Package: metalearnedridge2f (via r-universe)

May 26, 2026

Type Package

Title Forecasting With Meta-learned ridge2f

Version 0.1.0

Date 2026-05-17

Description Forecasting with meta-learned ridge2f.

License BSD_3_clause + file LICENSE

Imports Rcpp (>= 1.1.0), foreach, randtoolbox, np, tseries

Depends forecast, R (>= 3.5)

LinkingTo Rcpp

Suggests fpp2, snow, doSNOW, knitr, ahead, garchf, misc

RoxygenNote 7.3.2

Remotes Techtonique/ahead, thierrymoudiki/garchf, thierrymoudiki/misc

VignetteBuilder knitr

LazyData true

Config/pak/sysreqs make libssl-dev

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

Date/Publication 2026-05-26 11:34:53 UTC

RemoteUrl https://github.com/Techtonique/metalearnedridge2f

RemoteRef HEAD

RemoteSha dbd0c371729d0da333decb5d5c3359747dd7f127

Contents

`metalearnedridge2f-package` *Forecasting With Meta-learned ridge2f*

Description

Forecasting with meta-learned ridge2f.

Package Content

Index: This package was not yet installed at build time.

Maintainer

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

Author(s)

T. Moudiki [aut, cre]

`conformalize` *Conformal-style forecasting via dependent residual bootstrap*

Description

Constructs predictive distributions for time series forecasting models by combining a generic forecasting function with calibration residuals and dependent resampling via stationary bootstrap.

The method is not strict conformal prediction. Instead, it is a simulation-based forecasting framework where uncertainty is obtained by resampling calibrated residuals using a data-driven block length estimated via `b.star`.

Usage

```
conformalize(
  FUN,
  y,
  h,
  level = 95,
  nsim = 250L,
  scale_residuals = TRUE,
  seed = 123L,
  ...
)
```

Arguments

FUN	A forecasting function compatible with genericforecast (e.g. ETS, ARIMA, naive methods).
y	A numeric vector or ts object representing the time series.
h	Forecast horizon (number of steps ahead).
level	Confidence level for prediction intervals (e.g. 95).
nsim	Number of bootstrap simulations for residual paths.
scale_residuals	Logical. If TRUE, residuals are standardized before resampling (stabilization, not volatility modeling).
seed	Random seed for reproducibility.
...	Additional arguments passed to FUN through genericforecast.

Details

The procedure is:

1. Split the series into training and calibration sets.
2. Fit the forecasting model on the training set.
3. Compute calibration residuals.
4. Estimate dependence structure using np::b.star.
5. Generate dependent residual paths using tseries::tsbootstrap (stationary bootstrap).
6. Add simulated residuals to forecast means to obtain predictive distributions.

This method is particularly suited for financial time series such as log-returns, where dependence and non-Gaussianity are present but explicit parametric volatility models may be undesirable.

Value

An object of class forecast with components:

mean Point forecasts as a ts object

lower Lower prediction bounds

upper Upper prediction bounds

sims Simulated forecast paths

calib_resids Calibration residuals used for bootstrap

block_size Data-driven block length from `np::b.star`

fitted Fitted values from final model

See Also

`genericforecast`, `b.star`

Examples

```
## Not run:
y <- diff(log(fpp2::goog))
fit <- conformalize(FUN = forecast::auto.arima, y = y, h = 20)
plot(fit)

## End(Not run)
```

`forecast.ridge2` *Forecast method for ridge2 model using simulation-based uncertainty*

Description

Generate multi-step ahead forecasts from a fitted ridge2 model by refitting the underlying ridge2f model and propagating uncertainty through Monte Carlo simulation of residuals.

Usage

```
## S3 method for class 'ridge2'
forecast(
  object,
  h = 5L,
  level = 95,
  type_pi = c("bootstrap", "gaussian", "sv_generic"),
  vol_model = forecast::auto.arima,
  nsim = 250,
  seed = 42,
  ...
)
```

Arguments

object	An object containing a fitted ridge2 model, including: x, xreg, lags, nb_hidden, nodes_sim, activ, a, lambda_1, lambda_2, dropout, seed, centers, type_clustering, and cl.
h	Integer. Forecast horizon (number of steps ahead).
level	Numeric vector. Confidence levels for prediction intervals (e.g. 95). Intervals are computed empirically from simulations.
type_pi	A string. The type of prediction interval. "bootstrap" or "gaussian" or "sv_generic" (the latter use vol_model to model the latent volatility)
vol_model	volatility model, could be anything, e.g forecast::auto.arima, or forecast::auto.arima or forecast::ets, etc.
nsim	Integer. Number of Monte Carlo simulation paths used to approximate the predictive distribution. Default is 250.
seed	Integer. Random seed for reproducibility of simulations.
...	Additional parameters to be passed to the forecasting function for type_pi = "sv_generic".

Details

This method produces probabilistic forecasts by: (1) reconstructing the underlying ridge2f model, (2) generating future sample paths via residual-based simulation, (3) summarizing the simulated paths into point forecasts and prediction intervals.

Forecasts are generated using a simulation-based approximation of the predictive distribution. The underlying ridge2f model is refitted internally with horizon h, and future paths are generated using residual-based bootstrap simulation via metalearnedridge2f::simulate.

Let $y_t^{(b)}$ denote simulated future paths:

$$y_t^{(b)} = \hat{y}_t + \varepsilon_t^{*(b)}$$

where $\varepsilon_t^{*(b)}$ are resampled residuals.

Summary statistics are then computed as:

- Mean: rowMeans(sims)
- Lower bound: empirical quantiles at (1 - level)/2
- Upper bound: empirical quantiles at 1 - (1 - level)/2

This approach produces coherent probabilistic forecasts suitable for evaluation using proper scoring rules such as CRPS.

Value

An object of class "forecast" containing:

mean Point forecasts computed as row means of simulated paths

lower Lower prediction bounds computed from empirical quantiles

`upper` Upper prediction bounds computed from empirical quantiles

`level` Requested confidence levels for prediction intervals

`method` Forecasting method identifier ("ridge2")

`fitted` In-sample fitted values from original model

`residuals` In-sample residuals used for simulation

See Also

`ridge2f`, `simulate`, `crps_sample`, `forecast`

Examples

```
require(forecast)

par(mfrow=c(2, 2))

fit <- ridge2(USAccDeaths, lags = 15)
(fc1 <- forecast(fit, h = 20, nsim = 500, type_pi = "bootstrap"))
plot(fc1)

fit <- ridge2(USAccDeaths, lags = 15)
(fc2 <- forecast(fit, h = 20, nsim = 500, type_pi = "gaussian"))
plot(fc2)

fit <- ridge2(USAccDeaths, lags = 15)
(fc3 <- forecast(fit, h = 20, nsim = 500, type_pi = "sv_generic"))
plot(fc3)


fit <- ridge2(USAccDeaths, lags = 15)
(fc4 <- forecast(fit, h = 20, nsim = 500, type_pi = "sv_generic", vol_model=forecast::thetaf))
plot(fc4)

par(mfrow=c(1, 1))


par(mfrow=c(2, 2))

fit <- ridge2(diff(log(fpp2::goog200)), lags = 15)
(fc1 <- forecast(fit, h = 20, nsim = 500, type_pi = "bootstrap"))
plot(fc1)

fit <- ridge2(diff(log(fpp2::goog200)), lags = 15)
(fc2 <- forecast(fit, h = 20, nsim = 500, type_pi = "gaussian"))
plot(fc2)

fit <- ridge2(diff(log(fpp2::goog200)), lags = 15)
(fc3 <- forecast(fit, h = 20, nsim = 500, type_pi = "sv_generic"))
plot(fc3)
```

```
fit <- ridge2(diff(log(fpp2::goog200)), lags = 15)
(fc4 <- forecast(fit, h = 20, nsim = 500, type_pi = "sv_generic", vol_model=forecast::thetaf))
plot(fc4)

par(mfrow=c(1, 1))

# CRPS evaluation:
# scoringRules::crps_sample(y_true, dat = fc$sims)
```

`get_stock_params` *Retrieve Meta-Learned (on RMSE) Hyperparameters for Stock Return Forecasting*

Description

Returns the internally stored meta-learned hyperparameters obtained from clustering-based optimization for stock return forecasting models.

Usage

```
get_stock_params()
```

Details

These hyperparameters are loaded when the package is attached and can be used as defaults, warm starts, or priors for forecasting workflows.

The returned object is stored internally in the package namespace and accessed through a package-private environment.

Value

A list, data frame, or other structured object containing the best hyperparameter configurations associated with clustered stock return forecasting tasks.

Examples

```
params <- get_stock_params()
```

get_stock_params_crps *Retrieve Meta-Learned (on CRPS) Hyperparameters for Stock Return Forecasting*

Description

Returns the internally stored meta-learned hyperparameters obtained from clustering-based optimization for stock return forecasting models.

Usage

```
get_stock_params_crps()
```

Details

These hyperparameters are loaded when the package is attached and can be used as defaults, warm starts, or priors for forecasting workflows.

The returned object is stored internally in the package namespace and accessed through a package-private environment.

Value

A list, data frame, or other structured object containing the best hyperparameter configurations associated with clustered stock return forecasting tasks.

Examples

```
params <- get_stock_params_crps()
```

get_stock_params_crps_stochvol *Retrieve Meta-Learned (on CRPS) Hyperparameters for Stock Return Forecasting "Stoch vol"*

Description

Returns the internally stored meta-learned hyperparameters obtained from clustering-based optimization for stock return forecasting models.

Usage

```
get_stock_params_crps_stochvol()
```

Details

These hyperparameters are loaded when the package is attached and can be used as defaults, warm starts, or priors for forecasting workflows.

The returned object is stored internally in the package namespace and accessed through a package-private environment.

Value

A list, data frame, or other structured object containing the best hyperparameter configurations associated with clustered stock return forecasting tasks.

Examples

```
params <- get_stock_params_crps_stochvol()
```

`get_stock_params_crps_stochvol_on_garch_synth`
Retrieve Meta-Learned (on CRPS) Hyperparameters for Stock Return Forecasting "Stoch vol", pretrained on GARCH(1, 1) synthetic data

---

Description

Returns the internally stored meta-learned hyperparameters obtained from clustering-based optimization for stock return forecasting models.

Usage

```
get_stock_params_crps_stochvol_on_garch_synth()
```

Details

These hyperparameters are loaded when the package is attached and can be used as defaults, warm starts, or priors for forecasting workflows.

The returned object is stored internally in the package namespace and accessed through a package-private environment.

Value

A list, data frame, or other structured object containing the best hyperparameter configurations associated with clustered stock return forecasting tasks.

Examples

```
params <- get_stock_params_crps_stochvol_on_garch_synth()
```

get_stock_params_crps_with_garch_synth

Retrieve Meta-Learned (on CRPS) Hyperparameters for Stock Return Forecasting pretrained on GARCH(1, 1) synthetic data

Description

These hyperparameters are loaded when the package is attached and can be used as defaults, warm starts, or priors for forecasting workflows.

Usage

```
get_stock_params_crps_with_garch_synth()
```

Details

The returned object is stored internally in the package namespace and accessed through a package-private environment.

Value

A list, data frame, or other structured object containing the best hyperparameter configurations associated with clustered stock return forecasting tasks.

Examples

```
params <- get_stock_params_crps_with_garch_synth()
```

rcpp_hello_world *Simple function using Rcpp*

Description

Simple function using Rcpp

Usage

```
rcpp_hello_world()
```

Examples

```
## Not run:
rcpp_hello_world()

## End(Not run)
```

`returns_list` *Daily log-returns for international equity tickers*

Description

A named list of univariate time series objects containing daily log-returns computed from adjusted closing prices downloaded from Yahoo Finance using ‘quantmod::getSymbols‘.

Format

A named list of 10 objects of class ‘ts‘.

AAPL Apple daily log-returns

MSFT Microsoft daily log-returns

JPM JPMorgan Chase daily log-returns

AIR.PA Airbus daily log-returns

MC.PA LVMH daily log-returns

SAN.PA Sanofi daily log-returns

VOW3.DE Volkswagen daily log-returns

SAP.DE SAP daily log-returns

NESN.SW Nestlé daily log-returns

NOVN.SW Novartis daily log-returns

Details

Returns are computed as:

$$r_t = \log(P_t) - \log(P_{t-1})$$

where P_t denotes the adjusted closing price at time t.

The series have trading-day frequency (‘frequency = 252‘) and cover the period from 2018-01-01 to 2019-12-02.

The dataset includes equity tickers from multiple international indices:

- United States: AAPL, MSFT, JPM
- France (CAC 40): AIR.PA, MC.PA, SAN.PA
- Germany (DAX): VOW3.DE, SAP.DE
- Switzerland (SMI): NESN.SW, NOVN.SW

Source

Adjusted daily prices obtained from Yahoo Finance.

Examples

```
data(returns_list)

names(returns_list)

plot(returns_list$AAPL)

summary(returns_list$MC.PA)
```

`ridge2` *Ridge2 model for use with* `forecast::forecast`

Description

Random Vector functional link network model with 2 regularization parameters

Usage

```
ridge2(
  y,
  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,
  seed = 1,
  centers = NULL,
  type_clustering = c("kmeans", "hclust"),
  cl = 1L,
  ...
)
```

Arguments

`y`	A univariate of multivariate time series of class `ts` (preferred) or a `matrix`
`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)
seed	Reproducibility seed for ‘nodes_sim == unif‘
centers	Number of clusters for type_clustering
type_clustering	"kmeans" (K-Means clustering) or "hclust" (Hierarchical clustering)
cl	An integer; the number of clusters for parallel execution, for bootstrap
...	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

```
(fit <- ridge2(AirPassengers, lags = 20))
```

ridge2f *Ridge2 model retrieved from https://github.com/Techtonique/ahead/blob/main/R/ridge2.R on 2026-05-17, removed "rvinecopula" for univariate forecasting*

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", "none"),
  block_length = NULL,
  seed = 1,
  B = 100L,
  type_aggregation = c("mean", "median"),
  centers = NULL,
  type_clustering = c("kmeans", "hclust"),
  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"
block_length	Length of block for circular or moving block bootstrap
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)
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(forecast)
plot(metalearnedridge2f::ridge2f(AirPassengers, h=20, lags=20))
plot(metalearnedridge2f::ridge2f(USAccDeaths, h=20, lags=15))
```

`simulate.ridge2` *Simulate future paths from a ridge2f forecast object*

Description

Generate Monte Carlo sample paths from a fitted `ridge2f` model using a residual bootstrap approach. Future values are simulated by adding resampled in-sample residuals to the model's point forecasts.

Usage

```
## S3 method for class 'ridge2'
simulate(
  object,
  nsim = length(object$x),
  type = c("bootstrap", "gaussian", "sv_generic"),
  vol_model = forecast::auto.arima,
  seed = 42,
  ...
)
```

Arguments

`object`	An object of class `ridge2f`, typically the output of `ridge2f()` containing at least `mean` and `residuals`.
`nsim`	Integer. Number of Monte Carlo simulation paths to generate. Default is `1000`.
`type`	A string. The type of simulation, either "bootstrap" or "gaussian" or "sv_generic" (the latter use `vol_model` to model the latent volatility)
`vol_model`	volatility model, could be anything, e.g `forecast::auto.arima`, or `forecast::auto.arima` or `forecast::ets`, etc.
`seed`	Integer. Random seed for reproducibility. Default is `42`.
`...`	Additional parameters to be passed to the forecasting function for `type = "sv_generic"`.

Details

This provides an empirical approximation of the predictive distribution, which can be used for probabilistic forecasting, uncertainty quantification, and proper scoring rules such as CRPS via `scoringRules::crps_sample`.

The simulation model assumes:

$$y_t^{(b)} = \mu_t + \varepsilon^{*(b)}$$

where μ_t is the point forecast and $\varepsilon^{*(b)}$ is a residual sampled with replacement from in-sample residuals.

This is a non-parametric bootstrap approximation of the predictive distribution and does not assume Gaussianity.

Value

A numeric matrix of dimension `h` x `nsim`, where:

`h` Forecast horizon (length of `object$mean`)

`nsim` Number of simulated future trajectories

Each column represents one simulated future path.

See Also

`ridge2f`, `crps_sample`, `forecast`

Examples

```
fit <- metalearnedridge2f::ridge2(AirPassengers, lags = 20L)
sims <- simulate(fit, nsim = 500, seed = 123)

# CRPS evaluation
# scoringRules::crps_sample(y = true_values, dat = sims)
```

---

stocklogreturns2f *Forecasting stock log-returns 2 (pretrained to minimize CRPS)*

Description

Forecasting stock log-returns with a meta-learned (to minimize CRPS) model based on ridge2f

Usage

```
stocklogreturns2f(y, h = 10L, level = 95, seed = 1, B = 250L)
```

Arguments

`y`	A univariate time series of class `ts`
`h`	Forecasting horizon
`level`	Confidence level for prediction intervals
`seed`	Reproducibility seed for random stuff
`B`	Number of bootstrap replications or number of simulations

Examples

```
require(fpp2)
plot(metalearnedridge2f::stocklogreturns2f(diff(log(fpp2::goog200)), h=20))
```

`stocklogreturns3f` *Forecasting stock log-returns 2 (pretrained to minimize CRPS) with stoch vol (finally, vol with forecast::meanf)*

Description

Forecasting stock log-returns with a meta-learned (to minimize CRPS) model based on ridge2f with stoch vol (finally, vol with forecast::meanf)

Usage

```
stocklogreturns3f(y, h = 10L, level = 95, seed = 1, B = 250L)
```

Arguments

`y`	A univariate time series of class `ts`
`h`	Forecasting horizon
`level`	Confidence level for prediction intervals
`seed`	Reproducibility seed for random stuff
`B`	Number of bootstrap replications or number of simulations

Examples

```
# this returns an error; the hyperparams produce a degenerate matrix
#require(fpp2)
#plot(metalearnedridge2f::stocklogreturns3f(diff(log(fpp2::goog200)), h=20))
```

`stocklogreturns4f` *Forecasting stock log-returns 2 (pretrained to minimize CRPS) with stoch vol (finally, vol with forecast::meanf), pretrained on GARCH(1, 1) synthetic data*

Description

Forecasting stock log-returns with a meta-learned (to minimize CRPS) model based on ridge2f with stoch vol (finally, vol with forecast::meanf)

Usage

```
stocklogreturns4f(y, h = 10L, level = 95, seed = 1, B = 250L)
```

Arguments

y	A univariate time series of class ts
h	Forecasting horizon
level	Confidence level for prediction intervals
seed	Reproducibility seed for random stuff
B	Number of bootstrap replications or number of simulations

Examples

```
require(fpp2)
plot(metalearnedridge2f::stocklogreturns4f(diff(log(fpp2::goog200)), h=20))
```

---

stocklogreturns5f *Forecasting stock log-returns, pretrained to minimize CRPS on GARCH(1, 1) synthetic data*

Description

Forecasting stock log-returns with a meta-learned (to minimize CRPS) model based on ridge2f pretrained on GARCH(1, 1) synthetic data

Usage

```
stocklogreturns5f(y, h = 10L, level = 95, seed = 1, B = 250L)
```

Arguments

y	A univariate time series of class ts
h	Forecasting horizon
level	Confidence level for prediction intervals
seed	Reproducibility seed for random stuff
B	Number of bootstrap replications or number of simulations

Examples

```
require(fpp2)
plot(metalearnedridge2f::stocklogreturns5f(diff(log(fpp2::goog200)), h=20))
```

`stocklogreturnsf` *Forecasting stock log-returns*

Description

Forecasting stock log-returns with a meta-learned (to minimize RMSE) model based on ridge2f

Usage

```
stocklogreturnsf(
  y,
  h = 10L,
  level = 95,
 type_pi = c("gaussian", "bootstrap", "blockbootstrap", "movingblockbootstrap"),
  seed = 1,
  B = 250L
)
```

Arguments

`y`	A univariate time series of class `ts`
`h`	Forecasting horizon
`level`	Confidence level for prediction intervals
`type_pi`	Type of prediction interval currently "gaussian", "bootstrap", "blockbootstrap", "movingblockbootstrap", "conformal-split", "conformal-bootstrap", "conformal-block-bootstrap"
`seed`	Reproducibility seed for random stuff
`B`	Number of bootstrap replications or number of simulations (yes, 'B' is unfortunate)

Examples

```
require(fpp2)
plot(metalearnedridge2f::stocklogreturnsf(diff(log(fpp2::goog200)), h=20))
```

Index