Package 'lmForc'

January 3, 2022

Title Linear Model Forecasting

Version 0.1.0

Description Introduces in-sample, out-of-sample, pseudo out-of-sample, and benchmark linear model forecast tests and a new class for working with forecast data: Forecast.

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

Depends R (>= 3.6.0)

Imports methods

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Nelson Rayl [aut, cre]

Maintainer Nelson Rayl <nelsonrayl14@gmail.com>

Repository CRAN

Date/Publication 2022-01-03 18:10:02 UTC

R topics documented:

`autoreg_forc` *Autoregression forecast*

Description

autoreg_forc takes a vector of realized values, an integer number of periods ahead to forecast, an integer number of lags to include in the autoregressive model, a period to end the initial model estimation and begin forecasting, an optional vector of time data associated with the realized values, and an optional integer number of past periods to estimate the model over. An AR(ar_lags) autoregressive model is originally estimated with realized values up to estimation_end minus the number of periods specified in estimation_window. If estimation_window is left NULL then the autoregressive model is estimated with all realized values up to estimation_end. The AR(ar_lags) model is estimated by regressing the realized values on the same realized values that have been lagged by one to ar_lags steps. The AR coefficients of this model are multiplied by lagged values and the present period realized value to create a forecast for one period ahead. If h_ahead is greater than one, this process of forecasting one period ahead is iteratively repeated so that the two period ahead forecast conditions on the one period ahead forecasted value and so on until a h_ahead forecast is obtained. This forecasting process is repeated for each period after estimation_end with AR model coefficients updating as more information would have become available to the forecaster. Optionally returns the coefficients used to create each forecast. Returns an autoregression forecast based on information that **would** have been available at the forecast origin and replicates the forecasts that an AR model would have produced in real-time.

Usage

```
autoreg_forc(
  realized_vec,
  h_ahead,
  ar_lags,
  estimation_end,
  time_vec = NULL,
  estimation_window = NULL,
  return_betas = FALSE
)
```

Arguments

realized_vec	Vector of realized values. This is the series that is being forecasted.
h_ahead	Integer representing the number of periods ahead that is being forecasted.
ar_lags	Integer representing the number of lags included in the AR model.
estimation_end	Value of any class representing when to end the initial coefficient estimation period and begin forecasting.
time_vec	Vector of any class that is equal in length to the realized_vec vector.
estimation_window	Integer representing the number of past periods that the autoregressive model should be estimated over in each period.
return_betas	Boolean, selects whether the coefficients used in each period to create the forecast are returned. If TRUE, a data frame of betas is returned to the Global Environment.

Value

Forecast object that contains the autoregression forecast.

See Also

For a detailed example see the help vignette: `vignette("lmForc",package = "lmForc")`

Examples

```
date <- as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31",
                  "2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31",
                  "2012-03-31", "2012-06-30", "2012-09-30", "2012-12-31"))
y <- c(1.09, 1.71, 1.09, 2.46, 1.78, 1.35, 2.89, 2.11, 2.97, 0.99, 1.31, 2.33)
data <- data.frame(date, y)

autoreg_forc(
  realized_vec = data$y,
  h_ahead = 1L,
  ar_lags = 2L,
  estimation_end = as.Date("2011-06-30"),
  time_vec = data$date,
  estimation_window = 4L,
  return_betas = FALSE
)

autoreg_forc(
  realized_vec = data$y,
  h_ahead = 4L,
  ar_lags = 2L,
  estimation_end = 4L,
  time_vec = NULL,
  estimation_window = NULL
)
```

---

conditional_forc *Linear model forecast conditioned on an input forecast*

Description

`conditional_forc` takes a linear model call, a vector of time data associated with the linear model, and a forecast for each covariate in the linear model. The linear model is estimated once over the entire sample period and the coefficients are multiplied by the forecasts of each covariate. Returns a forecast conditional on forecasts of each covariate. Used to create a forecast for the present period or replicate a forecast made at a specific period in the past.

Usage

```
conditional_forc(lm_call, time_vec, ...)
```

Arguments

`lm_call`	Linear model call of the class lm.
`time_vec`	Vector of any class that is equal in length to the data in `lm_call`.
`...`	One or more forecasts of class Forecast, one forecast for each covariate in the linear model.

Value

`Forecast` object that contains the conditional forecast.

See Also

For a detailed example see the help vignette: `vignette("lmForc",package = "lmForc")`

Examples

```
x1_forecast <- Forecast(
   origin   = as.Date(c("2012-06-30", "2012-06-30", "2012-06-30", "2012-06-30")),
   future   = as.Date(c("2012-09-30", "2012-12-31", "2013-03-31", "2013-06-30")),
   forecast = c(4.14, 4.04, 4.97, 5.12),
   realized = NULL,
   h_ahead  = NULL
)

x2_forecast <- Forecast(
   origin   = as.Date(c("2012-06-30", "2012-06-30", "2012-06-30", "2012-06-30")),
   future   = as.Date(c("2012-09-30", "2012-12-31", "2013-03-31", "2013-06-30")),
   forecast = c(6.01, 6.05, 6.55, 7.45),
   realized = NULL,
   h_ahead  = NULL
)

date <- as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31",
                  "2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31",
                  "2012-03-31", "2012-06-30"))
y  <- c(1.09, 1.71, 1.09, 2.46, 1.78, 1.35, 2.89, 2.11, 2.97, 0.99)
x1 <- c(4.22, 3.86, 4.27, 5.60, 5.11, 4.31, 4.92, 5.80, 6.30, 4.17)
x2 <- c(10.03, 10.49, 10.85, 10.47, 9.09, 10.91, 8.68, 9.91, 7.87, 6.63)
data <- data.frame(date, y, x1, x2)

conditional_forc(
  lm_call = lm(y ~ x1 + x2, data),
  time_vec = data$date,
  x1_forecast, x2_forecast
)
```

forc *Get the forecast slot of a Forecast object*

Description

forc takes a Forecast object and gets the forecast vector of the forecast.

Usage

```
forc(Forecast)
```

Arguments

Forecast	Forecast object.

Value

Vector of forecast values stored in the Forecast object.

Examples

```
## Not run:

forc(Forecast)


## End(Not run)
```

forc,Forecast-method *Get the forecast slot of a Forecast object*

Description

forc takes a Forecast object and gets the forecast vector of the forecast.

Usage

```
## S4 method for signature 'Forecast'
forc(Forecast)
```

Arguments

Forecast	Forecast object.

Value

Vector of forecast values stored in the Forecast object.

Examples

```
## Not run:

forc(Forecast)


## End(Not run)
```

`forc2df` *Collect a Forecast object to a data frame*

Description

`forc2df` takes one or more objects of the Forecast class and collects them into a data frame. Returns a data frame with all of the information that was stored in the Forecast objects. If multiple forecasts are being collected, all forecasts must have identical future and realized values.

Usage

```
forc2df(...)
```

Arguments

`...`	One or multiple forecasts of the class Forecast.

Value

`data.frame` object that contains forecast information.

See Also

For a detailed example see the help vignette: `vignette("lmForc",package = "lmForc")`

Examples

```
## Not run:

forc2df(x1_forecast)

forc2df(x1_forecast, x2_forecast)


## End(Not run)
```

forc<- *Set forecast slot of a Forecast object*

Description

forc takes a Forecast object and sets the forecast vector of the forecast.

Usage

```
forc(Forecast) <- value
```

Arguments

Forecast	Forecast object.
value	Vector of values assigned to the forecast slot of the Forecast.

Value

Forecast object that contains the new forecast vector.

Examples

```
## Not run:

forc(Forecast) <- c(2.45, 2.76, 3.31)


## End(Not run)
```

forc<-,Forecast-method *Set forecast slot of a Forecast object*

Description

forc takes a Forecast object and sets the forecast vector of the forecast.

Usage

```
## S4 replacement method for signature 'Forecast'
forc(Forecast) <- value
```

Arguments

Forecast	Forecast object.
value	Vector of values assigned to the forecast slot of the Forecast.

Value

`Forecast` object that contains the new forecast vector.

Examples

```
## Not run:

forc(Forecast) <- c(2.45, 2.76, 3.31)


## End(Not run)
```

`Forecast` *Create an object of the Forecast class*

Description

An S4 class for storing forecasts. An object of the Forecast class has equal length vectors that contain the time the forecast was made, the future time being forecasted, the forecast, and realized values if available. Optionally includes the number of periods ahead being forecasted.

Usage

```
Forecast(origin, future, forecast, realized = NULL, h_ahead = NULL)
```

Arguments

`origin`	A vector of any class representing the time when the forecast was made.
`future`	A vector of any class representing the time that is being forecasted, i.e. when the forecast will be realized.
`forecast`	A numeric vector of forecasts.
`realized`	Optional numeric vector of realized values, i.e. the true value at the future time.
`h_ahead`	Optional length-one object representing the number of periods ahead being forecasted.

See Also

For a detailed example see the help vignette: `vignette("lmForc",package = "lmForc")`

Examples

```
my_forecast <- Forecast(
   origin   = c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31"),
   future   = c("2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31"),
   forecast = c(4.21, 4.27, 5.32, 5.11),
   realized = c(4.40, 4.45, 4.87, 4.77),
   h_ahead  = 4L
)

origin(my_forecast) <- c("2010-04-01", "2010-07-01", "2010-10-01", "2011-01-01")
future(my_forecast) <- c("2012-04-01", "2012-07-01", "2012-10-01", "2013-01-01")
forc(my_forecast) <- c(8.87, 7.61, 7.56, 5.96)
realized(my_forecast) <- c(6.64, 6.10, 6.33, 6.67)
h_ahead(my_forecast) <- 8L

origin(my_forecast)
future(my_forecast)
forc(my_forecast)
realized(my_forecast)
h_ahead(my_forecast)
```

Forecast-class *S4 class for storing forecasts*

Description

An S4 class for storing forecasts. An object of the Forecast class has equal length vectors that contain the time the forecast was made, the future time being forecasted, the forecast, and realized values if available. Optionally includes the number of periods ahead being forecasted.

Slots

`origin` A vector of any class representing the time when the forecast was made.

`future` A vector of any class representing the time that is being forecasted, i.e. when the forecast will be realized.

`forecast` A numeric vector of forecasts.

`realized` Optional numeric vector of realized values, i.e. the true value at the future time.

`h_ahead` Optional length-one object representing the number of periods ahead being forecasted.

See Also

For a detailed example see the help vignette: `vignette("lmForc",package = "lmForc")`

future *Get the future slot of a Forecast object*

Description

future takes a Forecast object and gets the future vector of the forecast.

Usage

```
future(Forecast)
```

Arguments

Forecast	object.

Value

Vector of future values stored in the Forecast object.

Examples

```
## Not run:

future(Forecast)


## End(Not run)
```

future,Forecast-method *Get the future slot of a Forecast object*

Description

future takes a Forecast object and gets the future vector of the forecast.

Usage

```
## S4 method for signature 'Forecast'
future(Forecast)
```

Arguments

Forecast	object.

Value

Vector of future values stored in the Forecast object.

Examples

```
## Not run:

future(Forecast)


## End(Not run)
```

`future<-` *Set the future slot of a Forecast object*

Description

`future` takes a `Forecast` object and sets the future vector of the forecast.

Usage

```
future(Forecast) <- value
```

Arguments

`Forecast`	Forecast object.
`value`	Vector of values assigned to the future slot of the Forecast.

Value

`Forecast` object that contains the new future vector.

Examples

```
## Not run:

future(Forecast) <- c("2015-03-01", "2015-03-02", "2015-03-03")


## End(Not run)
```

future<-,Forecast-method

Set future slot of a Forecast object

Description

`future` takes a `Forecast` object and sets the future vector of the forecast.

Usage

```
## S4 replacement method for signature 'Forecast'
future(Forecast) <- value
```

Arguments

`Forecast`	Forecast object.
`value`	Vector of values assigned to the future slot of the Forecast.

Value

`Forecast` object that contains the new future vector.

Examples

```
## Not run:

future(Forecast) <- c("2015-03-01", "2015-03-02", "2015-03-03")


## End(Not run)
```

historical_average_forc

Historical average forecast

Description

`historical_average_forc` takes an average function, a vector of realized values, an integer number of periods ahead to forecast, a period to end the initial average estimation and begin forecasting, an optional vector of time data associated with the realized values, and an optional integer number of past periods to estimate the average over. The historical average is originally calculated with realized values up to `estimation_end` minus the number of periods specified in `estimation_window`. If `estimation_window` is left `NULL` then the historical average is calculated with all available realized values up to `estimation_end`. In each period the historical average is set as the `h_ahead` period ahead forecast. This process is iteratively repeated for each period after `estimation_end`

with the historical average updating in each period as more information would have become available to the forecaster. Returns a historical average forecast where the h_ahead period ahead forecast is simply the historical average or rolling window average of the series being forecasted.

Usage

```
historical_average_forc(
  avg_function,
  realized_vec,
  h_ahead,
  estimation_end,
  time_vec = NULL,
  estimation_window = NULL
)
```

Arguments

avg_function	Character, either "mean" or "median". Selects whether forecasts are made using the historical mean or historical median of the series.
realized_vec	Vector of realized values. This is the series that is being forecasted.
h_ahead	Integer representing the number of periods ahead that is being forecasted.
estimation_end	Value of any class representing when to end the initial average estimation period and begin forecasting.
time_vec	Vector of any class that is equal in length to the realized_vec vector.
estimation_window	Integer representing the number of past periods that the historical average should be estimated over in each period.

Value

Forecast object that contains the historical average forecast.

See Also

For a detailed example see the help vignette: vignette("lmForc",package = "lmForc")

Examples

```
date <- as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31",
                  "2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31",
                  "2012-03-31", "2012-06-30"))
y  <- c(1.09, 1.71, 1.09, 2.46, 1.78, 1.35, 2.89, 2.11, 2.97, 0.99)
data <- data.frame(date, y)

historical_average_forc(
  avg_function = "mean",
  realized_vec = data$y,
  h_ahead = 2L,
  estimation_end = as.Date("2011-03-31"),
```

```
  time_vec = data$date,
  estimation_window = 4L
)

historical_average_forc(
  avg_function = "median",
  realized_vec = data$y,
  h_ahead = 4L,
  estimation_end = 4L
)
```

---

h_ahead *Get the h_ahead slot of a h_ahead object*

Description

h_ahead takes a Forecast object and gets the h_ahead vector of the forecast.

Usage

```
h_ahead(Forecast)
```

Arguments

Forecast	Forecast object.

Value

Vector of h_ahead values stored in the Forecast object.

Examples

```
## Not run:

h_ahead(Forecast)


## End(Not run)
```

h_ahead,Forecast-method *Get the h_ahead slot of a h_ahead object*

Description

h_ahead takes a Forecast object and gets the h_ahead vector of the forecast.

Usage

```
## S4 method for signature 'Forecast'
h_ahead(Forecast)
```

Arguments

Forecast	Forecast object.

Value

Vector of h_ahead values stored in the Forecast object.

Examples

```
## Not run:

h_ahead(Forecast)


## End(Not run)
```

h_ahead<- *Set h_ahead slot of a Forecast object*

Description

h_ahead takes a Forecast object and sets the h_ahead vector of the forecast.

Usage

```
h_ahead(Forecast) <- value
```

Arguments

Forecast	Forecast object.
value	Vector of values assigned to the h_ahead slot of the Forecast.

Value

Forecast object that contains the new h_ahead vector.

Examples

```
## Not run:

h_ahead(Forecast) <- 4L


## End(Not run)
```

---

h_ahead<-,Forecast-method

Set h_ahead slot of a Forecast object

Description

h_ahead takes a Forecast object and sets the h_ahead vector of the forecast.

Usage

```
## S4 replacement method for signature 'Forecast'
h_ahead(Forecast) <- value
```

Arguments

Forecast	Forecast object.
value	Vector of values assigned to the h_ahead slot of the Forecast.

Value

Forecast object that contains the new h_ahead vector.

Examples

```
## Not run:

h_ahead(Forecast) <- 4L


## End(Not run)
```

`is_forc` *In-sample linear model forecast*

Description

`is_forc` takes a linear model call and an optional vector of time data associated with the linear model. The linear model is estimated once over the entire sample period and the coefficients are multiplied by the realized values in each period of the sample. Returns an in-sample forecast conditional on realized values.

Usage

```
is_forc(lm_call, time_vec = NULL)
```

Arguments

`lm_call`	Linear model call of the class lm.
`time_vec`	Vector of any class that is equal in length to the data in `lm_call`.

Value

`Forecast` object that contains the in-sample forecast.

See Also

For a detailed example see the help vignette: `vignette("lmForc",package = "lmForc")`

Examples

```
date <- as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31",
                  "2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31",
                  "2012-03-31", "2012-06-30"))
y  <- c(1.09, 1.71, 1.09, 2.46, 1.78, 1.35, 2.89, 2.11, 2.97, 0.99)
x1 <- c(4.22, 3.86, 4.27, 5.60, 5.11, 4.31, 4.92, 5.80, 6.30, 4.17)
x2  <- c(10.03, 10.49, 10.85, 10.47, 9.09, 10.91, 8.68, 9.91, 7.87, 6.63)
data <- data.frame(date, y, x1, x2)

is_forc(
  lm_call = lm(y ~ x1 + x2, data),
  time_vec = data$date
)

is_forc(
  lm_call = lm(y ~ x1 + x2, data)
)
```

mae *Calculate MAE of a Forecast object*

Description

mae takes a Forecast object and returns the MAE of the forecast. MAE is calculated as: 1/length(forecast) * sum(abs(forecast -realized))

Usage

```
mae(Forecast)
```

Arguments

Forecast	Forecast object.

Value

MAE value.

Examples

```
my_forecast <- Forecast(
  origin   = as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31")),
  future   = as.Date(c("2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31")),
  forecast = c(4.21, 4.27, 5.32, 5.11),
  realized = c(4.40, 4.45, 4.87, 4.77),
  h_ahead  = 4L
)

mae(my_forecast)
```

mae,Forecast-method *Calculate MAE of a Forecast object*

Description

mae takes a Forecast object and returns the MAE of the forecast. MAE is calculated as: 1/length(forecast) * sum(abs(forecast -realized))

Usage

```
## S4 method for signature 'Forecast'
mae(Forecast)
```

Arguments

Forecast	Forecast object.

Value

MAE value.

Examples

```
my_forecast <- Forecast(
  origin   = as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31")),
  future   = as.Date(c("2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31")),
  forecast = c(4.21, 4.27, 5.32, 5.11),
  realized = c(4.40, 4.45, 4.87, 4.77),
  h_ahead  = 4L
)

mae(my_forecast)
```

---

mape *Calculate MAPE of a Forecast object*

Description

mape takes a Forecast object and returns the MAPE of the forecast. MAPE is calculated as: 1/length(forecast) * sum(abs(realized -forecast) / realized)

Usage

```
mape(Forecast)
```

Arguments

Forecast	Forecast object.

Value

MAPE value.

Examples

```
my_forecast <- Forecast(
  origin   = as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31")),
  future   = as.Date(c("2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31")),
  forecast = c(4.21, 4.27, 5.32, 5.11),
  realized = c(4.40, 4.45, 4.87, 4.77),
```

```
  h_ahead  = 4L
)

mape(my_forecast)
```

`mape,Forecast-method` *Calculate MAPE of a Forecast object*

Description

mape takes a Forecast object and returns the MAPE of the forecast. MAPE is calculated as: `1/length(forecast) * sum(abs(realized - forecast) / realized)`

Usage

```
## S4 method for signature 'Forecast'
mape(Forecast)
```

Arguments

Forecast	Forecast object.

Value

MAPE value.

Examples

```
my_forecast <- Forecast(
  origin   = as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31")),
  future   = as.Date(c("2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31")),
  forecast = c(4.21, 4.27, 5.32, 5.11),
  realized = c(4.40, 4.45, 4.87, 4.77),
  h_ahead  = 4L
)

mape(my_forecast)
```

mse *Calculate MSE of a Forecast object*

Description

mse takes a Forecast object and returns the MSE of the forecast. MSE is calculated as: 1/length(forecast) * sum((realized -forecast)^2)

Usage

```
mse(Forecast)
```

Arguments

Forecast	Forecast object.

Value

MSE value.

Examples

```
my_forecast <- Forecast(
  origin   = as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31")),
  future   = as.Date(c("2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31")),
  forecast = c(4.21, 4.27, 5.32, 5.11),
  realized = c(4.40, 4.45, 4.87, 4.77),
  h_ahead  = 4L
)

mse(my_forecast)
```

mse,Forecast-method *Calculate MSE of a Forecast object*

Description

mse takes a Forecast object and returns the MSE of the forecast. MSE is calculated as: 1/length(forecast) * sum((realized -forecast)^2)

Usage

```
## S4 method for signature 'Forecast'
mse(Forecast)
```

Arguments

Forecast	Forecast object.

Value

MSE value.

Examples

```
my_forecast <- Forecast(
  origin   = as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31")),
  future   = as.Date(c("2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31")),
  forecast = c(4.21, 4.27, 5.32, 5.11),
  realized = c(4.40, 4.45, 4.87, 4.77),
  h_ahead  = 4L
)

mse(my_forecast)
```

---

oos_lag_forc *Out-of-sample lagged linear model forecast conditioned on realized values*

Description

oos_lag_forc takes a linear model call, an integer number of periods ahead to forecast, a period to end the initial coefficient estimation and begin forecasting, an optional vector of time data associated with the linear model, and an optional integer number of past periods to estimate the linear model over. Linear model data is lagged by h_ahead periods and the linear model is re-estimated with data up to estimation_end minus the number of periods specified in estimation_window to create a lagged linear model. If estimation_window is left NULL then the linear model is estimated with all available data up to estimation_end. Coefficients are multiplied by present period realized values of the covariates to create a forecast for h_ahead periods ahead. This process is iteratively repeated for each period after estimation_end with coefficients updating in each period. Returns an out-of-sample forecast conditional on realized values that **would** have been available at the forecast origin. Optionally returns the coefficients used to create each forecast. Tests the out-of-sample performance of a linear model had it been lagged and conditioned on available information.

Usage

```
oos_lag_forc(
  lm_call,
  h_ahead,
  estimation_end,
  time_vec = NULL,
  estimation_window = NULL,
```

```
  return_betas = FALSE
)
```

Arguments

lm_call	Linear model call of the class lm.
h_ahead	Integer representing the number of periods ahead that is being forecasted.
estimation_end	Value of any class representing when to end the initial coefficient estimation period and begin forecasting.
time_vec	Vector of any class that is equal in length to the data in lm_call.
estimation_window	Integer representing the number of past periods that the linear model should be estimated over in each period.
return_betas	Boolean, selects whether the coefficients used in each period to create the forecast are returned. If TRUE, a data frame of betas is returned to the Global Environment.

Value

Forecast object that contains the out-of-sample forecast.

See Also

For a detailed example see the help vignette: `vignette("lmForc",package = "lmForc")`

Examples

```
date <- as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31",
                  "2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31",
                  "2012-03-31", "2012-06-30"))
y  <- c(1.09, 1.71, 1.09, 2.46, 1.78, 1.35, 2.89, 2.11, 2.97, 0.99)
x1 <- c(4.22, 3.86, 4.27, 5.60, 5.11, 4.31, 4.92, 5.80, 6.30, 4.17)
x2  <- c(10.03, 10.49, 10.85, 10.47, 9.09, 10.91, 8.68, 9.91, 7.87, 6.63)
data <- data.frame(date, y, x1, x2)

oos_lag_forc(
  lm_call = lm(y ~ x1 + x2, data),
  h_ahead = 2L,
  estimation_end = as.Date("2011-03-31"),
  time_vec = data$date,
  estimation_window = NULL,
  return_betas = FALSE
)

oos_lag_forc(
  lm_call = lm(y ~ x1 + x2, data),
  h_ahead = 2L,
  estimation_end = 6L
)
```

`oos_realized_forc` *Out-of-sample linear model forecast conditioned on realized values*

Description

`oos_realized_forc` takes a linear model call, an integer number of periods ahead to forecast, a period to end the initial coefficient estimation and begin forecasting, an optional vector of time data associated with the linear model, and an optional integer number of past periods to estimate the linear model over. The linear model is originally estimated with data up to `estimation_end` minus the number of periods specified in `estimation_window`. If `estimation_window` is left `NULL` then the linear model is estimated with all available data up to `estimation_end`. Coefficients are multiplied by realized values of the covariates `h_ahead` periods ahead to create an `h_ahead` period ahead forecast. This process is iteratively repeated for each period after `estimation_end` with coefficients updating in each period. Returns an out-of-sample forecast conditional on realized values that **would not** have been available at the forecast origin. Optionally returns the coefficients used to create each forecast. Tests the out-of-sample performance of a linear model had it been conditioned on perfect information.

Usage

```
oos_realized_forc(
  lm_call,
  h_ahead,
  estimation_end,
  time_vec = NULL,
  estimation_window = NULL,
  return_betas = FALSE
)
```

Arguments

`lm_call`	Linear model call of the class lm.
`h_ahead`	Integer representing the number of periods ahead that is being forecasted.
`estimation_end`	Value of any class representing when to end the initial coefficient estimation period and begin forecasting.
`time_vec`	Vector of any class that is equal in length to the data in `lm_call`.
`estimation_window`	Integer representing the number of past periods that the linear model should be estimated over in each period.
`return_betas`	Boolean, selects whether the coefficients used in each period to create the forecast are returned. If TRUE, a data frame of betas is returned to the Global Environment.

Value

`Forecast` object that contains the out-of-sample forecast.

See Also

For a detailed example see the help vignette: `vignette("lmForc",package = "lmForc")`

Examples

```
date <- as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31",
                  "2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31",
                  "2012-03-31", "2012-06-30"))
y  <- c(1.09, 1.71, 1.09, 2.46, 1.78, 1.35, 2.89, 2.11, 2.97, 0.99)
x1 <- c(4.22, 3.86, 4.27, 5.60, 5.11, 4.31, 4.92, 5.80, 6.30, 4.17)
x2  <- c(10.03, 10.49, 10.85, 10.47, 9.09, 10.91, 8.68, 9.91, 7.87, 6.63)
data <- data.frame(date, y, x1, x2)

oos_realized_forc(
  lm_call = lm(y ~ x1 + x2, data),
  h_ahead = 2L,
  estimation_end = as.Date("2011-03-31"),
  time_vec = data$date,
  estimation_window = NULL,
  return_betas = FALSE
)
```

`oos_vintage_forc` *Out-of-sample linear model forecast conditioned on vintage forecasts*

Description

`oos_vintage_forc` takes a linear model call, a vector of time data associated with the linear model, a forecast for each covariate in the linear model, and an optional integer number of past periods to estimate the linear model over. For each period in the vintage forecasts, coefficients are estimated with data up to the current period minus the number of periods specified in `estimation_window`. If `estimation_window` is left `NULL` then the linear model is estimated with all available data up to the current period. Coefficients are then multiplied by vintage forecast values. Returns an out-of-sample forecast conditional on vintage forecasts that **would** have been available at the forecast origin. Optionally returns the coefficients used to create each forecast. Replicates the forecasts that a linear model would have produced in real time.

Usage

```
oos_vintage_forc(
  lm_call,
  time_vec,
  ...,
  estimation_window = NULL,
  return_betas = FALSE
)
```

Arguments

lm_call	Linear model call of the class lm.
time_vec	Vector of any class that is equal in length to the data in lm_call.
...	Set of forecasts of class Forecast, one forecast for each covariate in the linear model.
estimation_window	Integer representing the number of past periods that the linear model should be estimated over in each period.
return_betas	Boolean, selects whether the coefficients used in each period to create the forecast are returned. If TRUE, a data frame of betas is returned to the Global Environment.

Value

Forecast object that contains the out-of-sample forecast.

See Also

For a detailed example see the help vignette: vignette("lmForc",package = "lmForc")

Examples

```
date <- as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31",
                  "2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31",
                  "2012-03-31", "2012-06-30"))
y  <- c(1.09, 1.71, 1.09, 2.46, 1.78, 1.35, 2.89, 2.11, 2.97, 0.99)
x1 <- c(4.22, 3.86, 4.27, 5.60, 5.11, 4.31, 4.92, 5.80, 6.30, 4.17)
x2  <- c(10.03, 10.49, 10.85, 10.47, 9.09, 10.91, 8.68, 9.91, 7.87, 6.63)
data <- data.frame(date, y, x1, x2)

x1_forecast_vintage <- Forecast(
   origin   = as.Date(c("2010-09-30", "2010-12-31", "2011-03-31", "2011-06-30")),
   future   = as.Date(c("2011-09-30", "2011-12-31", "2012-03-31", "2012-06-30")),
   forecast = c(6.30, 4.17, 5.30, 4.84),
   realized = c(4.92, 5.80, 6.30, 4.17),
   h_ahead  = 4L
)

x2_forecast_vintage <- Forecast(
   origin   = as.Date(c("2010-09-30", "2010-12-31", "2011-03-31", "2011-06-30")),
   future   = as.Date(c("2011-09-30", "2011-12-31", "2012-03-31", "2012-06-30")),
 forecast = c(7.32, 6.88, 6.82, 6.95),
 realized = c(8.68, 9.91, 7.87, 6.63),
 h_ahead  = 4L
)

oos_vintage_forc(
  lm_call = lm(y ~ x1 + x2, data),
  time_vec = data$date,
  x1_forecast_vintage, x2_forecast_vintage,
```

```
  estimation_window = 4L,
  return_betas = FALSE
)

oos_vintage_forc(
  lm_call = lm(y ~ x1 + x2, data),
  time_vec = data$date,
  x1_forecast_vintage, x2_forecast_vintage
)
```

`origin` *Get the origin slot of a Forecast object*

Description

`origin` takes a `Forecast` object and gets the origin vector of the forecast.

Usage

```
origin(Forecast)
```

Arguments

`Forecast`	Forecast object.

Value

Vector of origin values stored in the Forecast object.

Examples

```
## Not run:

origin(Forecast)


## End(Not run)
```

origin,Forecast-method *Get the origin slot of a Forecast object*

Description

`origin` takes a `Forecast` object and gets the origin vector of the forecast.

Usage

```
## S4 method for signature 'Forecast'
origin(Forecast)
```

Arguments

`Forecast`	Forecast object.

Value

Vector of origin values stored in the Forecast object.

Examples

```
## Not run:

origin(Forecast)


## End(Not run)
```

origin<- *Set the origin slot of a Forecast object*

Description

`origin` takes a `Forecast` object and sets the origin vector of the forecast.

Usage

```
origin(Forecast) <- value
```

Arguments

`Forecast`	Forecast object.
`value`	Vector of values assigned to the origin slot of the Forecast.

Value

Forecast object that contains the new origin vector.

Examples

```
## Not run:

origin(Forecast) <- c("2015-01-01", "2015-01-02", "2015-01-03")


## End(Not run)
```

`origin<-,Forecast-method`

Set origin slot of a Forecast object

Description

`origin` takes a Forecast object and sets the origin vector of the forecast.

Usage

```
## S4 replacement method for signature 'Forecast'
origin(Forecast) <- value
```

Arguments

`Forecast`	Forecast object.
`value`	Vector of values assigned to the origin slot of the Forecast.

Value

Forecast object that contains the new origin vector.

Examples

```
## Not run:

origin(Forecast) <- c("2015-01-01", "2015-01-02", "2015-01-03")


## End(Not run)
```

`performance_weighted_forc` *MSE or RMSE weighted forecast*

Description

`performance_weighted_forc` takes two or more forecasts, an evaluation window, and an error function. For each forecast period, the error function is used to calculate forecast accuracy over the past `eval_window` number of periods. The forecast accuracy of each forecast is used to weight forecasts based on performance. Returns a weighted forecast. Optionally returns the set of weights used to weight forecasts in each period.

Usage

```
performance_weighted_forc(
  ...,
  eval_window,
  errors = "mse",
  return_weights = FALSE
)
```

Arguments

`...`	Two or more forecasts of class Forecast.
`eval_window`	Integer representing the window over which forecast accuracy is evaluated. Forecasts are weighted based on their accuracy over the past `eval_window` number of periods.
`errors`	Character, either "mse" or "rmse". Selects whether forecast accuracy is evaluated using mean squared errors or root mean squared errors.
`return_weights`	Boolean, selects whether the weights used to weight forecasts in each period are returned. If TRUE, a data frame of weights is returned to the Global Environment.

Details

Forecasts are weighted in each period with the following function. The error function used is MSE or RMSE depending on user selection. This example shows MSE errors.

$$weight = (1/MSE(forecast))/(1/sum(MSE(forecasts)))$$

Value

`Forecast` object that contains the weighted forecast.

See Also

For a detailed example see the help vignette: `vignette("lmForc",package = "lmForc")`

Examples

```
y1_forecast <- Forecast(
  origin = as.Date(c("2009-03-31", "2009-06-30", "2009-09-30", "2009-12-31",
                     "2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31",
                     "2011-03-31", "2011-06-30")),
  future = as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31",
                     "2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31",
                     "2012-03-31", "2012-06-30")),
  forecast = c(1.33, 1.36, 1.38, 1.68, 1.60, 1.55, 1.32, 1.22, 1.08, 0.88),
  realized = c(1.09, 1.71, 1.09, 2.46, 1.78, 1.35, 2.89, 2.11, 2.97, 0.99),
  h_ahead = 4L
)

y2_forecast <- Forecast(
  origin = as.Date(c("2009-03-31", "2009-06-30", "2009-09-30", "2009-12-31",
                     "2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31",
                     "2011-03-31", "2011-06-30")),
  future = as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31",
                     "2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31",
                     "2012-03-31", "2012-06-30")),
  forecast = c(0.70, 0.88, 1.03, 1.05, 1.01, 0.82, 0.95, 1.09, 1.07, 1.06),
  realized = c(1.09, 1.71, 1.09, 2.46, 1.78, 1.35, 2.89, 2.11, 2.97, 0.99),
  h_ahead = 4L
)

performance_weighted_forc(
  y1_forecast, y2_forecast,
  eval_window = 2L,
  errors = "mse",
  return_weights = FALSE
)
```

R2 *Calculate R2 of a Forecast object*

Description

R2 takes a Forecast object and returns the R2 of the forecast. R2 is calculated as: `cor(forecast,realized)^2`

Usage

```
R2(Forecast)
```

Arguments

Forecast	Forecast object.

Value

R2 value.

Examples

```
my_forecast <- Forecast(
  origin   = as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31")),
  future   = as.Date(c("2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31")),
  forecast = c(4.21, 4.27, 5.32, 5.11),
  realized = c(4.40, 4.45, 4.87, 4.77),
  h_ahead  = 4L
)

R2(my_forecast)
```

R2,Forecast-method *Calculate R2 of a Forecast object*

Description

R2 takes a Forecast object and returns the R2 of the forecast. R2 is calculated as: `cor(forecast,realized)^2`

Usage

```
## S4 method for signature 'Forecast'
R2(Forecast)
```

Arguments

Forecast	Forecast object.

Value

R2 value.

Examples

```
my_forecast <- Forecast(
  origin   = as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31")),
  future   = as.Date(c("2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31")),
  forecast = c(4.21, 4.27, 5.32, 5.11),
  realized = c(4.40, 4.45, 4.87, 4.77),
  h_ahead  = 4L
)

R2(my_forecast)
```

`random_walk_forc` *Random walk forecast*

Description

`random_walk_forc` takes a vector of realized values, an integer number of periods ahead to forecast, and an optional vector of time data associated with the realized values. In each period, the current period value of the `realized_vec` series is set as the `h_ahead` period ahead forecast. Returns a random walk forecast where the `h_ahead` period ahead forecast is simply the present value of the series being forecasted.

Usage

```
random_walk_forc(realized_vec, h_ahead, time_vec = NULL)
```

Arguments

`realized_vec`	Vector of realized values. This is the series that is being forecasted.
`h_ahead`	Integer representing the number of periods ahead that is being forecasted.
`time_vec`	Vector of any class that is equal in length to the `realized_vec` vector.

Value

`Forecast` object that contains the random walk forecast.

See Also

For a detailed example see the help vignette: `vignette("lmForc",package = "lmForc")`

Examples

```
date <- as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31",
                  "2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31",
                  "2012-03-31", "2012-06-30"))
y  <- c(1.09, 1.71, 1.09, 2.46, 1.78, 1.35, 2.89, 2.11, 2.97, 0.99)
data <- data.frame(date, y)

random_walk_forc(
  realized_vec = data$y,
  h_ahead = 4L,
  time_vec = data$date
)
```

realized *Get the realized slot of a realized object*

Description

realized takes a Forecast object and gets the realized vector of the forecast.

Usage

```
realized(Forecast)
```

Arguments

Forecast	Forecast object.

Value

Vector of realized values stored in the Forecast object.

Examples

```
## Not run:

realized(Forecast)


## End(Not run)
```

realized,Forecast-method *Get the realized slot of a realized object*

Description

realized takes a Forecast object and gets the realized vector of the forecast.

Usage

```
## S4 method for signature 'Forecast'
realized(Forecast)
```

Arguments

Forecast	Forecast object.

Value

Vector of realized values stored in the Forecast object.

Examples

```
## Not run:

realized(Forecast)


## End(Not run)
```

`realized<-` *Set realized slot of a Forecast object*

Description

`realized` takes a `Forecast` object and sets the realized vector of the forecast.

Usage

```
realized(Forecast) <- value
```

Arguments

`Forecast`	Forecast object.
`value`	Vector of values assigned to the realized slot of the Forecast.

Value

`Forecast` object that contains the new realized vector.

Examples

```
## Not run:

realized(Forecast) <- c("2015-03-01", "2015-03-02", "2015-03-03")


## End(Not run)
```

realized<-,Forecast-method *Set realized slot of a Forecast object*

Description

`realized` takes a `Forecast` object and sets the realized vector of the forecast.

Usage

```
## S4 replacement method for signature 'Forecast'
realized(Forecast) <- value
```

Arguments

`Forecast`	Forecast object.
`value`	Vector of values assigned to the realized slot of the Forecast.

Value

`Forecast` object that contains the new realized vector.

Examples

```
## Not run:

realized(Forecast) <- c("2015-03-01", "2015-03-02", "2015-03-03")


## End(Not run)
```

rmse *Calculate RMSE of a Forecast object*

Description

`rmse` takes a `Forecast` object and returns the RMSE of the forecast. RMSE is calculated as: `sqrt(mse)`

Usage

```
rmse(Forecast)
```

Arguments

`Forecast`	Forecast object.

Value

RMSE value.

Examples

```
my_forecast <- Forecast(
  origin   = as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31")),
  future   = as.Date(c("2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31")),
  forecast = c(4.21, 4.27, 5.32, 5.11),
  realized = c(4.40, 4.45, 4.87, 4.77),
  h_ahead  = 4L
)

rmse(my_forecast)
```

---

rmse,Forecast-method *Calculate RMSE of a Forecast object*

Description

rmse takes a Forecast object and returns the RMSE of the forecast. RMSE is calculated as: sqrt(mse)

Usage

```
## S4 method for signature 'Forecast'
rmse(Forecast)
```

Arguments

Forecast	Forecast object.

Value

RMSE value.

Examples

```
my_forecast <- Forecast(
  origin   = as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31")),
  future   = as.Date(c("2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31")),
  forecast = c(4.21, 4.27, 5.32, 5.11),
  realized = c(4.40, 4.45, 4.87, 4.77),
  h_ahead  = 4L
)
```

```
rmse(my_forecast)
```

`show,Forecast-method` *Print Forecast object to console.*

Description

show takes a `Forecast` object and prints it to console.

Usage

```
## S4 method for signature 'Forecast'
show(object)
```

Arguments

`object`	Forecast object.

Value

Printed `Forecast` object.

Examples

```
my_forecast <- Forecast(
  origin   = as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31")),
  future   = as.Date(c("2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31")),
  forecast = c(4.21, 4.27, 5.32, 5.11),
  realized = c(4.40, 4.45, 4.87, 4.77),
  h_ahead  = 4L
)

print(my_forecast)
```

`states_weighted_forc` *States weighted forecast*

Description

`states_weighted_forc` takes two or more forecasts, a data frame, matrix, or array of matching variables, an optional vector of time data associated with the matching variables, a matching window size, a matching function, and an error function. For each forecast period, `matching_vars` are standardized and the current state of the world is set as the the past `matching_window` periods of the matching variables. The current state is compared to all past periods of the matching variables using the matching function. The current state is matched to the past state that minimizes the matching function. The forecast error function is then used to compute the accuracy of each forecast over the matched past state. Forecast weights are computed based on this forecast accuracy, and the current period forecast is subsequently computed based on the forecast weights. Produces a weighted average of multiple forecasts based on how each forecast performed during the past state that is most similar to the current state of the world.

Usage

```
states_weighted_forc(
  ...,
  matching_vars,
  time_vec = NULL,
  matching_window,
  matching = "euclidean",
  errors = "mse",
  return_weights = FALSE
)
```

Arguments

`...`	Two or more forecasts of class Forecast.
`matching_vars`	data frame, array, or matrix of variables used to match the current state of the world to a past state.
`time_vec`	Vector of any class that is equal in length to the data in `matching_vars`.
`matching_window`	Integer representing the window size over which the current state of the world is matched to a past state. Forecasts are also weighted based on their accuracy over `matching_window` periods.
`matching`	Character, "euclidean", "mse", or "rmse". Selects the function used to match the current state of the world to a past state.
`errors`	Character, either "mse" or "rmse". Selects whether forecast accuracy is evaluated using mean squared errors or root mean squared errors.
`return_weights`	Boolean, selects whether the weights used to weight forecasts in each period are returned. If TRUE, a data frame of weights and matched periods is returned to the Global Environment.

Details

Forecasts are weighted in each period with the function below. The error function used is MSE or RMSE depending on user selection. This example shows MSE errors.

$$weight = (1/MSE(forecast))/(1/sum(MSE(forecasts)))$$

Value

`Forecast` object that contains the state weighted forecast.

See Also

For a detailed example see the help vignette: `vignette("lmForc",package = "lmForc")`

Examples

```
date <- as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31",
                  "2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31",
                  "2012-03-31", "2012-06-30"))

future <- as.Date(c("2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31",
                    "2012-03-31", "2012-06-30", "2012-09-30", "2012-12-31",
                    "2013-03-31", "2013-06-30"))

y  <- c(1.09, 1.71, 1.09, 2.46, 1.78, 1.35, 2.89, 2.11, 2.97, 0.99)
x1 <- c(4.22, 3.86, 4.27, 5.60, 5.11, 4.31, 4.92, 5.80, 6.30, 4.17)
x2 <- c(10.03, 10.49, 10.85, 10.47, 9.09, 10.91, 8.68, 9.91, 7.87, 6.63)

data <- data.frame(date, y, x1, x2)
matching_vars <- data[, c("x1", "x2")]

y1_forecast <- Forecast(
  origin = date,
  future = future,
  forecast = c(1.33, 1.36, 1.38, 1.68, 1.60, 1.55, 1.32, 1.22, 1.08, 0.88),
  realized = c(1.78, 1.35, 2.89, 2.11, 2.97, 0.99, 1.31, 1.41, 1.02, 1.05),
  h_ahead = 4L
)

y2_forecast <- Forecast(
  origin = date,
  future = future,
  forecast = c(0.70, 0.88, 1.03, 1.05, 1.01, 0.82, 0.95, 1.09, 1.07, 1.06),
  realized = c(1.78, 1.35, 2.89, 2.11, 2.97, 0.99, 1.31, 1.41, 1.02, 1.05),
  h_ahead = 4L
)

states_weighted_forc(
  y1_forecast, y2_forecast,
  matching_vars = matching_vars,
  time_vec = data$date,
```

```
  matching_window = 2L,
  matching = "euclidean",
  errors = "mse",
  return_weights = FALSE
)

states_weighted_forc(
  y1_forecast, y2_forecast,
  matching_vars = matching_vars,
  time_vec = data$date,
  matching_window = 3L,
  matching = "rmse",
  errors = "rmse"
)
```

`str,Forecast-method` *Display internal structure structure of Forecast object to the console.*

Description

`str` takes a `Forecast` object and prints its internal structure to the console.

Usage

```
## S4 method for signature 'Forecast'
str(object)
```

Arguments

`object`	Forecast object.

Value

Structure of `Forecast` object.

Examples

```
my_forecast <- Forecast(
  origin   = as.Date(c("2010-03-31", "2010-06-30", "2010-09-30", "2010-12-31")),
  future   = as.Date(c("2011-03-31", "2011-06-30", "2011-09-30", "2011-12-31")),
  forecast = c(4.21, 4.27, 5.32, 5.11),
  realized = c(4.40, 4.45, 4.87, 4.77),
  h_ahead  = 4L
)

str(my_forecast)
```

`[,Forecast-method` *Subset Forecast object.*

Description

`[]` takes a `Forecast` object and subsets it.

Usage

```
## S4 method for signature 'Forecast'
x[i, j, ..., drop = TRUE]
```

Arguments

`x`	ANY
`i`	ANY
`j`	ANY
`...`	ANY
`drop`	ANY
`Forecast`	Forecast object.

Value

Subsetted `Forecast` object.

Index