Package ‘LasForecast’

December 10, 2022

Title Time series linear predictive regression

Version 0.0.0.9000

Description This package develops a framework for economic forecasting in a data-rich environment with a particular emphasis on linear predictive regression. The goal is to automate the processes of parameter tuning, rolling window forecasting and backtesting, and visualization in a unified framework.

Depends R (>= 3.5.2),
glmnet,
slam

License MIT

Encoding UTF-8

LazyData true

Imports caret,
Matrix,
lubridate,
zoo,
reshape2,
CVXR,
SparseM

Suggests Rmosek,
doParallel,
parallel,
foreach,
dplyr,
knitr,
rmarkdown,
gurobi

RoxygenNote 7.2.1

VignetteBuilder knitr

R topics documented:

`adalasso` *Implement Adaptive Lasso via glmnet*

Description

Estimation function. Tuning parameter inputs needed.
Incorporates both high-dim(Lasso as initial estimator) and low-dim (OLS as initial estimator).

Usage

```
adalasso(
  x,
  y,
  lambda,
  lambda_lasso = NULL,
  gamma = 1,
  intercept = TRUE,
  scalex = FALSE
)
```

Arguments

`x`	Predictor matrix (n-by-p matrix)
`y`	Response variable
`lambda`	Shrinkage tuning parameter for the adaptive step
`lambda_lasso`	Shrinkage tuning parater for the initial step Lasso estimation (If applicable)
`gamma`	Parameter controlling the inverse of first step estimate
`intercept`	A boolean: include an intercept term or not
`scalex`	A boolean: standardize the design matrix or not

Value

A list contains estimated intercept and slope

ahat	Estimated intercept
bhat	Estimated slope

Examples

```
adalasso(x,y)
```

---

bss — *Implement best subset selection via Mixed Integer Optimization (MIO) for given k*

Description

Implement best subset selection via Mixed Integer Optimization (MIO) for given k

Usage

```
bss(
  y,
  X,
  k,
  intercept = TRUE,
  b0 = NULL,
  tau = 2,
  tol = 1e-04,
  MaxIter = 10000,
  polish = TRUE,
  time.limit = 1200
)
```

---

bss.bic — *Implement best subset selection via Mixed Integer Optimization (MIO) Select k by BIC*

Description

Make use of Gurobi solver

Usage

```
bss.bic(
  y,
  X,
  intercept = TRUE,
  b0 = NULL,
  tau = 2,
  tol = 1e-04,
  MaxIter = 10000,
  polish = TRUE,
  time.limit = 1200,
  RW = TRUE
)
```

Arguments

`y`	forecast target
`X`	predictors
`b0`	initial estimator
`tau`	parameter to obtain bounds
`tol`	precision tolerence
`polish`	whether post-selection polish is conducted
`time.limit`	in seconds
`RW`	consider k equal to 0 or not

Value

A list contains estimated k and corresponding coefficient

`k`	estimated k
`coef`	estimated coefficient

`csr` *Elliott, Gargano and Timmermann (2013) COMPLETE SUBSET REGRESSIONS*

Description

Incorporates techniques in section 3.4 This step aims to reduce the computation burden when K is large, but it is not feasible when K is too large using standard R functions For example, when K = 50, k = 25, the vector 1:choose(50,25) consumes 941832.4 Gb memory, which is an astronomical number Future worke: see Boot and Nibbering (2019) Random subspace method.

Usage

```
csr(y, X, k, C.upper = 5000, intercept = FALSE)
```

Arguments

`y`	response variable
`X`	Predictor matrix
`k`	subset size
`C.upper`	maximum number of subsets to be combined
`intercept`	A boolean: include an intercept term or not

`csr.bic` *Elliott, Gargano and Timmermann (2013) COMPLETE SUBSET REGRESSIONS with BIC*

Description

Choose tuning parameter k by Bayesian Information Criterion (BIC)

Usage

```
csr.bic(y, X, C.upper = 5000, intercept = FALSE, RW = TRUE)
```

Arguments

`y`	response variable
`X`	Predictor matrix
`C.upper`	maximum number of subsets to be combined
`intercept`	A boolean: include an intercept term or not

Value

A List contained the estimated coefficients and forecasts

`k.hat`	k chosen by BIC
`coef`	Averaged coefficients corresonding to the chosen k

`est_sigma_mat` *Estimate the sample covariance matrix*

Description

Consider different methods later on.

Usage

```
est_sigma_mat(y, x)
```

Arguments

`y`	forecast target
`x`	forecasts to be combined

Value

sigma_mat: The estimated sample covariance matrix

`find_tau_max` *Find the largest tau for L2-Relaxation*

Description

This function finds the smallest tau for L2-Relaxation such that equal-weight solves the forecast combination optimization.

Usage

```
find_tau_max(sigma_mat)
```

Arguments

`sigma_mat`	Sample covariance matrix

Value

smallest tau corresponds to equal-weight

`find_tau_max_reg` *Find the minimum tau such that equal weight solve the l_2 relaxation problem*

Description

Find the minimum tau such that equal weight solve the l_2 relaxation problem

Usage

```
find_tau_max_reg(y, X, solver = "CVXR", intercept = TRUE, tol = 1e-06)
```

Arguments

`solver`	The solver to use; "Rmosek" or "CVXR"
`tol`	Tolerance for the solver

l2_relax — *l2-relaxation forecast combination A wrapper function with parameter tuning and estimation.*

Description

Details refer to Shi, Su and Xie (2022) "l_2-relaxation: With applications to forecast combination and portfolio analysis"

Usage

```
l2_relax(y, x, tau, solver = "CVXR", tol = 1e-08)
```

Arguments

`y`	foreccasting target
`x`	forecasts to be combined
`tau`	The regularization parameter
`solver`	The solver to use; "Rmosek" or "CVXR"
`tol`	Tolerance for the solver

Value

A list

l2_relax_comb_opt — *Solve L2-relaxation primal problem Details refer to Shi, Su and Xie (2022) "l_2-relaxation: With applications to forecast combination and portfolio analysis"*

Description

Solve L2-relaxation primal problem Details refer to Shi, Su and Xie (2022) "l_2-relaxation: With applications to forecast combination and portfolio analysis"

Usage

```
l2_relax_comb_opt(sigma_mat, tau, solver = "CVXR", tol = 1e-08)
```

Arguments

`sigma_mat`	Sample covariance matrix
`tau`	The regularization parameter
`solver`	The solver to use; "Rmosek" or "CVXR"
`tol`	Tolerance for the solver

Value

w_hat: The estimated combination weights l2_relax_reg_opt *Solve L2-relaxation for a regression problem*

Description

Solve L2-relaxation for a regression problem

Usage

```
l2_relax_reg_opt(y, X, tau, intercept = TRUE, solver = "CVXR", tol = 1e-06)
```

Arguments

tau	The regularization parameter
solver	The solver to use; "Rmosek" or "CVXR"
tol	Tolerance for the solver

LasForecast *LasForecast: Time series linear predictive regression*

lasso_weight_opt *Implement weighted Lasso estimation (second step of adaptive lasso) via optimization solver*

Description

Estimation function. Tuning parameter inputs needed.

Usage

```
lasso_weight_opt(
  x,
  y,
  lambda,
  w = NULL,
  intercept = TRUE,
  scalex = FALSE,
  solver = "CVXR",
  rtol = 1e-08,
  verb = 0
)
```

Arguments

x	Predictor matrix (n-by-p matrix)
y	Response variable
lambda	Shrinkage tuning parameter
w	weights
intercept	A boolean: include an intercept term or not
scalex	A boolean: standardize the design matrix or not
solver	indicate solver in use, c("CVXR", "MOSEK")

Value

A list contains estimated intercept and slope

ahat	Estimated intercept
bhat	Estimated slope

Examples

```
lasso_weight_opt(x,y)
```

---

lasso_weight_single *Implement weighted Lasso estimation (second step of adaptive lasso) via coordinate descent for the case in which only one predictor remains*

Description

Estimation function. Tuning parameter inputs needed.

Usage

```
lasso_weight_single(x, y, lambda, w = NULL, intercept = TRUE, scalex = FALSE)
```

Arguments

x	Predictor matrix (n-by-p matrix)
y	Response variable
lambda	Shrinkage tuning parameter
w	weights
intercept	A boolean: include an intercept term or not
scalex	A boolean: standardize the design matrix or not

Value

A list contains estimated intercept and slope

ahat	Estimated intercept
bhat	Estimated slope

plot_coef *Coefficient plot*

Description

Coefficient plot

Usage

```
plot_coef(
  coef_est,
  dates,
  pt_num = 4,
  col_vec = NULL,
  line_size = rep(0.75, length(coef_est)),
  alpha_size = rep(0.75, length(coef_est)),
  xlab = NULL,
  ylab = NULL,
  num_col = 1
)
```

Arguments

`coef_est`	estimated slope by m different methods, list of length m with each element n-by-p matrix names of list well defined
`dates`	Date vector of class "Date".
`pt_num`	Number date points (pt_num + 1) wanted on the x-axis Use lubridate and zoo to transfer original date vector to "Date" class
`col_vec`	A vector indicates colors of different trends. e.g.: c("#D8DBE2", "#F46B7B", "#518DE8", "#FFBC42") If NULL, use ggplot default.
`line_size`	line size for each trend
`alpha_size`	degree of appearance
`xlab`	x-axis label
`ylab`	y-axis label

Value

A ggplot output

Examples

```
data("forecast_result")
data("raw_data_h1")
D <- raw_data_h1[!is.na(raw_data_h1$LongReturn), ]
dates <- zoo::as.Date.yearmon(D$yyyymm[-(1:180)])

coef_est <- list(lasso = forecast_result$Lasso$beta_hat[, 1:6],
```

```
                 alasso = forecast_result$ALasso$beta_hat[, 1:6])
plot_coef(coef_est, dates)
```

`plot_trend` *Trend plot*

Description

Trend plot

Usage

```
plot_trend(
  y_0,
  y_hat,
  dates,
  pt_num = 4,
  col_vec = NULL,
  line_size = rep(0.75, ncol(y_hat) + 1),
  alpha_size = rep(0.75, ncol(y_hat) + 1),
  xlab = NULL,
  ylab = NULL
)
```

Arguments

`y_0`	True predict target, length n vector
`y_hat`	Predicted values by m different methods, n-by-m matrix, colnames well defined
`dates`	Date vector of class "Date". Use lubridate and zoo to transfer original date vector to "Date" class
`pt_num`	Number date points (pt_num + 1) wanted on the x-axis
`col_vec`	A vector indicates colors of different trends. e.g.: c("#D8DBE2", "#F46B7B", "#518DE8", "#FFBC42") If NULL, use ggplot default.
`line_size`	line size for each trend
`alpha_size`	degree of appearance
`xlab`	x-axis label
`ylab`	y-axis label

Value

A ggplot output

Examples

```
data("forecast_result")
data("raw_data_h1")
D <- raw_data_h1[!is.na(raw_data_h1$LongReturn), ]
dates <- zoo::as.Date.yearmon(D$yyyymm[-(1:180)])
y0 <- forecast_result$y[-(1:180)]
y_hat <-cbind(forecast_result$Lasso$y_hat,
              forecast_result$Lasso_Std$y_hat,
              forecast_result$ALasso$y_hat)
colnames(y_hat) <- c("lasso", "lasso_std", "alasso")
plot_trend(y0, y_hat, dates)
```

---

post_lasso *post-selection estimation*

Description

post-selection estimation

Usage

```
post_lasso(x, y, coef.est, intercept = TRUE, scalex = FALSE)
```

Arguments

`x`	Predictor matrix (n-by-p matrix)
`y`	Response variable
`coef.est`	shrinkage estimation results
`intercept`	A boolean: include an intercept term or not

Value

A list contains estimated intercept and slope

`ahat`	Estimated intercept
`bhat`	Estimated slope

Examples

```
post_lasso(x,y,coef.est)
```

roll_predict *Rolling window forecast*

Description

The function reads data and make forecasts based on linear predictive regression with diverse methods. It incorporates both short-horizon and long-horizon forecasting.

Usage

```
roll_predict(
  x,
  y,
  roll_window,
  h = 1,
  methods_use = c("RW", "RWwD", "OLS", "Lasso", "Lasso_Std", "ALasso", "TALasso",
    "post_Lasso", "post_Lasso_Std", "post_ALasso", "post_TALasso", "bss"),
  train_method_las = "cv",
  verb = TRUE,
  ar_order = 0
)
```

Arguments

x	Full sample predictor
y	Full sample forecast target
roll_window	Length of the rolling window
h	forecast horizon
train_method_las	parameter tuning method for Lasso type methods. For comparison, all Lasso type methods shares the same train_method.
verb	boolean to control whether print information on screen
ar_order	0 or 1 to control whether include ar1 lag or not
method_use	method in use

roll_predict_l2relax *Rolling window forecast*

Description

The function reads data and make forecasts based on linear predictive regression with diverse methods. It incorporates both short-horizon and long-horizon forecasting.

Usage

```
roll_predict_l2relax(
  x,
  y,
  roll_window,
  h = 1,
  k_max = 4,
  m = 5,
  ntau = 100,
  tau_min_ratio = 0.01,
  train_method = "oos",
  solver = "CVXR",
  tol = 1e-07,
  verb = TRUE,
  csr = TRUE
)
```

Arguments

`x`	Full sample predictor
`y`	Full sample forecast target
`roll_window`	Length of the rolling window
`h`	forecast horizon
`m`	number of folds
`ntau`	number of tau values
`train_method`	parameter tuning method for L2relax "cv_random", "cv" or "oos"
`solver`	"Rmosek" or "CVXR"
`tol`	tolerance for the solver
`verb`	boolean to control whether print information on screen
`csr`	boolean to opt out for the csr
`tau.min.ratio`	ratio of the minimum tau in tau.seq over the maximum (which is the smallest tau such that equal-weight solves the forecast combination optimization.)

`talasso` *Implement repeated Lasso estimation*

Description

Estimation function. Tuning parameter inputs needed. First step adalasso estimate needed.

Usage

```
talasso(x, y, b.first, lambda, gamma = 1, intercept = TRUE, scalex = FALSE)
```

Arguments

`x`	Predictor matrix (n-by-p matrix)
`y`	Response variable
`b.first`	First step adaptive lasso estimates
`lambda`	Shrinkage tuning parameter
`gamma`	Parameter controlling the inverse of first step estimate
`intercept`	A boolean: include an intercept term or not
`scalex`	A boolean: standardize the design matrix or not

Value

A list contains estimated intercept and slope

`ahat`	Estimated intercept
`bhat`	Estimated slope

Examples

```
talasso(x,y, b.first,lambda)
```

---

train_l2_relax *Do parameter tuning for L2-Relaxation*

Description

Do parameter tuning for L2-Relaxation

Usage

```
train_l2_relax(
  y,
  x,
  m = 5,
  tau.seq = NULL,
  ntau = 100,
  tau.min.ratio = 0.01,
  train_method = "oos",
  solver = "Rmosek",
  tol = 1e-05
)
```

Arguments

`y`	foreccasting target
`x`	forecasts to be combined
`m`	number of folds
`tau.seq`	Sequence of tau values
`ntau`	number of tau values

tau.min.ratio	ratio of the minimum tau in tau.seq over the maximum (which is the smallest tau such that equal-weight solves the forecast combination optimization.)
train_method	"cv_random", "cv" or "oos"
solver	"Rmosek" or "CVXR"

Value

besttune

---

train_lasso *Do parameter tuning for Lasso and Adaptive lasso*

Description

Do parameter tuning for Lasso and Adaptive lasso

Usage

```
train_lasso(
  x,
  y,
  ada = TRUE,
  gamma = 1,
  intercept = TRUE,
  scalex = FALSE,
  lambda_seq = NULL,
  train_method = "timeslice",
  nlambda = 100,
  lambda_min_ratio = 1e-04,
  k = 10,
  initial_window = ceiling(nrow(x) * 0.7),
  horizon = 1,
  fixed_window = TRUE
)
```

Arguments

x	Predictor matrix (n-by-p matrix)
y	Response variable
ada	A boolean: Do parameter tuning for adaptive Lasso if TRUE (Default) For Lasso if FALSE.
gamma	Parameter controlling the inverse of first step estimate. By default = 1.
intercept	A boolean: include an intercept term or not
scalex	A boolean: standardize the design matrix or not
lambda_seq	Candidate sequnece of parameters. If NULL, the function generates the sequnce.
train_method	"timeslice", "cv", "aic", "bic", "aicc", "hqc"
nlambda	# of lambdas

`lambda_min_ratio`	# lambda_min_ratio * lambda_max = lambda_min
`k`	k-fold cv if "cv" is chosen
`initial_window`	control "timeslice"
`horizon`	control "timeslice"
`fixed_window`	control "timeslice"

Value

bestTune

Examples

```
train_lasso(x,y)
```

---

train_talasso *Do parameter tuning for replasso*

Description

If all variables are killed in the first step: return a random number
If more than 1 variables are left: just repeat the training process for alasso
If only 1 variable remained: use a brute-force process do the cross-validation.
FOR FUTURE WORK: INCORPORATE REPLASSO INTO CARET FRAMEWORK.

Usage

```
train_talasso(
  x,
  y,
  b_first,
  gamma = 1,
  intercept = TRUE,
  scalex = FALSE,
  train_method = "timeslice",
  lambda_seq = NULL,
  nlambda = 100,
  lambda_min_ratio = 1e-04,
  k = 10,
  initial_window = ceiling(nrow(x) * 0.7),
  horizon = 1,
  fixed_window = TRUE
)
```

Arguments

`x`	Predictor matrix (n-by-p matrix)
`y`	Response variable
`b_first`	estimated slope from first step alasso
`gamma`	Parameter controlling the inverse of first step estimate. By default = 1.

`intercept`	A boolean: include an intercept term or not
`scalex`	A boolean: standardize the design matrix or not
`train_method`	"timeslice", "cv", "aic", "bic", "aicc", "hqc"
`lambda_seq`	Candidate sequnece of parameters. If NULL, the function generates the sequnce.
`nlambda`	# of lambdas
`lambda_min_ratio`	# lambda_min_ratio * lambda_max = lambda_min
`k`	k-fold cv if "cv" is chosen
`initial_window`	control "timeslice"
`horizon`	control "timeslice"
`fixed_window`	control "timeslice"

Value

bestTune

Examples

```
train_talasso(x,y)
```

`US_industry_prod` *Growth Rate of US industrial Production Index.*

Description

A monthly dataset containing the growth rate of US Industrial Production Index and other macroeconomic variables from 1960-01-01 to 2022-04-01. Note that all variables have been rescaled to have sd of 1.

Usage

```
US_industry_prod
```

Format

A data frame with 748 rows and 110 variables:

Ind_Growth_Rate monthly growth rate of US Industrial Production Index

RPI real personal income

W875RX1 real personal income ex transfer receipts ...

Source

Industrial Production Index https://fred.stlouisfed.org/series/INDPRO

Macroeconomic Factors https://research.stlouisfed.org/econ/mccracken/fred-databases/

Index