Package: ahead (via r-universe)

September 16, 2024

Type Package

Title Time Series Forecasting with uncertainty quantification

Version 0.13.0

Date 2024-09-16

Author T. Moudiki

Maintainer T. Moudiki <thierry.moudiki@gmail.com>

Description Univariate and multivariate time series forecasting with uncertainty quantification.

License BSD_3_clause Clear + file LICENSE

Imports Rcpp (>= 1.0.6)

LinkingTo Rcpp

RoxygenNote 7.3.0

Encoding UTF-8

Depends R (>= 3.5.0), foreach, snow

Suggests dfoptim, doSNOW, knitr, rmarkdown, testthat, forecast, fpp, e1071, randomForest

VignetteBuilder knitr

Repository https://techtonique.r-universe.dev

RemoteUrl https://github.com/Techtonique/ahead

RemoteRef HEAD

RemoteSha 9a117aa57e3e4149a9724727da602d44aad36ffc

Contents

`armagarchf` *ARMA(1, 1)-GARCH(1, 1) forecasting (with simulation)*

Description

ARMA(1, 1)-GARCH(1, 1) forecasting (with simulation)

Usage

```
armagarchf(
  y,
  h = 5,
  level = 95,
  B = 250,
  cl = 1L,
  dist = c("student", "gaussian"),
  seed = 123
)
```

Arguments

`y`	a univariate time series
`h`	number of periods for forecasting
`level`	confidence level for prediction intervals
`B`	number of simulations for ‘arima.sim‘
`cl`	an integer; the number of clusters for parallel execution
`dist`	distribution of innovations ("student" or "gaussian")
`seed`	reproducibility seed

Value

An object of class "forecast"; a list containing the following elements:

`model`	A list containing information about the fitted model
`method`	The name of the forecasting method as a character string
`mean`	Point forecasts for the time series

lower	Lower bound for prediction interval
upper	Upper bound for prediction interval
x	The original time series
sims	Simulations of ARMA(1, 1)-GARCH(1, 1)

Author(s)

T. Moudiki

Examples

```
y <- datasets::EuStockMarkets[ , "DAX"]
log_returns <- ts(log(y[-1]/y[-length(y)]))

# require(forecast)
# z <- ahead::armagarchf(y=log_returns, h=200)
# plot(z)
```

---

basicf *Basic forecasting (mean, median, random walk)*

Description

Basic forecasting functions for multivariate time series

Usage

```
basicf(
  y,
  h = 5,
  level = 95,
  method = c("mean", "median", "rw"),
  type_pi = c("gaussian", "bootstrap", "blockbootstrap", "movingblockbootstrap"),
  block_length = NULL,
  seed = 1,
  B = 100,
  show_progress = TRUE
)
```

Arguments

y	A multivariate time series of class ts or a matrix
h	Forecasting horizon
level	Confidence level for prediction intervals

method	forecasting method, either "mean", "median", or random walk ("rw")
type_pi	type of prediction interval currently, "gaussian", "bootstrap", "blockbootstrap" or "movingblockbootstrap"
block_length	length of block for (circular) "blockbootstrap" or "movingblockbootstrap"
seed	reproducibility seed for type_pi == 'bootstrap'
B	Number of bootstrap replications for type_pi == 'bootstrap'
show_progress	A boolean; show progress bar for bootstrapping? Default is TRUE.

Value

An object of class "mtsforecast"; a list containing the following elements:

method	The name of the forecasting method as a character string
mean	Point forecasts for the time series
lower	Lower bound for prediction interval
upper	Upper bound for prediction interval
sims	Model simulations for bootstrapping (basic, or block)
x	The original time series
residuals	Residuals from the fitted model
coefficients	Regression coefficients for type_pi == 'gaussian' for now

Examples

```
require(fpp)

res <- ahead::basicf(fpp::insurance, h=10)
par(mfrow=c(1, 2))
plot(res, "TV.advert")
plot(res, "Quotes")


res <- ahead::basicf(fpp::insurance, method="rw", h=10)
par(mfrow=c(1, 2))
plot(res, "TV.advert")
plot(res, "Quotes")


# block bootstrap
res3 <- ahead::basicf(fpp::insurance, h=10, type_pi = "bootstrap", B=10)
res5 <- ahead::basicf(fpp::insurance, h=10, type_pi = "blockbootstrap", B=10,
                      block_length = 4)

print(res3$sims[[2]])
print(res5$sims[[2]])

par(mfrow=c(2, 2))
plot(res3, "Quotes")
```

```
plot(res3, "TV.advert")
plot(res5, "Quotes")
plot(res5, "TV.advert")


# moving block bootstrap
res6 <- ahead::basicf(fpp::insurance, h=10,
                      type_pi = "movingblockbootstrap", B=10,
                      block_length = 4, method = "rw")

par(mfrow=c(1, 2))
plot(res6, "Quotes")
plot(res6, "TV.advert")
```

---

createtrendseason *Create trend and seasonality features for univariate time series*

Description

Create trend and seasonality features for univariate time series

Usage

```
createtrendseason(y)
```

Arguments

y	a univariate time series

Value

a vector or matrix of features

Examples

```
y <- ts(rnorm(100), start = 1, frequency = 12)
createtrendseason(y)

createtrendseason(USAccDeaths)
```

dynrmf *Dynamic regression model*

Description

Adapted from forecast::nnetar, with alternative fitting functions (see examples)

Usage

```
dynrmf(
  y,
  h = 5,
  level = 95,
  fit_func = ahead::ridge,
  predict_func = predict,
  fit_params = NULL,
  type_pi = c("gaussian", "E", "A", "T"),
  xreg_fit = NULL,
  xreg_predict = NULL,
  ...
)
```

Arguments

y	A numeric vector or time series of class ts
h	Forecasting horizon
level	Confidence level for prediction intervals
fit_func	Fitting function (Statistical/ML model). Default is Ridge regression.
predict_func	Prediction function (Statistical/ML model)
fit_params	a list of additional parameters for the fitting function fit_func (see examples)
type_pi	Type of prediction interval (currently "gaussian", ETS: "E", Arima: "A" or Theta: "T")
xreg_fit	Optionally, a vector or matrix of external regressors, which must have the same number of rows as y. Must be numeric.
xreg_predict	Future values of external regressor variables.
...	additional parameters

Value

a list; an object of class forecast.

The function summary is used to obtain and print a summary of the results.

The generic accessor functions fitted.values and residuals extract useful features.

Author(s)

T. Moudiki

References

Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice. OTexts.

Hyndman R, Athanasopoulos G, Bergmeir C, Caceres G, Chhay L, O'Hara-Wild M, Petropoulos F, Razbash S, Wang E, Yasmeen F (2021). forecast: Forecasting functions for time series and linear models. R package version 8.14, <URL: https://pkg.robjhyndman.com/forecast/>.

Examples

```
# Example 0: with Ridge regression

par(mfrow=c(3, 2))
plot(dynrmf(USAccDeaths, h=20, level=95))
plot(dynrmf(AirPassengers, h=20, level=95))
plot(dynrmf(lynx, h=20, level=95))
plot(dynrmf(WWWusage, h=20, level=95))
plot(dynrmf(Nile, h=20, level=95))
plot(dynrmf(fdeaths, h=20, level=95))


# Example 1: with Random Forest

## Not run:

require(randomForest)

par(mfrow=c(3, 2))
plot(dynrmf(USAccDeaths, h=20, level=95, fit_func = randomForest::randomForest,
     fit_params = list(ntree = 50), predict_func = predict))
plot(dynrmf(AirPassengers, h=20, level=95, fit_func = randomForest::randomForest,
     fit_params = list(ntree = 50), predict_func = predict))
plot(dynrmf(lynx, h=20, level=95, fit_func = randomForest::randomForest,
     fit_params = list(ntree = 50), predict_func = predict))
plot(dynrmf(WWWusage, h=20, level=95, fit_func = randomForest::randomForest,
     fit_params = list(ntree = 50), predict_func = predict))
plot(dynrmf(Nile, h=20, level=95, fit_func = randomForest::randomForest,
     fit_params = list(ntree = 50), predict_func = predict))
plot(dynrmf(fdeaths, h=20, level=95, fit_func = randomForest::randomForest,
     fit_params = list(ntree = 50), predict_func = predict))

## End(Not run)

# Example 2: with SVM

## Not run:

require(e1071)
```

```
par(mfrow=c(2, 2))
plot(dynrmf(fdeaths, h=20, level=95, fit_func = e1071::svm,
fit_params = list(kernel = "linear"), predict_func = predict))
plot(dynrmf(fdeaths, h=20, level=95, fit_func = e1071::svm,
fit_params = list(kernel = "polynomial"), predict_func = predict))
plot(dynrmf(fdeaths, h=20, level=95, fit_func = e1071::svm,
fit_params = list(kernel = "radial"), predict_func = predict))
plot(dynrmf(fdeaths, h=20, level=95, fit_func = e1071::svm,
fit_params = list(kernel = "sigmoid"), predict_func = predict))


## End(Not run)
```

`eatf` *Combined ets-arima-theta forecasts*

Description

Combined ets, arima, and theta (eat) forecasting (uses `forecast::ets`, `forecast::auto.arima`, `forecast::thetaf`)

Usage

```
eatf(
  y,
  h = 5,
  level = 95,
  method = c("EAT", "E", "A", "T"),
  weights = rep(1/3, 3),
  type_pi = c("gaussian", "E", "A", "T"),
  ...
)
```

Arguments

`y`	a univariate time series
`h`	number of periods for forecasting
`level`	confidence level for prediction intervals
`method`	forecasting method: "E" for `forecast::ets`; "A"for `forecast::auto.arima`; "T" for `forecast::thetaf`; or "EAT" for the combination of the three (default, with `weights`)
`weights`	weights for each method, in method `EAT`. Must add up to 1.
`type_pi`	type of prediction interval: currently ETS: "E", Auto.Arima: "A" or Theta: "T"
`...`	additional parameters to be passed to `forecast::ets`, `forecast::auto.arima`, `forecast::thetaf` and `forecast::forecast`

Details

ensemble forecasts obtained from `forecast::ets`, `forecast::auto.arima` and `forecast::theta` (with weights)

Value

An object of class "forecast"; a list containing the following elements:

`model`	A list containing information about the fitted model
`method`	The name of the forecasting method as a character string
`mean`	Point forecasts for the time series
`lower`	Lower bound for prediction interval
`upper`	Upper bound for prediction interval
`x`	The original time series
`residuals`	Residuals from the fitted model

Author(s)

T. Moudiki

References

Hyndman R, Athanasopoulos G, Bergmeir C, Caceres G, Chhay L, O'Hara-Wild M, Petropoulos F, Razbash S, Wang E, Yasmeen F (2021). forecast: Forecasting functions for time series and linear models. R package version 8.14, <URL: https://pkg.robjhyndman.com/forecast/>.

Hyndman RJ, Khandakar Y (2008). 'Automatic time series forecasting: the forecast package for R.' Journal of Statistical Software, 26 (3), 1-22. <URL: https://www.jstatsoft.org/article/view/v027i03>.

Assimakopoulos, V. and Nikolopoulos, K. (2000). The theta model: a decomposition approach to forecasting. International Journal of Forecasting 16, 521-530.

Hyndman, R.J., and Billah, B. (2003) Unmasking the Theta method. International J. Forecasting, 19, 287-290.

Examples

```
require(forecast)

## Not run:

print(ahead::eatf(WWWusage, method = "EAT",
weights = c(0.5, 0, 0.5)))

print(ahead::eatf(WWWusage, method = "EAT"))


obj <- ahead::eatf(WWWusage, method = "EAT",
weights = c(0, 0.5, 0.5), h=10,
```

```
type_pi = "T")
plot(obj)


obj <- ahead::eatf(WWWusage, method = "EAT",
weights = c(0, 0.5, 0.5), h=10, type_pi="A")
plot(obj)

## End(Not run)


par(mfrow=c(3, 2))
plot(ahead::eatf(USAccDeaths, h=10, level=95))
plot(ahead::eatf(AirPassengers, h=10, level=95, type_pi = "T"))
plot(ahead::eatf(lynx, h=10, level=95, type_pi = "A"))
plot(ahead::eatf(WWWusage, h=10, level=95, type_pi = "E"))
plot(ahead::eatf(Nile, h=10, level=95))
plot(ahead::eatf(fdeaths, h=10, level=95))
```

`fitforecast` *Fit and forecast for benchmarking purposes*

Usage

```
fitforecast(
  y,
  h = NULL,
  pct_train = 0.9,
  pct_calibration = 0.5,
  method = c("thetaf", "arima", "ets", "te", "tbats", "tslm", "dynrmf", "ridge2f",
    "naive", "snaive"),
  level = 95,
  B = 1000L,
  seed = 17223L,
  graph = TRUE,
  conformalize = FALSE,
  type_calibration = c("splitconformal", "cv1", "loocv"),
  gap = 3L,
  agg = c("mean", "median"),
  vol = c("constant", "garch"),
  type_sim = c("kde", "surrogate", "bootstrap"),
  ...
)
```

Arguments

`y`	A univariate time series of class `ts`

h	Forecasting horizon (default is NULL, in that case, pct_train and pct_calibration are used)
pct_train	Percentage of data in the training set, when h is NULL
pct_calibration	Percentage of data in the calibration set for conformal prediction
method	For now "thetaf" (default), "arima", "ets", "tbats", "tslm", "dynrmf" (from ahead), "ridge2f" (from ahead), "naive", "snaive"
level	Confidence level for prediction intervals in
	\itemBNumber of bootstrap replications or number of simulations (yes, 'B' is unfortunate)
	\itemseedReproducibility seed
	\itemgraphPlot or not?
	\itemconformalizeCalibrate or not?
	\itemtype_calibration"splitconformal" (default conformal method), "cv1" (do not use), "loocv" (do not use)
	\itemgaplength of training set for loocv conformal (do not use)
	\itemagg"mean" or "median" (aggregation method) for
	\itemvol"constant" or "garch" (type of volatility modeling for calibrated residuals)
	\itemtype_sim"kde", "surrogate", "bootstrap" (type of simulation for calibrated residuals)
	\item...additional parameters

an object of class 'forecast' with additional information

Fit and forecast for benchmarking purposes

par(mfrow=c(2, 2)) obj1 <- ahead::fitforecast(AirPassengers) obj2 <- ahead::fitforecast(AirPassengers, conformalize = TRUE) plot(AirPassengers) plot(obj1) obj2$plot() obj2$plot("simulations")

`getreturns` *Calculate returns or log-returns for multivariate time series*

Description

Calculate returns or log-returns for multivariate time series

Usage

```
getreturns(x, type = c("basic", "log"))
```

Arguments

x	Multivariate time series
type	Type of return: basic return ("basic") or log-return ("log")

Value

The returns

Examples

```
returns <- getreturns(EuStockMarkets)
log_returns <- getreturns(EuStockMarkets,
                          type = "log")

par(mfrow=c(1, 3))
matplot(EuStockMarkets, type = 'l', main = "Closing Prices of \n European stocks (1991-1998)",
xlab = "time")
matplot(returns, type = 'l', main = "Returns", xlab = "time")
matplot(log_returns, type = 'l', main = "Log-returns", xlab = "time")
```

---

getsimulations *Obtain simulations (when relevant) from a selected time series*

Description

Obtain simulations (when relevant) from a selected time series

Usage

```
getsimulations(obj, selected_series, transpose = FALSE)
```

Arguments

obj	result from ridge2f (multivariate time series forecast with simulations)
selected_series	name of the time series selected
transpose	return a transposed time series

Examples

```
require(fpp)

obj <- ahead::ridge2f(fpp::insurance, h = 7,
                      type_pi = "bootstrap", B = 5)
print(getsimulations(obj, selected_series = "TV.advert"))
print(getsimulations(obj, selected_series = "Quotes"))
print(getsimulations(obj, selected_series = "TV.advert", transpose = TRUE))
print(getsimulations(obj, selected_series = "Quotes", transpose = TRUE))
```

`loessf` *Loess forecasting*

Description

Loess forecasting

Usage

```
loessf(
  y,
  h = 5,
  level = 95,
  span = 0.75,
  degree = 2,
  type_pi = c("bootstrap", "blockbootstrap", "movingblockbootstrap"),
  b = NULL,
  B = 250,
  type_aggregation = c("mean", "median"),
  seed = 123
)
```

Arguments

`y`	A numeric vector or time series of class `ts`
`h`	Forecasting horizon
`level`	Confidence level for prediction intervals
`span`	the parameter which controls the degree of smoothing
`degree`	the degree of the polynomials to be used, normally 1 or 2. (Degree 0 is also allowed, but see `stats::loess`)
`type_pi`	Type of prediction interval currently (independent) "bootstrap", (circular) "block-bootstrap", or "movingblockbootstrap"
`b`	block length for circular block bootstrap
`B`	number of bootstrap replications
`type_aggregation`	Type of aggregation, ONLY for bootstrapping; either "mean" or "median"
`seed`	reproducibility seed

Value

An object of class "forecast"; a list containing the following elements:

`model`	A list containing information about the fitted model
`method`	The name of the forecasting method as a character string

mean	Point forecasts for the time series
lower	Lower bound for prediction interval
upper	Upper bound for prediction interval
x	The original time series
residuals	Residuals from the fitted model
sims	Model simulations for bootstrapping

Examples

```
par(mfrow = c(3, 1))

plot(loessf(Nile, h=20, level=95, B=10))

plot(loessf(Nile, h=20, level=95, B=10,
     type_pi = "blockbootstrap"))

plot(loessf(Nile, h=20, level=95, B=10,
     type_pi = "movingblockbootstrap"))
```

---

loocvridge2f *LOOCV for Ridge2 model*

Description

LOOCV for Random Vector functional link network model with 2 regularization parameters

Usage

```
loocvridge2f(
  y,
  xreg = NULL,
  h = 5,
  level = 95,
  lags = 1,
  nb_hidden = 5,
  nodes_sim = c("sobol", "halton", "unif"),
  activ = c("relu", "sigmoid", "tanh", "leakyrelu", "elu", "linear"),
  a = 0.01,
  lambda_1 = 0.1,
  lambda_2 = 0.1,
  dropout = 0,
  type_forecast = c("recursive", "direct"),
  type_pi = c("gaussian", "bootstrap", "blockbootstrap", "movingblockbootstrap",
    "rvinecopula", "splitconformal"),
  block_length = NULL,
```

```
  margins = c("gaussian", "empirical", "student"),
  seed = 1,
  B = 100L,
  type_aggregation = c("mean", "median"),
  centers = NULL,
  type_clustering = c("kmeans", "hclust"),
  ym = NULL,
  cl = 1L,
  show_progress = TRUE,
  ...
)
```

Arguments

y	A multivariate time series of class ts (preferred) or a matrix
xreg	External regressors. A data.frame (preferred) or a matrix
h	Forecasting horizon
level	Confidence level for prediction intervals
lags	Number of lags
nb_hidden	Number of nodes in hidden layer
nodes_sim	Type of simulation for nodes in the hidden layer
activ	Activation function
a	Hyperparameter for activation function "leakyrelu", "elu"
lambda_1	Regularization parameter for original predictors
lambda_2	Regularization parameter for transformed predictors
dropout	dropout regularization parameter (dropping nodes in hidden layer)
type_forecast	Recursive or direct forecast
type_pi	Type of prediction interval currently "gaussian", "bootstrap", "blockbootstrap", "movingblockbootstrap", "splitconformal" (very experimental right now), "rvinecopula" (with Gaussian margins for now, Student-t coming soon)
block_length	Length of block for circular or moving block bootstrap
margins	Distribution of margins: "gaussian", "empirical", "student" (postponed or never) for type_pi == "rvinecopula"
seed	Reproducibility seed for random stuff
B	Number of bootstrap replications or number of simulations (yes, 'B' is unfortunate)
type_aggregation	Type of aggregation, ONLY for bootstrapping; either "mean" or "median"
centers	Number of clusters for type_clustering
type_clustering	"kmeans" (K-Means clustering) or "hclust" (Hierarchical clustering)
ym	Univariate time series (stats::ts) of yield to maturities with frequency = frequency(y) and start = tsp(y)[2] + 1 / frequency(y). Default is NULL.

cl	An integer; the number of clusters for parallel execution, for bootstrap
show_progress	A boolean; show progress bar for bootstrapping? Default is TRUE.
...	Additional parameters to be passed to kmeans or hclust

Value

An object of class "mtsforecast"; a list containing the following elements:

method	The name of the forecasting method as a character string
mean	Point forecasts for the time series
lower	Lower bound for prediction interval
upper	Upper bound for prediction interval
sims	Model simulations for bootstrapping (basic, or block)
x	The original time series
residuals	Residuals from the fitted model
coefficients	Regression coefficients for type_pi == 'gaussian' for now

Author(s)

T. Moudiki

References

Moudiki, T., Planchet, F., & Cousin, A. (2018). Multiple time series forecasting using quasi-randomized functional link neural networks. Risks, 6(1), 22.

Examples

```
require(fpp)

print(ahead::loocvridge2f(fpp::insurance))
print(ahead::loocvridge2f(fpp::usconsumption))

#foo <- function(xx) ahead::loocvridge2f(fpp::insurance, lambda_1=10^xx[1], lambda_2=10^xx[2])
#(opt <- stats::nlminb(objective=foo, lower=c(-10,-10), upper=c(10,10), start=c(0, 0)))
#print(ahead::loocvridge2f(fpp::insurance, lambda_1=10^opt$par[1], lambda_2=10^opt$par[2]))
```

`plot.mtsforecast` *Plot multivariate time series forecast or residuals*

Description

Plot multivariate time series forecast or residuals

Usage

```
## S3 method for class 'mtsforecast'
plot(x, selected_series, type = c("pi", "dist", "sims"), level = 95, ...)
```

Arguments

`x`	result from `basicf`, `ridge2f` or `varf` (multivariate time series forecast)
`selected_series`	name of the time series selected for plotting
`type`	"pi": basic prediction intervals; "dist": a distribution of predictions; "sims": the simulations
`level`	confidence levels for prediction intervals
`...`	additional parameters to be passed to `plot` or `matplot`

Examples

```
require(fpp)

fit_obj_VAR <- ahead::varf(fpp::insurance, lags = 2,
h = 10, level = 95)

fit_obj_ridge2 <- ahead::ridge2f(fpp::insurance, lags = 2,
h = 10, level = 95)

par(mfrow=c(2, 2))
plot(fit_obj_VAR, "Quotes")
plot(fit_obj_VAR, "TV.advert")
plot(fit_obj_ridge2, "Quotes")
plot(fit_obj_ridge2, "TV.advert")

obj <- ahead::ridge2f(fpp::insurance, h = 10, type_pi = "blockbootstrap",
block_length=5, B = 10)
par(mfrow=c(1, 2))
plot(obj, selected_series = "Quotes", type = "sims",
main = "Predictive simulation for Quotes")
plot(obj, selected_series = "TV.advert", type = "sims",
main = "Predictive simulation for TV.advert")


par(mfrow=c(1, 2))
```

```
plot(obj, selected_series = "Quotes", type = "dist",
main = "Predictive simulation for Quotes")
plot(obj, selected_series = "TV.advert", type = "dist",
main = "Predictive simulation for TV.advert")
```

`ridge2f` *Ridge2 model*

Description

Random Vector functional link network model with 2 regularization parameters

Usage

```
ridge2f(
  y,
  h = 5,
  level = 95,
  xreg = NULL,
  lags = 1,
  nb_hidden = 5,
  nodes_sim = c("sobol", "halton", "unif"),
  activ = c("relu", "sigmoid", "tanh", "leakyrelu", "elu", "linear"),
  a = 0.01,
  lambda_1 = 0.1,
  lambda_2 = 0.1,
  dropout = 0,
  type_forecast = c("recursive", "direct"),
  type_pi = c("gaussian", "bootstrap", "blockbootstrap", "movingblockbootstrap",
    "rvinecopula"),
  block_length = NULL,
  margins = c("gaussian", "empirical"),
  seed = 1,
  B = 100L,
  type_aggregation = c("mean", "median"),
  centers = NULL,
  type_clustering = c("kmeans", "hclust"),
  ym = NULL,
  cl = 1L,
  show_progress = TRUE,
  ...
)
```

Arguments

y	A univariate of multivariate time series of class ts (preferred) or a matrix
h	Forecasting horizon
level	Confidence level for prediction intervals
xreg	External regressors. A data.frame (preferred) or a matrix
lags	Number of lags
nb_hidden	Number of nodes in hidden layer
nodes_sim	Type of simulation for nodes in the hidden layer
activ	Activation function
a	Hyperparameter for activation function "leakyrelu", "elu"
lambda_1	Regularization parameter for original predictors
lambda_2	Regularization parameter for transformed predictors
dropout	dropout regularization parameter (dropping nodes in hidden layer)
type_forecast	Recursive or direct forecast
type_pi	Type of prediction interval currently "gaussian", "bootstrap", "blockbootstrap", "movingblockbootstrap", "rvinecopula" (with Gaussian and empirical margins for now)
block_length	Length of block for circular or moving block bootstrap
margins	Distribution of margins: "gaussian", "empirical" for type_pi == "rvinecopula"
seed	Reproducibility seed for random stuff
B	Number of bootstrap replications or number of simulations (yes, 'B' is unfortunate)
type_aggregation	Type of aggregation, ONLY for bootstrapping; either "mean" or "median"
centers	Number of clusters for type_clustering
type_clustering	"kmeans" (K-Means clustering) or "hclust" (Hierarchical clustering)
ym	Univariate time series (stats::ts) of yield to maturities with frequency = frequency(y) and start = tsp(y)[2] + 1 / frequency(y). Default is NULL.
cl	An integer; the number of clusters for parallel execution, for bootstrap
show_progress	A boolean; show progress bar for bootstrapping? Default is TRUE.
...	Additional parameters to be passed to kmeans or hclust

Value

An object of class "mtsforecast"; a list containing the following elements:

method	The name of the forecasting method as a character string
mean	Point forecasts for the time series
lower	Lower bound for prediction interval

upper	Upper bound for prediction interval
sims	Model simulations for bootstrapping (basic, or block)
x	The original time series
residuals	Residuals from the fitted model
coefficients	Regression coefficients for type_pi == 'gaussian' for now

Author(s)

T. Moudiki

References

Moudiki, T., Planchet, F., & Cousin, A. (2018). Multiple time series forecasting using quasi-randomized functional link neural networks. Risks, 6(1), 22.

Examples

```
require(fpp)

print(ahead::ridge2f(fpp::insurance)$mean)
print(ahead::ridge2f(fpp::usconsumption)$lower)

res <- ahead::ridge2f(fpp::insurance, h=10, lags=2)
par(mfrow=c(1, 2))
plot(res, "Quotes")
plot(res, "TV.advert")

# include a trend (just for the example)
xreg <- as.numeric(time(fpp::insurance))
res2 <- ahead::ridge2f(fpp::insurance, xreg=xreg,
h=10, lags=2)
par(mfrow=c(1, 2))
plot(res2, "Quotes")
plot(res2, "TV.advert")

# block bootstrap
xreg <- as.numeric(time(fpp::insurance))
res3 <- ahead::ridge2f(fpp::insurance, xreg=xreg,
                       h=10, lags=1L, type_pi = "bootstrap", B=10)
res5 <- ahead::ridge2f(fpp::insurance, xreg=xreg,
                       h=10, lags=1L, type_pi = "blockbootstrap", B=10,
                       block_length = 4)

print(res3$sims[[2]])
print(res5$sims[[2]])

par(mfrow=c(2, 2))
plot(res3, "Quotes")
plot(res3, "TV.advert")
plot(res5, "Quotes")
```

```
plot(res5, "TV.advert")


res4 <- ahead::ridge2f(fpp::usconsumption, h=20, lags=2L,
lambda_2=1)
par(mfrow=c(1, 2))
plot(res4, "income")
plot(res4, "consumption")


# moving block bootstrap
xreg <- as.numeric(time(fpp::insurance))
res6 <- ahead::ridge2f(fpp::insurance, xreg=xreg,
                       h=10, lags=1L,
                       type_pi = "movingblockbootstrap", B=10,
                       block_length = 4)

print(res6$sims[[2]])

par(mfrow=c(1, 2))
plot(res6, "Quotes")
plot(res6, "TV.advert")
```

`varf` *Vector Autoregressive model (adapted from vars::VAR)*

Description

Vector Autoregressive model adapted from vars::VAR (only for benchmarking)

Usage

```
varf(
  y,
  h = 5,
  level = 95,
  lags = 1,
  type_VAR = c("const", "trend", "both", "none"),
  ...
)
```

Arguments

`y`	A multivariate time series of class `ts`
`h`	Forecasting horizon
`level`	Confidence level for prediction intervals

lags	Number of lags
type_VAR	Type of deterministic regressors to include.
...	Additional parameters to be passed to vars::VAR.

Value

An object of class "mtsforecast"; a list containing the following elements:

method	The name of the forecasting method as a character string
mean	Point forecasts for the time series
lower	Lower bound for prediction interval
upper	Upper bound for prediction interval
x	The original time series
residuals	Residuals from the fitted model

Author(s)

T. Moudiki

References

Bernhard Pfaff (2008). VAR, SVAR and SVEC Models: Implementation Within R Package vars. Journal of Statistical Software 27(4). URL http://www.jstatsoft.org/v27/i04/.

Pfaff, B. (2008) Analysis of Integrated and Cointegrated Time Series with R. Second Edition. Springer, New York. ISBN 0-387-27960-1

Examples

```
require(fpp)

print(varf(fpp::insurance, lags=2, h=10))

res <- varf(fpp::usconsumption, h=20, lags=2)

par(mfrow=c(1, 2))
plot(res, "consumption")
plot(res, "income")
```

Index