Package 'utsf'

July 21, 2026

Title Univariate Time Series Forecasting

Version 1.3.4

Description An engine for univariate time series forecasting using different regression models in an autoregressive way. The engine provides an uniform interface for applying the different models. Furthermore, it is extensible so that users can easily apply their own regression models to univariate time series forecasting and benefit from all the features of the engine, such as preprocessings or estimation of forecast accuracy.

Maintainer Francisco Martinez <fmartin@ujaen.es>

License MIT + file LICENSE

URL https://github.com/franciscomartinezdelrio/utsf

BugReports https://github.com/franciscomartinezdelrio/utsf/issues

Imports Cubist, FNN, forecast, generics, ggplot2, ipred, methods, ranger, rpart, vctsfr, xgboost

Suggests knitr, nnet, randomForest, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 8.0.0

VignetteBuilder knitr

NeedsCompilation no

Author Maria Pilar Frias-Bustamante [aut] (ORCID: <https://orcid.org/0000-0001-6886-0953>), Francisco Martinez [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-5206-1898>)

Repository CRAN

Date/Publication 2026-07-21 08:30:02 UTC

Contents

`autoplot.utsf_forecast` *Create a ggplot object from an* `utsf_forecast` *object*

Description

Plot the time series and its associated forecast.

Usage

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

Arguments

`object`	An object of class `utsf_forecast`.
`...`	additional parameter.

Value

The `ggplot` object representing a plotting of the time series and its forecast.

Examples

```
m <- create_model(AirPassengers, lags = 1:12, method = "rf")
f <- forecast(m, h = 12)
library(ggplot2)
autoplot(f)
```

build_examples *Build the training examples*

Description

Build the training examples for a regressive model to forecast a time series using lagged values of the series as autoregressive features.

Usage

```
build_examples(timeS, lags)
```

Arguments

timeS	The time series.
lags	An integer vector with the lags used as feature vector in decreasing order.

Value

A list with two fields: 1) a matrix with the features of the examples and 2) a vector with the targets of the examples

Examples

```
build_examples(ts(1:5), lags = 2:1)
```

create_model *Train an univariate time series forecasting model*

Description

This function trains a model from the historical values of a time series using an autoregressive approach: the targets are the historical values and the features of the targets their lagged values.

Usage

```
create_model(
  timeS,
  lags = NULL,
  method = c("knn", "lm", "rt", "mt", "bagging", "rf", "xgboost"),
  trend = c("additive", "multiplicative", "differences", "none"),
  nfd = -1,
  transform_features = TRUE,
  ...
)
```

Arguments

`timeS`	A time series of class `ts` or a numeric vector.
`lags`	An integer vector, in increasing order, expressing the lags used as autoregressive variables. If the default value (`NULL`) is provided, a suitable vector is chosen.
`method`	A string indicating the method used for training and forecasting. Allowed values are: • `"knn"`: k-nearest neighbors (the default) • `"lm"`: linear regression • `"rt"`: regression trees • `"mt"`: model trees • `"bagging"` • `"rf"`: random forests • `"xgboost"`: extreme gradient boosting See details for a brief explanation of the models. It is also possible to use your own regression model, in that case a function explaining how to build your model must be provided, see the vignette for further details.
`trend`	A character indicating the type of preprocessing applied to the time series in order to deal with trending series, see the vignette for details.
`nfd`	In the case that the parameter `trend` has the value "differences", it specifies the order of first differences to be applied. If the default (-1) is used, the order of first differences needed by the time series will be estimated by the `forecast::ndiffs()` function.
`transform_features`	A logical value indicating whether the training features are also transformed if the additive or multiplicative transformation has been used as preprocessing to deal with trending series.
`...`	Parameters for the underlying function that builds the model. If no parameters are provided, the model is normally fitted with its default parameters. See details for the functions used to train the models.

Details

The functions used to build and train the model are:

- KNN: In this case no model is built and the function `FNN::knn.reg()` is used to predict the future values of the time series. By default, k is equal to 3.
- Linear models: Function `stats::lm()` to build the model and the method `stats::predict.lm()` associated with the trained model to forecast the future values of the time series.
- Regression trees: Function `rpart::rpart()` to build the model and the method `rpart::predict.rpart()` associated with the trained model to forecast the future values of the time series.
- Model trees: Function `Cubist::cubist()` to build the model and the method `Cubist::predict.cubist()` associated with the trained model to forecast the future values of the time series. By default, the parameter `committees` is set to 5.

- Bagging: Function `ipred::bagging()` to build the model and the method `ipred::predict.regbagg()` associated with the trained model to forecast the future values of the time series.
- Random forest: Function `ranger::ranger()` to build the model and the method `ranger::predict.ranger()` associated with the trained model to forecast the future values of the time series.
- Extreme gradient boosting: Function `xgboost::xgboost()` to build the model and the method `xgboost::predict.xgboost()` associated with the trained model to forecast the future values of the time series.

Value

An S3 object of class `utsf`, basically a list with, at least, the following components:

`ts`	The time series being forecast.
`features`	A data frame with the features of the training set. The column names of the data frame indicate the autoregressive lags.
`targets`	A vector with the targets of the training set.
`lags`	An integer vector with the autoregressive lags.
`model`	The regression model used recursively to make the forecast.

Examples

```
## Build model using k-nearest neighbors
create_model(AirPassengers, method = "knn")

## Using k-nearest neighbors changing the default k value
create_model(AirPassengers, method = "knn", k = 5)

## Using your own regression model

# Function to build the regression model
my_knn_model <- function(X, y, param) {
  structure(list(X = X, y = y), class = "my_knn")
}
# Function to predict a new example
predict.my_knn <- function(object, new_value) {
  FNN::knn.reg(train = object$X, test = new_value, y = object$y)$pred
}
create_model(AirPassengers, method = my_knn_model)
```

`efa` *Estimate the forecast accuracy of a model on a time series*

Description

It uses an object of class `utsf` to asses the forecasting accuracy of its associated model on its associated time series applying a rolling origin evaluation.

Usage

```
efa(model, h, type = c("normal", "minimum"), size = NULL, prop = NULL)
```

Arguments

model	An object of class utsf with a model trained with a time series.
h	A positive integer. The forecasting horizon.
type	A string. Possible values are "normal" (the default) and "minimum". See the vignette utsf for an explanation of both ways of evaluating forecast accuracy.
size	An integer. It is the size of the test set (how many of the last observations of the time series are used as test set). It can only be used when the type parameter is "normal". By default, it is the length of the forecasting horizon.
prop	A numeric value in the range (0, 1). It is the proportion of the time series used as test set. It can only be used when the type parameter is "normal".

Value

A list with four components:

per_horizon	A matrix with the estimated forecast accuracy per forecasting horizon using several forecasting accuracy measures.
global	The average estimated forecast accuracy for all the horizons. It is computed as the mean of the different rows of the per_horizon component.
test_sets	A matrix with the test sets used in the evaluation. Each row of the matrix is a test set.
predictions	The predictions for the test sets.

Examples

```
m <- create_model(UKgas, lags = 1:4, method = "rt")
efa(m, h = 4, type = "normal", size = 8)
```

---

forecast.utsf *Forecasting a time series*

Description

Forecasting a time series

Usage

```
## S3 method for class 'utsf'
forecast(object, h, PI = FALSE, level = 90, ...)
```

Arguments

object	an object of class utsf embedding a forecasting model for a time series.
h	A positive integer. Number of values to be forecast into the future, i.e., forecast horizon.
PI	If TRUE, prediction intervals are produced using simulation and assuming normally distributed errors.
level	Confidence level for predictions intervals.
...	Other arguments passed to methods

Value

an object of class utsf_forecast with the same components of the model received as first argument, plus several components:

pred	The forecast as an ts object.
lower	Lower limits for prediction interval.
upper	Upper limits for prediction interval.
level	Confidence value associated with the prediction interval

Examples

```
## Forecast time series using k-nearest neighbors
m <- create_model(USAccDeaths, method = "knn")
f <- forecast(m, h = 12)
f$pred
library(ggplot2)
autoplot(f)

## Using k-nearest neighbors changing the default k value
m <- create_model(USAccDeaths, method = "knn", k = 5)
forecast(m, h = 12)

## Using your own regression model

# Function to build the regression model
my_knn_model <- function(X, y, param) {
  structure(list(X = X, y = y), class = "my_knn")
}
# Function to predict a new example
predict.my_knn <- function(object, new_value) {
  FNN::knn.reg(train = object$X, test = new_value, y = object$y)$pred
}
m <- create_model(USAccDeaths, method = my_knn_model)
forecast(m, h = 12)
```

predict.utsf *Prediction from* utsf *objects*

Description

Predict the class of a new observation based on the model associated with the utsf object

Usage

```
## S3 method for class 'utsf'
predict(object, new_value, ...)
```

Arguments

object	object of class utsf.
new_value	a data frame with one row of a new observation.
...	further arguments passed to or from other methods.

Value

a numeric value with the forecast.

tune_grid *Estimate the forecast accuracy of a model on a time series according to a grid of parameters*

Description

It uses an object of class utsf to asses the forecasting accuracy of its associated model on its associated time series applying rolling origin evaluation according to different configurations of model parameters.

Usage

```
tune_grid(
  model,
  h,
  tuneGrid,
  type = c("normal", "minimum"),
  size = NULL,
  prop = NULL
)
```

Arguments

model	An object of class utsf with a model trained with a time series.
h	A positive integer. The forecasting horizon.
tuneGrid	A data frame with possible tuning values. The columns are named as the tuning parameters.
type	A string. Possible values are "normal" (the default) and "minimum". See the vignette utsf for an explanation of both ways of evaluating forecast accuracy.
size	An integer. It is the size of the test set (how many of the last observations of the time series are used as test set). It can only be used when the type parameter is "normal". By default, it is the length of the forecasting horizon.
prop	A numeric value in the range (0, 1). It is the proportion of the time series used as test set. It can only be used when the type parameter is "normal".

Details

The estimation of forecast accuracy is done with the efa() function. The best combination of parameters is used to train the model with all the historical values of the time series and forecast h values ahead.

Value

A list with three components:

tuneGrid	A data frame with the different combination of parameters and the estimated forecast accuracy of a model trained with those parameters.
best	The best combination of parameters according to root mean squared error.
forecast	An object of class utsf_forecast with the forecast for horizon h using the best estimated combination of parameters.

Examples

```
m <- create_model(UKgas, lags = 1:4, method = "knn")
tune_grid(m, h = 4, tuneGrid = expand.grid(k = 1:7), type = "normal", size = 8)
```

Index