

# Package ‘caretForecast’

February 1, 2026

**Title** Conformal Time Series Forecasting Using State of Art Machine Learning Algorithms

**Version** 0.1.3

**Description** Conformal time series forecasting using the caret infrastructure.

It provides access to state-of-the-art machine learning models for forecasting applications. The hyperparameter of each model is selected based on time series cross-validation, and forecasting is done recursively.

**License** GPL (>= 3)

**URL** <https://taf-society.github.io/caretForecast/>,  
<https://github.com/taf-society/caretForecast>

**BugReports** <https://github.com/taf-society/caretForecast/issues>

**Depends** R (>= 3.6)

**Imports** forecast (>= 8.15), caret (>= 6.0.88), magrittr (>= 2.0.1),  
methods (>= 4.1.1), dplyr (>= 1.0.9), generics (>= 0.1.3)

**Suggests** Cubist (>= 0.3.0), knitr (>= 1.29), testthat (>= 2.3.2)

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Author** Resul Akay [aut, cre]

**Maintainer** Resul Akay <resul.akay@taf-society.org>

**Repository** CRAN

**Date/Publication** 2026-02-01 06:10:26 UTC

## Contents

ARml	2
conformalRegressor	5
conformalRegressorByHorizon	5

forecast.ARml . . . . .	6
get_var_imp . . . . .	7
predict.conformalRegressor . . . . .	8
predict.conformalRegressorByHorizon . . . . .	9
retail . . . . .	10
retail_wide . . . . .	10
split_ts . . . . .	11
suggested_methods . . . . .	11

<b>Index</b>	<b>13</b>
--------------	-----------

---

## ARml

*Autoregressive forecasting using various Machine Learning models.*

---

### Description

Autoregressive forecasting using various Machine Learning models.

### Usage

```
ARml(
  y,
  max_lag = 5,
  xreg = NULL,
  caret_method = "cubist",
  metric = "RMSE",
  pre_process = NULL,
  cv = TRUE,
  cv_horizon = 4,
  initial_window = NULL,
  fixed_window = FALSE,
  verbose = TRUE,
  seasonal = TRUE,
  K = frequency(y)/2,
  tune_grid = NULL,
  lambda = NULL,
  BoxCox_method = c("guerrero", "loglik"),
  BoxCox_lower = -1,
  BoxCox_upper = 2,
  BoxCox_biasadj = FALSE,
  BoxCox_fvar = NULL,
  allow_parallel = FALSE,
  calibrate = TRUE,
  calibration_horizon = NULL,
  n_cal_windows = NULL,
  ...
)
```

## Arguments

<code>y</code>	A univariate time series object.
<code>max_lag</code>	Maximum value of lag.
<code>xreg</code>	Optional. A numerical vector or matrix of external regressors, which must have the same number of rows as <code>y</code> . (It should not be a data frame.).
<code>caret_method</code>	A string specifying which classification or regression model to use. Possible values are found using <code>names(getModelInfo())</code> . A list of functions can also be passed for a custom model function. See <a href="https://topepo.github.io/caret/">https://topepo.github.io/caret/</a> for details.
<code>metric</code>	A string that specifies what summary metric will be used to select the optimal model. See <code>?caret::train</code> .
<code>pre_process</code>	A string vector that defines a pre-processing of the predictor data. Current possibilities are "BoxCox", "YeoJohnson", "expoTrans", "center", "scale", "range", "knnImpute", "bagImpute", "medianImpute", "pca", "ica" and "spatialSign". The default is no pre-processing. See <code>preProcess</code> and <code>trainControl</code> on the procedures and how to adjust them. Pre-processing code is only designed to work when <code>x</code> is a simple matrix or data frame.
<code>cv</code>	Logical, if <code>cv</code> = TRUE model selection will be done via cross-validation. If <code>cv</code> = FALSE user need to provide a specific model via <code>tune_grid</code> argument.
<code>cv_horizon</code>	The number of consecutive values in test set sample.
<code>initial_window</code>	The initial number of consecutive values in each training set sample.
<code>fixed_window</code>	Logical, if FALSE, all training samples start at 1.
<code>verbose</code>	A logical for printing a training log.
<code>seasonal</code>	Boolean. If <code>seasonal</code> = TRUE the fourier terms will be used for modeling seasonality.
<code>K</code>	Maximum order(s) of Fourier terms
<code>tune_grid</code>	A data frame with possible tuning values. The columns are named the same as the tuning parameters. Use <code>getModelInfo</code> to get a list of tuning parameters for each model or see <a href="https://topepo.github.io/caret/available-models.html">https://topepo.github.io/caret/available-models.html</a> . (NOTE: If given, this argument must be named.)
<code>lambda</code>	BoxCox transformation parameter. If <code>lambda</code> = NULL If <code>lambda</code> = "auto", then the transformation parameter <code>lambda</code> is chosen using <a href="#">BoxCox.lambda</a> .
<code>BoxCox_method</code>	<a href="#">BoxCox.lambda</a> argument. Choose method to be used in calculating <code>lambda</code> .
<code>BoxCox_lower</code>	<a href="#">BoxCox.lambda</a> argument. Lower limit for possible <code>lambda</code> values.
<code>BoxCox_upper</code>	<a href="#">BoxCox.lambda</a> argument. Upper limit for possible <code>lambda</code> values.
<code>BoxCox_biasadj</code>	<a href="#">InvBoxCox</a> argument. Use adjusted back-transformed mean for Box-Cox transformations. If transformed data is used to produce forecasts and fitted values, a regular back transformation will result in median forecasts. If <code>biasadj</code> is TRUE, an adjustment will be made to produce mean forecasts and fitted values.
<code>BoxCox_fvar</code>	<a href="#">InvBoxCox</a> argument. Optional parameter required if <code>biasadj</code> =TRUE. Can either be the forecast variance, or a list containing the interval level, and the corresponding upper and lower intervals.

**allow\_parallel** If a parallel backend is loaded and available, should the function use it?  
**calibrate** Logical. If TRUE, performs rolling-origin calibration to compute horizon-specific conformal prediction intervals. This produces properly calibrated intervals that widen with forecast horizon (trumpet shape). Default is TRUE.  
**calibration\_horizon**  
**n\_cal\_windows** Number of rolling windows for calibration. If NULL (default), automatically determined based on data length (max 50).  
**...** Ignored.

### Value

A list class of forecast containing the following elements

- **x** : The input time series
- **method** : The name of the forecasting method as a character string
- **mean** : Point forecasts as a time series
- **lower** : Lower limits for prediction intervals
- **upper** : Upper limits for prediction intervals
- **level** : The confidence values associated with the prediction intervals
- **model** : A list containing information about the fitted model
- **newx** : A matrix containing regressors
- **calibration** : Horizon-specific conformal calibration scores (if calibrate=TRUE)

### Author(s)

Resul Akay

### Examples

```

library(caretForecast)

train_data <- window(AirPassengers, end = c(1959, 12))

test <- window(AirPassengers, start = c(1960, 1))

ARml(train_data, caret_method = "lm", max_lag = 12) -> fit

forecast(fit, h = length(test)) -> fc

autoplot(fc) + autolayer(test)

accuracy(fc, test)

```

---

conformalRegressor *Fit a conformal regressor.*

---

## Description

Fit a conformal regressor.

## Usage

```
conformalRegressor(residuals, sigmas = NULL)
```

## Arguments

residuals	Model residuals.
sigmas	A vector of difficulty estimates

## Value

A conformalRegressor object

## Author(s)

Resul Akay

## References

Boström, H., 2022. crepes: a Python Package for Generating Conformal Regressors and Predictive Systems. In Conformal and Probabilistic Prediction and Applications. PMLR, 179.

---

conformalRegressorByHorizon

*Fit a horizon-specific conformal regressor for time series forecasting.*

---

## Description

This function creates a conformal regressor that accounts for increasing uncertainty at longer forecast horizons. It uses separate nonconformity score distributions for each horizon  $h=1,2,3,\dots$ , resulting in prediction intervals that naturally widen as the forecast horizon increases (trumpet-shaped intervals).

## Usage

```
conformalRegressorByHorizon(horizon_errors)
```

## Arguments

`horizon_errors` A named list where each element contains sorted absolute errors for that horizon.  
Names should be "h1", "h2", etc. This is typically produced by `calibrate_horizon_scores()`.

## Value

A `conformalRegressorByHorizon` object containing:

<code>alphas_by_horizon</code>	List of sorted nonconformity scores for each horizon
<code>max_horizon</code>	Maximum calibrated horizon
<code>n_samples</code>	Number of calibration samples per horizon

## Author(s)

Resul Akay

## References

Boström, H., 2022. *crepes*: a Python Package for Generating Conformal Regressors and Predictive Systems. In *Conformal and Probabilistic Prediction and Applications*. PMLR, 179.

Stankeviciute, K., Alaa, A. M., & van der Schaar, M., 2021. Conformal Time-series Forecasting. *NeurIPS 2021*.

---

<code>forecast.ARml</code>	<i>Forecasting using ARml model</i>
----------------------------	-------------------------------------

---

## Description

Forecasting using ARml model

## Usage

```
## S3 method for class 'ARml'
forecast(object, h = frequency(object$y), xreg = NULL, level = c(80, 95), ...)
```

## Arguments

<code>object</code>	An object of class "ARml", the result of a call to ARml.
<code>h</code>	forecast horizon
<code>xreg</code>	Optionally, a numerical vector or matrix of future external regressors
<code>level</code>	Confidence level for prediction intervals.
<code>...</code>	Ignored

**Value**

A list class of forecast containing the following elements

- x : The input time series
- method : The name of the forecasting method as a character string
- mean : Point forecasts as a time series
- lower : Lower limits for prediction intervals
- upper : Upper limits for prediction intervals
- level : The confidence values associated with the prediction intervals
- model : A list containing information about the fitted model
- newxreg : A matrix containing regressors

**Author(s)**

Resul Akay

**Examples**

```
library(caretForecast)

train_data <- window(AirPassengers, end = c(1959, 12))

test <- window(AirPassengers, start = c(1960, 1))

ARml(train_data, caret_method = "lm", max_lag = 12) -> fit

forecast(fit, h = length(test), level = c(80,95)) -> fc

autoplot(fc)+ autolayer(test)

accuracy(fc, test)
```

get\_var\_imp

*Variable importance for forecasting model.*

**Description**

Variable importance for forecasting model.

**Usage**

```
get_var_imp(object, plot = TRUE)
```

**Arguments**

object	A list class of ARml or forecast object derived from ARml
plot	Boolean, if TRUE, variable importance will be plotted.

**Value**

A list class of "varImp.train". See [varImp](#) or a "trellis" plot.

**Author(s)**

Resul Akay

**Examples**

```
train <- window(AirPassengers, end = c(1959, 12))

test <- window(AirPassengers, start = c(1960, 1))

ARml(train, caret_method = "lm", max_lag = 12,
pre_process = "center") -> fit

forecast(fit, h = length(test), level = c(80,95)) -> fc

autoplot(fc)+ autolayer(test)

accuracy(fc, test)

get_var_imp(fc, plot = TRUE)
```

***predict.conformalRegressor***

*Predict a conformalRegressor*

**Description**

Predict a conformalRegressor

**Usage**

```
## S3 method for class 'conformalRegressor'
predict(
  object,
  y_hat = NULL,
  sigmas = NULL,
  confidence = 0.95,
  y_min = -Inf,
  y_max = Inf,
  ...
)
```

**Arguments**

object	A conformalRegressor object
y_hat	Predicted values
sigmas	Difficulty estimates
confidence	Confidence level
y_min	The minimum value to include in prediction intervals
y_max	The maximum value to include in prediction intervals
...	Ignored

**Value**

Prediction intervals

**Author(s)**

Resul Akay

---

predict.conformalRegressorByHorizon

*Predict intervals from a horizon-specific conformal regressor*

---

**Description**

This function generates prediction intervals that account for increasing uncertainty at longer forecast horizons. Each horizon  $h$  uses its own calibrated nonconformity score distribution, resulting in trumpet-shaped prediction intervals.

**Usage**

```
## S3 method for class 'conformalRegressorByHorizon'
predict(
  object,
  y_hat = NULL,
  confidence = 0.95,
  y_min = -Inf,
  y_max = Inf,
  ...
)
```

**Arguments**

object	A conformalRegressorByHorizon object
y_hat	Predicted values (one per horizon)
confidence	Confidence level(s) between 0 and 1 (e.g., 0.95 for 95 percent)
y_min	The minimum value to include in prediction intervals
y_max	The maximum value to include in prediction intervals
...	Ignored

**Value**

A data frame with lower and upper bounds for each confidence level

**Author(s)**

Resul Akay

`retail`

*Grouped sales data from an Australian Retailer*

**Description**

A dataset containing 42 products' sales

**Usage**

`retail`

**Format**

A data class of "tbl\_df", "tbl", "data.frame" with 13986 rows and 3 columns:

**date** date

**item** products

**value** sales

**Source**

<https://robjhyndman.com/data/ausretail.csv>

`retail_wide`

*Sales data from an Australian Retailer in time series format*

**Description**

A dataset containing 42 products' sales

**Usage**

`retail_wide`

**Format**

An object of class mts (inherits from ts, matrix) with 333 rows and 43 columns.

This data set is the wide format of `retail` data.

**Source**

<https://robjhyndman.com/data/ausretail.csv>

---

split\_ts

*Split a time series into training and testing sets*

---

**Description**

Split a time series into training and testing sets

**Usage**

```
split_ts(y, test_size = 10)
```

**Arguments**

y	A univariate time series
test_size	The number of observations to keep in the test set

**Value**

A list with train and test elements

**Author(s)**

Resul Akay

**Examples**

```
dlist <- split_ts(retail_wide[,1], test_size = 12)
```

---

suggested\_methods

*Suggested methods for ARml*

---

**Description**

Suggested methods for ARml

**Usage**

```
suggested_methods()
```

**Value**

A character vector of Suggested methods

**Author(s)**

Resul Akay

**Examples**

```
suggested_methods()
```

# Index

```
* datasets
  retail, 10
  retail_wide, 10

  ARml, 2

  BoxCox.lambda, 3

  conformalRegressor, 5
  conformalRegressorByHorizon, 5

  forecast.ARml, 6

  get_var_imp, 7

  InvBoxCox, 3

  predict.conformalRegressor, 8
  predict.conformalRegressorByHorizon, 9

  retail, 10, 10
  retail_wide, 10

  split_ts, 11
  suggested_methods, 11

  varImp, 8
```