# Package: conformalForecast (via r-universe)

October 15, 2026

**Title** Conformal Prediction Methods for Multistep-Ahead Time Series Forecasting

**Version** 0.2.0

**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

**Depends** R (>= 4.1.0), forecast

**Imports** ggdist, rlang, stats, zoo

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

**Config/testthat/edition** 3

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**VignetteBuilder** knitr

**Repository** https://robjhyndman.r-universe.dev

**Date/Publication** 2026-10-15 05:24:50 UTC

**RemoteUrl** https://github.com/xqnwang/conformalForecast

**RemoteRef** HEAD

**RemoteSha** daf667743757d034c2a4d43e98ed42796a4136a3

# Contents

conformalForecast-package . . . 2
accuracy.cvforecast . . . 3
acmcp . . . 5
acp . . . 8
conformal . . . 11
coverage . . . 12
cvforecast . . . 13
lagmatrix . . . 16
ME . . . 17
MSIS . . . 19
pid . . . 20
scp . . . 24
update.cpforecast . . . 27
width . . . 28

**Index** **30**

---

`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)
- Daniele Girolimetto <daniele.girolimetto@unipd.it> (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 a range of summary measures of the out-of-sample forecast accuracy. If `x` is given, the function also measures the test set forecast accuracy. If `x` is not given, the function only produces accuracy measures on the 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 the actual values of the same length as `mean` in `object`. |
| `CV` | If `TRUE`, the cross-validation forecast accuracy will be returned. Defaults to `TRUE`. |
| `period` | The seasonal period of the data. Defaults to `NULL`, in which case it is taken from the frequency of the series. |
| `measures` | A list of accuracy measure functions to compute (such as point_measures or interval_measures). Defaults to `interval_measures`. |
| `byhorizon` | If `TRUE`, accuracy measures will be calculated for each individual forecast horizon `h` separately. Defaults to `FALSE`. |
| `...` | 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: Winkler Score
- MSIS: Mean Scaled Interval Score

## Value

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

## See Also

`point_measures` and `interval_measures` for the measures that can be supplied through `measures`.

Other evaluation functions: `coverage()`, `width()`

## Examples

```
# Simulate time series from an AR(2) model
library(forecast)
set.seed(1)
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, 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)

# Accuracy of conformal prediction intervals
scpfc <- conformal(fc, method = "scp", ncal = 50, rolling = TRUE)
accuracy(scpfc, 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,
         CV = FALSE, 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$level,
  ncal = 10,
  rolling = FALSE,
  integrate = TRUE,
  scorecast = TRUE,
  lr = 0.1,
  Tg = NROW(object$ERROR),
  delta = 0.01,
  Csat = NULL,
  KI = max(abs(object$ERROR), na.rm = TRUE),
  update = FALSE,
  ma_method = c("CSS-ML", "CSS"),
  ...
)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `"cvforecast"`. It must have an argument `x` for the original univariate time series, an argument `MEAN` for the point forecasts and `ERROR` for the forecast errors on the validation set. See the results of a call to `cvforecast`. |
| `alpha` | A numeric vector of significance levels to achieve a desired coverage level $1-\alpha$. Defaults to `1 - 0.01 * object$level`, the levels used in `object`. |
| `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 the learning rate calculation and of the windows for the calibration set. If `rolling = FALSE`, it is used as the initial period of the calibration sets and of the trailing windows for the learning rate calculation. Defaults to `10`. |
| `rolling` | If `TRUE`, a rolling window strategy will be adopted to form the trailing window for the learning rate calculation and the calibration set for the scorecaster if applicable. Otherwise, an expanding window strategy will be used. Defaults to `FALSE`. |
| `integrate` | If `TRUE`, error integration will be included in the update process. Defaults to `TRUE`. |

| | |
|---|---|
| scorecast | If TRUE, scorecasting will be included in the update process. Defaults to TRUE. |
| lr | A positive initial learning rate used for quantile tracking. Defaults to 0.1. |
| Tg | The time that is set to achieve the target absolute coverage guarantee before this. It must be greater than 1 when Csat is not supplied. Defaults to NROW(object$ERROR), the number of cross-validation periods in object. |
| delta | A number in $(0, 1)$. The target absolute coverage guarantee is set to $1 - \alpha - \delta$. Defaults to 0.01. |
| Csat | A positive constant ensuring that by time Tg, an absolute guarantee is of at least $1 - \alpha - \delta$ coverage. Defaults to NULL, in which case it is derived from Tg and delta as 2 / pi * (ceiling(log(Tg) * delta) - 1 / log(Tg)). |
| KI | A non-negative constant to place the integrator on the same scale as the scores. Defaults to max(abs(object$ERROR), na.rm = TRUE), the largest absolute forecast error in object. |
| update | If TRUE, object already holds the results of a previous call and only the newly added time steps are computed; the prediction intervals produced earlier are carried over unchanged. Set by update.cpforecast and not normally set by hand. Defaults to FALSE. |
| ma_method | Estimation method for the MA$(h-1)$ scorecaster. "CSS-ML" uses conditional sum of squares for starting values followed by maximum likelihood. "CSS" uses conditional sum of squares only and may be faster, especially for longer forecast horizons, but can produce different estimates. Defaults to "CSS-ML". |
| ... | Not used. |

## 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 the past errors.

Scorecasts are constructed recursively, so longer forecast horizons require more history before all scorecaster inputs are available. For horizon $h$, the first scorecast can be computed at cross-validation error index ncal + $h(h-1)/2$. The number of successful scorecasts is reported in scorecast_times.

## Value

A list of class c("acmcp", "cpforecast", "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 "acmcp". |
| cp_times | An integer vector giving the number of conformal predictions performed in cross-validation for each forecast horizon. |

`scorecast_times`
: An integer vector giving the number of successful scorecasts for each forecast horizon. Returned when `scorecast = TRUE`.

`MEAN`
: Point forecasts as a multivariate time series, where the $h$th 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 about the conformal prediction model: the resolved arguments in `model$args`, the call and the arguments of the underlying cross-validation in `model$cvforecast`, the learning rates actually used in `model$lr_update`, and the recursion state needed to extend the results in `model$state` and `model$t_last`. The integrator and scorecaster series are returned in `model$integrator` and `model$scorecaster` when `integrate` and `scorecast` are `TRUE`; each holds a `lower` and an `upper` element.

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

`cvforecast` to produce `object`, and `update.cpforecast` to extend the results with new observations.

Other conformal prediction methods: `acp()`, `conformal()`, `pid()`, `scp()`

## Examples

```
# Simulate time series from an AR(2) model
library(forecast)
set.seed(1)
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, window = 50)

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

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

---

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 the original univariate time series, an argument `MEAN` for the point forecasts and `ERROR` for the forecast errors on the validation set. See the results of a call to `cvforecast`. |
| `alpha` | A numeric vector of significance levels to achieve a desired coverage level $1-\alpha$. Defaults to `1 - 0.01 * object$level`, the levels used in `object`. |
| `gamma` | The step size parameter $\gamma > 0$ for the $\alpha$ updating. Defaults to `0.005`. |

| | |
|---|---|
| symmetric | If TRUE, symmetric nonconformity scores (i.e. $\lvert e_{t+h\lvert t}\rvert$) are used. If FALSE, asymmetric nonconformity scores (i.e. $e_{t+h\lvert t}$) are used, and then upper bounds and lower bounds are produced separately. Defaults to FALSE. |
| ncal | Length of the calibration set. If rolling = FALSE, it denotes the initial period of the calibration sets. Otherwise, it indicates the period of every rolling calibration set. Defaults to 10. |
| rolling | If TRUE, a rolling window strategy will be adopted to form the calibration set. Otherwise, an expanding window strategy will be used. Defaults to FALSE. |
| 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. Defaults to 1, the order-statistic estimator used by standard conformal calibration. |
| update | If TRUE, object already holds the results of a previous call and only the newly added time steps are computed; the prediction intervals produced earlier are carried over unchanged. Set by update.cpforecast and not normally set by hand. Defaults to FALSE. |
| na.rm | If TRUE, corresponding entries in the sample values are removed if they are NA when calculating the sample quantile. Defaults to TRUE. |
| ... | Other arguments are passed to the weighted_quantile function for the 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", "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 "acp". |
| cp_times | An integer vector giving the number of conformal predictions performed in cross-validation for each forecast horizon. |
| MEAN | Point forecasts as a multivariate time series, where the $h$th 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 $e_{t+h\lvert t} = y_{t+h} - \hat{y}_{t+h\lvert t}$. |

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 about the conformal prediction model: the resolved arguments in `model$args`, the call and the arguments of the underlying cross-validation in `model$cvforecast`, and the sequence of updated significance levels in `model$alpha_update`. The latter holds a single element `alpha` when `symmetric = TRUE`, and the two elements `lower` and `upper` when `symmetric = FALSE`; each is a list with one multivariate time series per confidence level, laid out like `MEAN`.

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.

## See Also

`cvforecast` to produce `object`, and `update.cpforecast` to extend the results with new observations.

Other conformal prediction methods: `acmcp()`, `conformal()`, `pid()`, `scp()`

## Examples

```
# Simulate time series from an AR(2) model
library(forecast)
set.seed(1)
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, window = 50)

# ACP with asymmetric nonconformity scores and rolling calibration sets
acpfc <- acp(fc, 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 the original univariate time series, an argument MEAN for the point forecasts and ERROR for the forecast errors on the 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). Defaults to "scp". |

## Value

An object whose class depends on the method invoked.

## See Also

cvforecast to produce object, and update.cpforecast to extend the results with new observations.

Other conformal prediction methods: acmcp(), acp(), pid(), scp()

## Examples

```
# Simulate time series from an AR(2) model
library(forecast)
set.seed(1)
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, window = 50)
```

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

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

---

`coverage` *Calculate interval forecast coverage*

---

## Description

Calculate the mean coverage and the `ifinn` matrix for the prediction intervals on the validation set. If `window` is not `NULL`, a matrix of the rolling means of the 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"`. |
| `...` | Arguments `x`, `LOWER`, and `UPPER` if `object` is missing. Bounds may be time-series matrices or lists keyed by confidence level. |
| `level` | Target confidence level for the prediction intervals. Only one level can be specified. Defaults to `95`. |
| `window` | If not `NULL`, the rolling mean matrix for the coverage is also returned. Defaults to `NULL`. |
| `na.rm` | A logical indicating whether `NA` values should be stripped before the rolling mean computation proceeds. Defaults to `FALSE`. |

## Value

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

| | |
|---|---|
| `mean` | Mean coverage across the validation set. |
| `ifinn` | An indicator matrix as a multivariate time series, where the $h$th 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 the interval forecast coverage will be returned. |

## See Also

cvforecast and the conformal methods, which produce object.

Other evaluation functions: accuracy.cvforecast(), width()

## Examples

```
# Simulate time series from an AR(2) model
library(forecast)
set.seed(1)
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, window = 50)

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

# Coverage calculated directly from interval components
coverage(x = fc$x, LOWER = fc$LOWER, UPPER = fc$UPPER, level = 95)
```

---

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 the test periods, respectively. |
| `h` | Forecast horizon. Defaults to `1`. |
| `level` | Confidence level for prediction intervals, as a numeric vector of percentages. It must not be `NULL`, since the conformal methods need at least one interval to start from. Defaults to `c(80, 95)`. |
| `forward` | If `TRUE`, the final forecast origin for forecasting is $y_T$. Otherwise, the final forecast origin is $y_{T-1}$. Defaults to `TRUE`. |
| `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. Defaults to `NULL`. |
| `initial` | Initial period of the time series where no cross-validation forecasting is performed. Defaults to `1`. |
| `window` | Length of the rolling window. Defaults to `NULL`, in which case a rolling window will not be used. |
| `...` | Other arguments are passed to `forecastfun`. |

### Details

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

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

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 $h$th 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 the forecasts using all available observations, see e.g. `forecast.ets` for more details.

## See Also

`conformal` and the conformal methods `scp`, `acp`, `pid` and `acmcp`, which all take the result of `cvforecast()` as their input; `update.cpforecast` to extend the results with new observations.

## Examples

```
# Simulate time series from an AR(2) model
library(forecast)
set.seed(1)
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, 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,
                      xreg = matrix(rnorm(406), ncol = 2, nrow = 203),
                      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 a multivariate time series. |
| lag | A vector of finite integer 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.

### See Also

cvforecast, which uses this function to align the forecasts and the forecast errors by horizon.

### Examples

```
set.seed(1)
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 the test data. |

| | |
|---|---|
| na.rm | If TRUE, remove missing values before calculating the measure. Defaults to TRUE. |
| ... | Additional arguments for each measure. |
| actual | A numeric vector of the realised values matching the forecasts. Required by the percentage measures. |
| train | A numeric vector of responses used to train the model. Required by the scaled measures. |
| demean | Should the response be demeaned (for MASE and RMSSE)? Defaults to FALSE. |
| period | The seasonal period of the data. Required by the scaled measures. |
| d | Should the response model include a first difference? Defaults to period == 1. |
| D | Should the response model include a seasonal difference? Defaults to period > 1. |

## Format

An object of class list of length 8.

## Value

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

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

## See Also

interval_measures for the interval counterparts, and accuracy.cvforecast, which applies these measures to a cross-validation or a conformal prediction object.

## 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 the interval forecasts. |
| upper | A numeric vector of upper bounds of the interval forecasts. |
| actual | A numeric vector of realised values. |
| train | A numeric vector of responses used to train the model. Required by the scaled scores. |
| level | The nominal level of the forecast interval (e.g., 95 or 0.95). Defaults to 95. |
| period | The seasonal period of the data. Required by the scaled scores. |
| d | Should the response model include a first difference? Defaults to period == 1. |
| D | Should the response model include a seasonal difference? Defaults to period > 1. |
| na.rm | If TRUE, remove missing values before calculating the measure. Defaults to TRUE. |
| ... | Additional arguments for each measure. |

### Format

An object of class `list` of length 2.

### Value

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

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

### See Also

`point_measures` for the point counterparts, and `accuracy.cvforecast`, which applies these measures to a cross-validation or a conformal prediction object.

### 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 = NROW(object$ERROR),
  delta = 0.01,
  Csat = NULL,
  KI = max(abs(object$ERROR), na.rm = TRUE),
  update = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| object | An object of class "cvforecast". It must have an argument x for the original univariate time series, an argument MEAN for the point forecasts and ERROR for the forecast errors on the validation set. See the results of a call to cvforecast. |
| alpha | A numeric vector of significance levels to achieve a desired coverage level $1-\alpha$. Defaults to 1 - 0.01 * object$level, the levels used in object. |
| symmetric | If TRUE, symmetric nonconformity scores (i.e. $\lvert e_{t+h\vert t}\rvert$) are used. If FALSE, asymmetric nonconformity scores (i.e. $e_{t+h\vert t}$) are used, and then upper bounds and lower bounds are produced separately. Defaults to FALSE. |
| 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 the learning rate calculation and of the windows for the calibration set. If rolling = FALSE, it is used as the initial period of the calibration sets and of the trailing windows for the learning rate calculation. Defaults to 10. |
| rolling | If TRUE, a rolling window strategy will be adopted to form the trailing window for the learning rate calculation and the calibration set for the scorecaster if applicable. Otherwise, an expanding window strategy will be used. Defaults to FALSE. |
| integrate | If TRUE, error integration will be included in the update process. Defaults to TRUE. |
| scorecast | If TRUE, scorecasting will be included in the update process, and scorecastfun should be given. Defaults to !symmetric. |
| 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. Defaults to NULL, which is only allowed when scorecast = FALSE. |
| lr | A positive initial learning rate used for quantile tracking. Defaults to 0.1. |
| Tg | The time that is set to achieve the target absolute coverage guarantee before this. It must be greater than 1 when Csat is not supplied. Defaults to NROW(object$ERROR), the number of cross-validation periods in object. |
| delta | A number in $(0, 1)$. The target absolute coverage guarantee is set to $1 - \alpha - \delta$. Defaults to 0.01. |
| Csat | A positive constant ensuring that by time Tg, an absolute guarantee is of at least $1 - \alpha - \delta$ coverage. Defaults to NULL, in which case it is derived from Tg and delta as 2 / pi * (ceiling(log(Tg) * delta) - 1 / log(Tg)). |

| | |
|---|---|
| KI | A non-negative constant to place the integrator on the same scale as the scores. Defaults to `max(abs(object$ERROR), na.rm = TRUE)`, the largest absolute forecast error in `object`. |
| update | If `TRUE`, `object` already holds the results of a previous call and only the newly added time steps are computed; the prediction intervals produced earlier are carried over unchanged. Set by `update.cpforecast` and not normally set by hand. Defaults to `FALSE`. |
| ... | 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 $r_t(x) = K_{\mathrm{I}} \tan\left(x \log(t) / \left(t C_{\text{sat}}\right)\right)$, where we set $\tan(x) = \mathrm{sign}(x) \cdot \infty$ for $x \notin [-\pi/2, \pi/2]$, and $C_{\text{sat}}$ , $K_{\mathrm{I}} > 0$ are constants that we choose heuristically.
- Scorecasting part (D) is $\hat{s}_{t+h|t}$, the forecast generated by training a scorecaster based on the nonconformity scores available at time $t$.

## Value

A list of class `c("pid", "cpforecast", "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 "pid". |
| cp_times | An integer vector giving the number of conformal predictions performed in cross-validation for each forecast horizon. |
| MEAN | Point forecasts as a multivariate time series, where the $h$th 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 about the conformal prediction model: the resolved arguments in `model$args`, the call and the arguments of the underlying cross-validation in `model$cvforecast`, the learning rates actually used in `model$lr_update`, and the recursion state needed to extend the results in `model$state` and `model$t_last`. The integrator and scorecaster series are returned in `model$integrator` and `model$scorecaster` when `integrate` and `scorecast` are `TRUE`; when `symmetric = FALSE` each of the two holds a `lower` and an `upper` element.

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. (2023). "Conformal PID control for time series prediction", *Advances in Neural Information Processing Systems*, **36**, 23047–23074.

## See Also

`cvforecast` to produce `object`, and `update.cpforecast` to extend the results with new observations.

Other conformal prediction methods: `acmcp()`, `acp()`, `conformal()`, `scp()`

## Examples

```
# Simulate time series from an AR(2) model
library(forecast)
set.seed(1)
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, window = 50)

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

# PID without scorecaster
pidfc_nsf <- pid(fc, ncal = 50, rolling = TRUE, scorecast = FALSE,
                 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, ncal = 50, rolling = TRUE, scorecastfun = naivefun,
             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 the original univariate time series, an argument MEAN for the point forecasts and ERROR for the forecast errors on the validation set. See the results of a call to cvforecast. |
| alpha | A numeric vector of significance levels to achieve a desired coverage level $1-\alpha$. Defaults to 1 - 0.01 * object$level, the levels used in object. |
| symmetric | If TRUE, symmetric nonconformity scores (i.e. $\lvert e_{t+h\vert t}\rvert$) are used. If FALSE, asymmetric nonconformity scores (i.e. $e_{t+h\vert t}$) are used, and then upper bounds and lower bounds are produced separately. Defaults to FALSE. |
| ncal | Length of the calibration set. If rolling = FALSE, it denotes the initial period of the calibration sets. Otherwise, it indicates the period of every rolling calibration set. Defaults to 10. |

| | |
|---|---|
| rolling | If TRUE, a rolling window strategy will be adopted to form the calibration set. Otherwise, an expanding window strategy will be used. Defaults to FALSE. |
| 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. Defaults to 1, the order-statistic estimator used by standard conformal calibration. |
| weightfun | Function to return a vector of weights used for the sample quantile computation. Its first argument must be an integer indicating the number of observations for which weights are generated. Defaults to NULL, in which case equal weights are used. Currently, only non-data-dependent weights are supported. |
| kess | If TRUE, Kish's effective sample size is used for the sample quantile computation. This adjustment is only supported for the continuous quantile estimators, quantiletype = 4:9. Defaults to FALSE. |
| update | If TRUE, object already holds the results of a previous call and only the newly added time steps are computed; the prediction intervals produced earlier are carried over unchanged. Set by update.cpforecast and not normally set by hand. Defaults to FALSE. |
| na.rm | If TRUE, corresponding entries in the sample values and weights are removed if either is NA when calculating the sample quantile. Defaults to TRUE. |
| ... | 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 $(1-\alpha)$-quantiles $\hat{q}_{t+h|t}$ are computed successively on expanding calibration sets $s_{1+h|1}, \ldots, s_{t|t-h}$, for $t = \mathrm{ncal} + h, \ldots, T$. 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 the same length ncal.

If symmetric is FALSE, $s^u_{t+h|t} = e_{t+h|t}$ for the upper interval bounds and $s^l_{t+h|t} = -e_{t+h|t}$ for the lower bounds. Instead of computing the $(1-\alpha)$-quantile, the $(1-\alpha/2)$-quantiles for the lower bound $\hat{q}^l_{t+h|t}$ and the 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", "cpforecast", "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 | An integer vector giving the number of conformal predictions performed in cross-validation for each forecast horizon. |

| | |
|---|---|
| MEAN | Point forecasts as a multivariate time series, where the $h$th 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 about the conformal prediction model: the resolved arguments in `model$args`, and the call and the arguments of the underlying cross-validation in `model$cvforecast`. |

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

`cvforecast` to produce `object`, `update.cpforecast` to extend the results with new observations, and `weighted_quantile` for the weighted quantile estimator used here.

Other conformal prediction methods: `acmcp()`, `acp()`, `conformal()`, `pid()`

## Examples

```
# Simulate time series from an AR(2) model
library(forecast)
set.seed(1)
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, window = 50)

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

```
                    weightfun = expweight, kess = TRUE, quantiletype = 7)
```

---

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

---

## Description

Update conformal prediction intervals and other information by applying the `cvforecast` and `conformal` functions. Only the newly added time steps are computed: the prediction intervals produced earlier are carried over unchanged.

## Usage

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

## Arguments

| | |
|---|---|
| `object` | An object of class `"cpforecast"`, as returned by one of the conformal prediction methods. |
| `new_data` | A non-empty numeric 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 the 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`. These rows extend `object$xreg` and correspond to the periods immediately following it. Defaults to `NULL`. |
| `...` | 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 the newly appended data and the re-computed cross-validation forecasts and conformal prediction intervals.

## See Also

`cvforecast` and `conformal`, which produce the object to be updated.

## Examples

```
# Simulate time series from an AR(2) model
library(forecast)
set.seed(1)
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, window = 50)

# Classical conformal prediction with equal weights
scpfc <- conformal(fc, method = "scp", 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 the prediction intervals on the validation set. If `window` is not `NULL`, a matrix of the rolling means of the 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"`. |
| `...` | Time-series matrices `LOWER` and `UPPER` if `object` is missing. They may also be lists keyed by confidence level. |

| | |
|---|---|
| level | Target confidence level for the prediction intervals. Only one level can be specified. Defaults to 95. |
| includemedian | If TRUE, the median interval width will also be returned. Defaults to FALSE. |
| window | If not NULL, the rolling mean (and the rolling median if applicable) matrix for the interval width will also be returned. Defaults to NULL. |
| na.rm | A logical indicating whether NA values should be stripped before the rolling mean and rolling median computation proceeds. Defaults to FALSE. |

## Value

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

| | |
|---|---|
| width | Forecast interval width as a multivariate time series, where the $h$th 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 the 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 the interval width will be returned. |

## See Also

cvforecast and the conformal methods, which produce object.

Other evaluation functions: accuracy.cvforecast(), coverage()

## Examples

```
# Simulate time series from an AR(2) model
library(forecast)
set.seed(1)
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, window = 50)

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

# Width calculated directly from interval components
width(LOWER = fc$LOWER, UPPER = fc$UPPER, level = 95)
```

# Index


∗ **conformal prediction methods**
acmcp, 5
acp, 8
conformal, 11
pid, 20
scp, 24
∗ **datasets**
ME, 17
MSIS, 19
∗ **evaluation functions**
accuracy.cvforecast, 3
coverage, 12
width, 28
∗ **package**
conformalForecast-package, 2

accuracy.cpforecast *(accuracy.cvforecast)*, 3
accuracy.cvforecast, 3, *13*, *18*, *20*, *29*
acmcp, 5, *10*, *11*, *15*, *23*, *26*
acp, *7*, 8, *11*, *15*, *23*, *26*

conformal, *7*, *10*, 11, *15*, *23*, *26*, *27*
conformalForecast *(conformalForecast-package)*, 2
conformalForecast-package, 2
coverage, *4*, 12, *29*
cvforecast, *5*, *7*, *8*, *10*, *11*, 13, *13*, *16*, *21*, *23*, *24*, *26*, *27*, *29*

forecast.ets, *15*

interval_measures, *3*, *4*, *18*
interval_measures *(MSIS)*, 19

lagmatrix, 16

MAE *(ME)*, 17
MAPE *(ME)*, 17
MASE *(ME)*, 17
ME, 17
MPE *(ME)*, 17
MSE *(ME)*, 17
MSIS, 19

pid, *7*, *10*, *11*, *15*, 20, *26*
point_measures, *3*, *4*, *20*
point_measures *(ME)*, 17
print.coverage *(coverage)*, 12
print.cvforecast *(cvforecast)*, 13
print.summary.cvforecast *(cvforecast)*, 13
print.width *(width)*, 28

RMSE *(ME)*, 17
RMSSE *(ME)*, 17

scp, *7*, *10*, *11*, *15*, *23*, 24
summary.cvforecast *(cvforecast)*, 13

update.cpforecast, *6*, *7*, *9*–*11*, *15*, *22*, *23*, *25*, *26*, 27

weighted_quantile, *9*, *25*, *26*
width, *4*, *13*, 28
winkler_score *(MSIS)*, 19