Package ‘SynthVolForecast’

January 25, 2024

Type Package

Title Apply Synthetic Methods to Forecast Volatility in Time Series

Version 0.1.0

Author David Lundquist

Maintainer David Lundquist <davidpatricklundquist@gmail.com>

Description Provides functions for forecasting using synthetic methods,
both for the observable time series and the unobservable time-varying volatility.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports forecast, garchx, lmtest, RColorBrewer, Rsolnp

RoxygenNote 7.2.3

R topics documented:

dbw *A function that carries out distance-based weighting.*

Description

A function that carries out distance-based weighting.

Usage

```
dbw(X, dbw_indices, shock_time_vec, scale = FALSE, center = FALSE, sum_to_1 = 1, bounded_below_by = 0, b
```

Arguments

X

dbw_indices

shock_time_vec

scale

center

sum_to_1

bounded_below_by

bounded_above_by

normchoice

penalty_normchoice

penalty_lambda

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
```

plot_maker_garch *A function that makes plots for SynthVolForecast*

Usage

```
plot_maker_garch(fitted_vol, shock_time_labels, shock_time_vec, shock_length_vec, unadjusted_pred, w_
```

Arguments

`fitted_vol`

`shock_time_labels`

`shock_time_vec`

`shock_length_vec`

`unadjusted_pred`

`w_hat`

`omega_star_hat`

`omega_star_hat_vec`

`adjusted_pred`

`arithmetic_mean_based_pred`

`ground_truth_vec`

Author(s)

David Lundquist

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
```

plot_maker_synthprediction

Function that makes plots for SynthPrediction

Usage

```
plot_maker_synthprediction(Y, shock_time_labels, shock_time_vec, shock_length_vec, unadjusted_pred, w
```

Arguments

Y

shock_time_labels

shock_time_vec

shock_length_vec

unadjusted_pred

w_hat

omega_star_hat

omega_star_hat_vec

adjusted_pred

display_ground_truth

Author(s)

David Lundquist

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
```

QL_loss_function *Quasi-likelihood Loss*

Usage

```
QL_loss_function(x)
```

Arguments

`x`

Author(s)

David Lundquist

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
```

SynthPrediction *A function that uses synthetic methods to predict observable levels.*

Usage

```
SynthPrediction(Y_series_list, covariates_series_list, shock_time_vec, shock_length_vec, dbw_scale =
```

Arguments

`Y_series_list`

`covariates_series_list`

`shock_time_vec`

`shock_length_vec`

`dbw_scale`

`dbw_center`

`dbw_indices`

`covariate_indices`

`geometric_sets`

`days_before_shocktime_vec`

arima_order

user_ic_choice

plots

display_ground_truth_choice

Author(s)

David Lundquist

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
```

SynthVolForecast *A function to forecast volatility using synthetic methods*

Usage

```
SynthVolForecast(Y_series_list, covariates_series_list, shock_time_vec, shock_length_vec, dbw_scale =
```

Arguments

Y_series_list

covariates_series_list

shock_time_vec

shock_length_vec

dbw_scale

dbw_center

dbw_indices

covariate_indices

geometric_sets

days_before_shocktime_vec

garch_order

common_series_assumption

plots

shock_time_labels

ground_truth_vec

Author(s)

David Lundquist

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
### START SynthVolForecast
                      function(Y_series_list
                             ,covariates_series_list
                             ,shock_time_vec
                             ,shock_length_vec
                             ,dbw_scale = TRUE
                             ,dbw_center = TRUE
                             ,dbw_indices = NULL
                             ,covariate_indices = NULL
                             ,geometric_sets = NULL #tk
                          ,days_before_shocktime_vec = NULL #tk I may want to remove this
                             ,garch_order = NULL
                             ,common_series_assumption = FALSE
                             ,plots = TRUE
                             ,shock_time_labels
                             ,ground_truth_vec
){
  ### BEGIN Doc string
  #tk
  ### END Doc string

  ### BEGIN Populate defaults
  n <- length(Y_series_list) - 1

  if (is.null(garch_order) == TRUE) {garch_order <- c(1,1,1)}

  if (is.null(dbw_indices) == TRUE) {dbw_indices <- 1:ncol(covariates_series_list[[1]])}

  ### END Populate defaults

  ## BEGIN Check that inputs are all comformable/acceptable
  n <- length(Y_series_list) - 1 #tk
  ## END Check that inputs are all comformable/acceptable

  integer_shock_time_vec <- c() #mk
  integer_shock_time_vec_for_convex_hull_based_optimization <- c() #mk

  ## BEGIN Check whether shock_time_vec is int/date

  for (i in 1:(n+1)){

    if (is.character(shock_time_vec[i]) == TRUE){
```

```
    integer_shock_time_vec[i] <- which(index(Y[[i]]) == shock_time_vec[i]) #mk
  integer_shock_time_vec_for_convex_hull_based_optimization[i] <- which(index(X[[i]]) == shock_time_vec[i]) #m
  }
  else{
    integer_shock_time_vec[i] <- shock_time_vec[i]
    integer_shock_time_vec_for_convex_hull_based_optimization <- shock_time_vec[i]
  }

}

## END Check whether shock_time_vec is int/date

## BEGIN calculate weight vector
dbw_output <- dbw(covariates_series_list, #tk
             dbw_indices,
             integer_shock_time_vec_for_convex_hull_based_optimization,
             scale = dbw_scale,
             center = dbw_center,
             sum_to_1 = TRUE, #tk
             bounded_below_by = 0, #tk
             bounded_above_by = 1, #tk
             # normchoice = normchoice, #tk
             # penalty_normchoice = penalty_normchoice,
             # penalty_lambda = penalty_lambda
             )

w_hat <- dbw_output[[1]]

## END calculate weight vector

## BEGIN estimate fixed effects in donors
omega_star_hat_vec <- c()

if (common_series_assumption == TRUE){
  print('tk TODO')

  #step 1: create dummy vector with n+1 shocks
  #NOTA BENE: n different fixed effects, or
  # 1 fixed effect estimated at n shocks?
    vec_of_zeros <- rep(0, integer_shock_time_vec[i])
    vec_of_ones <- rep(1, shock_length_vec[i])
    post_shock_indicator <- c(vec_of_zeros, vec_of_ones)
    last_shock_point <- integer_shock_time_vec[i] + shock_length_vec[i]

  #step 2: fit model


}

else{

  for (i in 2:(n+1)){
    # Make indicator variable w/ a 1 at only T*+1, T*+2,...,T*+shock_length_vec[i]
    vec_of_zeros <- rep(0, integer_shock_time_vec[i])
    vec_of_ones <- rep(1, shock_length_vec[i])
    post_shock_indicator <- c(vec_of_zeros, vec_of_ones)
    last_shock_point <- integer_shock_time_vec[i] + shock_length_vec[i]

    #subset X_i
    if (is.null(covariate_indices) == TRUE) {
      X_i_penultimate <- cbind(Y_series_list[[i]][1:last_shock_point] #tk
                               , post_shock_indicator)
      X_i_final <- X_i_penultimate[,2]
    }
    else {
      X_i_subset <- X[[i]][1:last_shock_point,covariate_indices]
      X_i_with_indicator <- cbind(X_i_subset, post_shock_indicator)
      X_i_final <- X_i_with_indicator
    }

    fitted_garch <- garchx(Y_series_list[[i]][1:last_shock_point] #tk
                   , order = garch_order
                   , xreg = X_i_final
                   , backcast.values = NULL
                   , control = list(eval.max = 100000
                   , iter.max = 1500000
                   , rel.tol = 1e-8))

    cat('\n===============================================================\n')
   print(paste('Outputting GARCH estimates for donor series number ', i,'.', sep = ''))
    print(fitted_garch)
    print(paste('Outputting AIC for donor series number ', i,'.', sep = ''))
    print(AIC(fitted_garch))
    cat('\n===============================================================\n')

    coef_test <- coeftest(fitted_garch)
    extracted_fixed_effect <- coef_test[dim(coeftest(fitted_garch))[1], 1]
    omega_star_hat_vec <- c(omega_star_hat_vec, extracted_fixed_effect)

  } ## END loop for computing fixed effects

}

## END estimate fixed effects in donors

## BEGIN compute linear combination of fixed effects
omega_star_hat <- w_hat
## END compute linear combination of fixed effects

## BEGIN fit GARCH to target series

if (is.null(covariate_indices) == TRUE){

  fitted_garch <- garchx(Y_series_list[[1]][1:integer_shock_time_vec[1]]
                         , order = garch_order
                         , xreg = NULL
                         , backcast.values = NULL
                         , control = list(eval.max = 100000
                                          , iter.max = 1500000
                                          , rel.tol = 1e-8))

  cat('\n================================================================\n')
  print('Outputting the fitted GARCH for time series under study.')
  print(fitted_garch)
  print('Outputting AIC for time series under study.')
  print(AIC(fitted_garch))
  cat('\n================================================================\n')

  unadjusted_pred <- predict(fitted_garch, n.ahead = shock_length_vec[1])
}
else{
  ## BEGIN fit GARCH to target series
  fitted_garch <- garchx(Y_series_list[[1]][1:integer_shock_time_vec[1]]
                         , order = garch_order
                        , xreg = X[[1]][1:integer_shock_time_vec[1],covariate_indices]
                         , backcast.values = NULL
                         , control = list(eval.max = 100000
                                          , iter.max = 1500000
                                          , rel.tol = 1e-8))

  cat('\n================================================================\n')
  print('Outputting the fitted GARCH for the time series under study.')
  print(fitted_garch)
  cat('\n================================================================\n')

  #Note: for forecasting, we use last-observed X value
  X_to_use_in_forecast <- X[[1]][integer_shock_time_vec[1],covariate_indices]

  X_replicated_for_forecast_length <- matrix(rep(X_to_use_in_forecast, k)
                                             , nrow = shock_length_vec[1]
                                             , byrow = TRUE)

 forecast_period <- (integer_shock_time_vec[1]+1):(integer_shock_time_vec[1]+shock_length_vec[1])
  mat_X_for_forecast <- cbind(Y_series_list[[1]][forecast_period]
                         , X_replicated_for_forecast_length)

  unadjusted_pred <- predict(fitted_garch
                             , n.ahead = shock_length_vec[1]
                             , newxreg = mat_X_for_forecast[,-1])
}

adjusted_pred <- unadjusted_pred + rep(omega_star_hat, k)

arithmetic_mean_based_pred <- rep(mean(omega_star_hat_vec), k) + unadjusted_pred

if (is.null(ground_truth_vec) == TRUE){
  QL_loss_unadjusted_pred <- NA
  QL_loss_adjusted_pred <- NA
```

```
}
else {
  QL_loss_unadjusted_pred <- sum(QL_loss_function(unadjusted_pred, ground_truth_vec))
  QL_loss_adjusted_pred <- sum(QL_loss_function(adjusted_pred, ground_truth_vec))
}


list_of_linear_combinations <- list(w_hat)
list_of_forecasts <- list(unadjusted_pred, adjusted_pred)
names(list_of_forecasts) <- c('unadjusted_pred', 'adjusted_pred')

output_list <- list(list_of_linear_combinations
                    , list_of_forecasts)

names(output_list) <- c('linear_combinations', 'predictions')

## tk OUTPUT
cat('--------------------------------------------------------------\n',
    '-------------------SynthVolForecast Results-------------------','\n',
    '--------------------------------------------------------------\n',
    'Donors:', n, '\n',  '\n',
    'Shock times:', shock_time_vec, '\n', '\n',
    'Lengths of shock times:', shock_length_vec, '\n', '\n',
    'Optimization Success:', dbw_output[[2]], '\n', '\n',
    'Convex combination:',w_hat,'\n', '\n',
    'Shock estimates:', omega_star_hat_vec, '\n', '\n',
    'Aggregate estimated shock effect:', omega_star_hat, '\n', '\n',
    'Unadjusted Forecast:', unadjusted_pred,'\n', '\n',
    'Adjusted Forecast:', adjusted_pred,'\n', '\n',
    'Arithmetic-Mean-Based Forecast:',arithmetic_mean_based_pred,'\n','\n',
    'Ground Truth (estimated by realized volatility):', ground_truth_vec,'\n', '\n',
    'QL Loss of unadjusted:', QL_loss_unadjusted_pred,'\n', '\n',
    'QL Loss of adjusted:', QL_loss_adjusted_pred,'\n', '\n'
)

## PLOTS

if (plots == TRUE){
  cat('\n User has opted to produce plots.','\n')
  plot_maker_garch(fitted(fitted_garch)
             ,shock_time_labels
             ,integer_shock_time_vec
             ,shock_length_vec
             ,unadjusted_pred
             ,w_hat
             ,omega_star_hat
             ,omega_star_hat_vec
             ,adjusted_pred
             ,arithmetic_mean_based_pred
             ,ground_truth_vec)
                }

return(output_list)
```

```
} ### END SynthVolForecast
```

Index