FALSE, ncal = 50, rolling = TRUE,
                 integrate = TRUE, scorecast = FALSE,
                 lr = lr, KI = KI, Csat = Csat)
print(pidfc_nsf)
summary(pidfc_nsf)
# PID with a Naive model for the scorecaster
naivefun <- function(x, h) {
  naive(x) |> forecast(h = h)
}
pidfc <- pid(fc, symmetric = FALSE, ncal = 50, rolling = TRUE,
             integrate = TRUE, scorecast = TRUE, scorecastfun = naivefun,
             lr = lr, KI = KI, Csat = Csat)
```

---

scp *Classical split conformal prediction method*

Description

Compute prediction intervals and other information by applying the classical split conformal prediction (SCP) method.

Usage

```
scp(
  object,
  alpha = 1 - 0.01 * object$level,
  symmetric = FALSE,
  ncal = 10,
  rolling = FALSE,
  quantiletype = 1,
  weightfun = NULL,
  kess = FALSE,
  update = FALSE,
  na.rm = TRUE,
  ...
)
```

Arguments

object	An object of class "cvforecast". It must have an argument x for original univariate time series, an argument MEAN for point forecasts and ERROR for forecast errors on validation set. See the results of a call to cvforecast.
alpha	A numeric vector of significance levels to achieve a desired coverage level $1-\alpha$.
symmetric	If TRUE, symmetric nonconformity scores (i.e. $\|e_{t+h\|t}\|$) are used. If FALSE, asymmetric nonconformity scores (i.e. $e_{t+h\|t}$) are used, and then upper bounds and lower bounds are produced separately.

ncal	Length of the calibration set. If rolling = FALSE, it denotes the initial period of calibration sets. Otherwise, it indicates the period of every rolling calibration set.
rolling	If TRUE, a rolling window strategy will be adopted to form the calibration set. Otherwise, expanding window strategy will be used.
quantiletype	An integer between 1 and 9 determining the type of quantile estimator to be used. Types 1 to 3 are for discontinuous quantiles, types 4 to 9 are for continuous quantiles. See the weighted_quantile function in the ggdist package.
weightfun	Function to return a vector of weights used for sample quantile computation. Its first argument must be an integer indicating the number of observations for which weights are generated. If NULL, equal weights will be used for sample quantile computation. Currently, only non-data-dependent weights are supported.
kess	If TRUE, Kish's effective sample size is used for sample quantile computation.
update	If TRUE, the function will be compatible with the update(update.cpforecast) function, allowing for easy updates of conformal prediction.
na.rm	If TRUE, corresponding entries in sample values and weights are removed if either is NA when calculating sample quantile.
...	Other arguments are passed to weightfun.

Details

Consider a vector $s_{t+h|t}$ that contains the nonconformity scores for the h-step-ahead forecasts.

If symmetric is TRUE, $s_{t+h|t} = |e_{t+h|t}|$. When rolling is FALSE, the $(1-\alpha)$-quantile $\hat{q}_{t+h|t}$ are computed successively on expanding calibration sets $s_{1+h|1}, \ldots, s_{t|t-h}$, for $t = \text{ncal} + h, \ldots, T$. Then the prediction intervals will be $[\hat{y}_{t+h|t} - \hat{q}_{t+h|t}, \hat{y}_{t+h|t} + \hat{q}_{t+h|t}]$. When rolling is TRUE, the calibration sets will be of same length ncal.

If symmetric is FALSE, $s^u_{t+h|t} = e_{t+h|t}$ for upper interval bounds and $s^l_{t+h|t} = -e_{t+h|t}$ for lower bounds. Instead of computing $(1-\alpha)$-quantile, $(1-\alpha/2)$-quantiles for lower bound ($\hat{q}^l_{t+h|t}$) and upper bound ($\hat{q}^u_{t+h|t}$) are calculated based on their nonconformity scores, respectively. Then the prediction intervals will be $[\hat{y}_{t+h|t} - \hat{q}^l_{t+h|t}, \hat{y}_{t+h|t} + \hat{q}^u_{t+h|t}]$.

Value

A list of class c("scp", "cvforecast", "forecast") with the following components:

x	The original time series.
series	The name of the series x.
xreg	Exogenous predictor variables used, if applicable.
method	A character string "scp".
cp_times	The number of times the conformal prediction is performed in cross-validation.
MEAN	Point forecasts as a multivariate time series, where the hth column holds the point forecasts for forecast horizon h. The time index corresponds to the period for which the forecast is produced.

ERROR	Forecast errors given by $e_{t+h\|t} = y_{t+h} - \hat{y}_{t+h\|t}$.
LOWER	A list containing lower bounds for prediction intervals for each `level`. Each element within the list will be a multivariate time series with the same dimensional characteristics as MEAN.
UPPER	A list containing upper bounds for prediction intervals for each `level`. Each element within the list will be a multivariate time series with the same dimensional characteristics as MEAN.
`level`	The confidence values associated with the prediction intervals.
`call`	The matched call.
`model`	A list containing detailed information about the `cvforecast` and `conformal` models.

If `mean` is included in the `object`, the components `mean`, `lower`, and `upper` will also be returned, showing the information about the test set forecasts generated using all available observations.

See Also

`weighted_quantile`

Examples

```
# Simulate time series from an AR(2) model
library(forecast)
series <- arima.sim(n = 200, list(ar = c(0.8, -0.5)), sd = sqrt(1))

# Cross-validation forecasting
far2 <- function(x, h, level) {
  Arima(x, order = c(2, 0, 0)) |>
    forecast(h = h, level)
}
fc <- cvforecast(series, forecastfun = far2, h = 3, level = 95,
                 forward = TRUE, initial = 1, window = 50)

# Classical conformal prediction with equal weights
scpfc <- scp(fc, symmetric = FALSE, ncal = 50, rolling = TRUE)
print(scpfc)
summary(scpfc)

# Classical conformal prediction with exponential weights
expweight <- function(n) {
  0.99^{n+1-(1:n)}
}
scpfc_exp <- scp(fc, symmetric = FALSE, ncal = 50, rolling = TRUE,
                 weightfun = expweight, kess = TRUE)
```

`update.cpforecast` *Update and repeform cross-validation forecasting and conformal prediction*

Description

Update conformal prediction intervals and other information by applying the `cvforecast` and `conformal` functions.

Usage

```
## S3 method for class 'cpforecast'
update(object, new_data, forecastfun, new_xreg = NULL, ...)
```

Arguments

`object`	An object of class `"cpforecast"`.
`new_data`	A vector of newly available data.
`forecastfun`	Function to return an object of class `"forecast"`. Its first argument must be a univariate time series, and it must have an argument `h` for the forecast horizon and an argument `level` for the confidence level for prediction intervals. If exogenous predictors are used, then it must also have `xreg` and `newxreg` arguments corresponding to the training and test periods, respectively.
`new_xreg`	Newly available exogenous predictor variables passed to `forecastfun` if required. The number of rows should match the length of `new_data`, and the number of columns should match the dimensions of the `xreg` argument in `object`.
`...`	Other arguments are passed to `forecastfun`.

Value

A refreshed object of class `"cpforecast"` with updated fields (e.g., `x`, `MEAN`, `ERROR`, `LOWER`, `UPPER`, and any method-specific components), reflecting newly appended data and re-computed cross-validation forecasts and conformal prediction intervals.

Examples

```
# Simulate time series from an AR(2) model
library(forecast)
series <- arima.sim(n = 200, list(ar = c(0.8, -0.5)), sd = sqrt(1))

# Cross-validation forecasting
far2 <- function(x, h, level) {
  Arima(x, order = c(2, 0, 0)) |>
    forecast(h = h, level)
}
fc <- cvforecast(series, forecastfun = far2, h = 3, level = 95,
                 forward = TRUE, initial = 1, window = 50)
```

```
# Classical conformal prediction with equal weights
scpfc <- conformal(fc, method = "scp", symmetric = FALSE, ncal = 50, rolling = TRUE)

# Update conformal prediction using newly available data
scpfc_update <- update(scpfc, forecastfun = far2, new_data = c(1.5, 0.8, 2.3))
print(scpfc_update)
summary(scpfc_update)
```

`width` *Calculate interval forecast width*

Description

Calculate the mean width of prediction intervals on the validation set. If `window` is not `NULL`, a matrix of the rolling means of interval width is also returned. If `includemedian` is `TRUE`, the information of the median interval width will be returned.

Usage

```
width(
  object,
  ...,
  level = 95,
  includemedian = FALSE,
  window = NULL,
  na.rm = FALSE
)
```

Arguments

`object`	An object of class `"cvforecast"` or `"cpforecast"`.
`...`	Additional inputs if `object` is missing.
`level`	Target confidence level for prediction intervals.
`includemedian`	If `TRUE`, the median interval width will also be returned.
`window`	If not `NULL`, the rolling mean (and rolling median if applicable) matrix for interval width will also be returned.
`na.rm`	A logical indicating whether `NA` values should be stripped before the rolling mean and rolling median computation proceeds.

Value

A list of class `"width"` with the following components:

`width`	Forecast interval width as a multivariate time series, where the hth column holds the interval width for the forecast horizon h. The time index corresponds to the period for which the forecast is produced.

mean	Mean interval width across the validation set.
rollmean	If window is not NULL, a matrix of the rolling means of interval width will be returned.
median	Median interval width across the validation set.
rollmedian	If window is not NULL, a matrix of the rolling medians of interval width will be returned.

Examples

```
# Simulate time series from an AR(2) model
library(forecast)
series <- arima.sim(n = 200, list(ar = c(0.8, -0.5)), sd = sqrt(1))

# Cross-validation forecasting with a rolling window
far2 <- function(x, h, level) {
  Arima(x, order = c(2, 0, 0)) |>
    forecast(h = h, level)
}
fc <- cvforecast(series, forecastfun = far2, h = 3, level = 95,
                 forward = TRUE, initial = 1, window = 50)

# Mean and rolling mean width for interval forecasts on validation set
wid_fc <- width(fc, level = 95, window = 50)
str(wid_fc)
```

Index