Package: GARCH.X (via r-universe)

May 24, 2026

Title Estimation and Exogenous Covariate Selection for ARCH-m(X), Additive ARCH-m(x), and GARCH-X Models

Version 2.0

Description Estimates the parameters and nonparametric functions of an ARCH-m(X) model with exogenous covariates, estimates the parameters and nonparametric functions of an Additive ARCH-m(X) model with exogenous covariates, estimates the parameters of a GARCH-X model with exogenous covariates, performs hypothesis tests for the covariates returning the p-values, and performs stepwise variable selection on the exogenous covariates, and uses False Discovery Rate p-value corrections to select the exogenous variables.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.3

Imports stats, methods, nnls, utils, GA, GenSA, pso, KernSmooth

NeedsCompilation no

Author Adriano Zambom [aut, cre], Vincent Alegrete [aut], Elijah Sagaran [aut], Avni Israni [aut]

Maintainer Adriano Zambom <adriano.zambom@csun.edu>

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

Date/Publication 2026-04-21 17:31:12 UTC

RemoteUrl https://github.com/cran/GARCH.X

RemoteRef HEAD

RemoteSha 044d8fd21d09db1c8ea7bf05715f6785448c2683

Contents

`ARCHmX` *Fitting an ARCH-m(X) model*

Description

Fits an ARCH-m(X) model with given data and estimates the alphas and the nonparametric function m(X).

Usage

```
ARCHmX(
  eps,
  X,
  p,
  delta=2,
  covariates="all",
  hypothesisTest=TRUE,
  bandwidth = "plugin",
  k.choice = "plugin"
)
```

Arguments

`eps`	Time series
`X`	Matrix with exogenous covariates where the number of rows is equal to the length of eps and columns represent the covariates.
`p`	Order of the ARCH-m(X) model. Value of p cannot be 0.
`delta`	Value of the power of the main time series for a POWER-GARCH, default is 2.
`covariates`	Instead of slicing X when building partial models, you can specify a vector of column indices. Defaults to "all".
`hypothesisTest`	Whether to perform covariate hypothesis testing, default is TRUE.
`bandwidth`	Choice of bandwidth for the Nadaraya-Watson Kernel regression estimation. Options are "plugin" or "CV". Default is "plugin".
`k.choice`	Choice of window size k for the ANOVA hypotghesis test. Options are "plugin" or "CV". Default is "plugin".

Details

Performs semiparametric estimation of $\alpha_1, \cdots, \alpha_p$ and $m(X_t)$ for the ARCH-m(X) model

$$\epsilon_t = z_t \sigma_t,$$

$$\sigma_t^2 = \sum_{j=1}^{p} \alpha_j \epsilon_{t-j}^2 + m(\mathbf{X}_{t-1}).$$

Under the model, it can be shown that

$$\mathbb{E}\left[\epsilon_t^2 - \sum_{j=1}^{p} \alpha_j \epsilon_{t-j}^2\right] = m(\mathbf{X}_{t-1}).$$

A Nadaraya-Watson kernel is used to estimate $m(\mathbf{x}_{t-1})$ as

$$\hat{m}(\mathbf{x}_{t-1}) = \hat{m}(\mathbf{x}_{t-1}, \alpha) = \sum_{\ell=p+1}^{T} W_\ell(\mathbf{x}_{t-1}) \left(\epsilon_\ell^2 - \sum_{j=1}^{p} \alpha_j \epsilon_{\ell-j}^2\right).$$

This gives rise to a residual sum of squares

$$\text{RSS} = \sum_{t=p+1}^{T} \left(\tilde{\epsilon}_{t,0}^2 - \sum_{k=1}^{p} \alpha_k \tilde{\epsilon}_{t,k}^2\right)^2 = \|\tilde{\mathbf{Y}} - \tilde{\mathbf{X}}\alpha\|^2$$

where

$$\tilde{\epsilon}_{t,k}^2 = \epsilon_{t-k}^2 - \sum_{\ell=p+1}^{T} W_\ell(\mathbf{x}_{t-1}) \epsilon_{\ell-k}^2.$$

Non-negative least squares regression is used to obtain the final estimates of $\alpha_1, \cdots, \alpha_p$.

No variable selection is done in this function, but an ANOVA-type hypothesis test may be performed for the significance of each covariate. This is enabled by default.

Value

An object of class ARCHmX

Examples

```
# We simulate a time series as in the simulate_ARCHmX example.
set.seed(1)
n <- 2000
alpha <- c(0.01, 0.02, 0.03)
valinit <- 200
X <- matrix(rnorm((n + valinit) * 5), ncol = 5)

m <- function(X) {
  abs(X[, 1])
}
```

```
data <- sim.ARCHmX(n = n, alpha = alpha, m = m, X = X, delta = 2, valinit = valinit)

p <- length(alpha)

# Then we fit the model
model <- ARCHmX(eps = data$eps, X = data$X, p=p)

# Model summary shows Covariate 1 significant at the <0.001 level.
summary(model)
```

`ARCHmXAdditive` *Creating and fitting an Additive ARCH-m(X) model for variable selection*

Description

Fits an Additive ARCH-m(X) model with given data and estimates the alphas and nonparametric functions m(X).

Usage

```
ARCHmXAdditive(
  eps,
  X,
  p,
  delta=2,
  covariates="all",
  method="smooth_means_Positive_Constraint_Exp",
  train_end = NULL,
  n_burn = 10,
  max_outer_iter = 30,
  tol = 1e-4,
  hypothesisTest=TRUE,
  k.choice = "plugin"
)
```

Arguments

`eps`	Time series.
`X`	Matrix with exogenous covariates where the number of rows is equal to the length of eps and columns represent the covariates
`p`	Order of the Additive ARCH-m(X) model. Value of p cannot be 0.
`delta`	Value of the power of the main time series for a POWER-GARCH. Default is 2.
`covariates`	Instead of slicing X when building partial models, you can specify a vector of column indices. Defaults to "all".

`method`	Backfitting strategy used for the additive nonparametric component. One of `"smooth_means_Positive_Constraint_Exp"`, `"smooth_means_lowes"`, or `"smooth_variance"`.

`"smooth_means_Positive_Constraint_Exp"`	Smooth log-partial-residuals and exponentiate (log/exp positivity constraint)
`"smooth_means_lowes"`	Smooth partial residuals and shift upward by subtracting the minimum (smo
`"smooth_variance"`	Smooth a local weighted variance curve of partial residuals, take a square ro

`train_end`	Optional integer. If provided, fit on `1:train_end` and compute test metrics on the remainder. If not provided, fit on the full sample.
`n_burn`	Integer burn-in. Iterations and fitted quantities are evaluated only after burn-in has passed. Default is 10.
`max_outer_iter`	Maximum number of outer backfitting iterations. Default is 30.
`tol`	Convergence tolerance for the outer iterations. Default is `1e-4`.
`hypothesisTest`	Whether to perform covariate hypothesis testing, default is TRUE.
`k.choice`	Choice of window size k for the ANOVA hypotghesis test. Options are "plugin" or "CV". Default is "plugin".

Details

This function fits an additive semiparametric volatility model of the form

$$\epsilon_t = \sigma_t w_t,$$

$$\sigma_t^2 = \sum_{i=1}^{p} \alpha_i |\epsilon_{t-i}|^2 + \sum_{\ell=1}^{d} m_\ell(X_{t-1,\ell}).$$

Estimation proceeds by iterating between:

1. Updating the ARCH parameters $\alpha_1, \ldots, \alpha_p$ (subject to nonnegativity constraints), and
2. Updating the additive components m_ℓ using backfitting with a positivity-enforcing strategy controlled by `method`.

The outer loop stops when the change in the objective (or updates) is below `tol` or when `max_outer_iter` is reached.

No variable selection is done in this function, but an ANOVA-type hypothesis test may be performed for the significance of each covariate. This is enabled by default.

Value

An object of class ARCHmXAdditive

Examples

```
# We simulate a time series as in the simulate.ARCHmXAdditive example.
set.seed(1)
n <- 2000
alpha <- c(0.2, 0.1)
valinit <- 200
X <- matrix(rnorm((n+valinit) * 3), ncol = 3)

m <- list(
  function(x) 4 * abs(sin(x)),
  function(x) 2 * x^2,
  function(x) 1.5
)

data <- sim.ARCHmXAdditive(n = n, alpha = alpha, X = X, m = m)
p <- length(alpha)

# Then we fit the model
model <- ARCHmXAdditive(X = data$X, p = p, eps = data$eps,
                        method = "smooth_means_lowes", max_outer_iter = 15)
model$alphas
model$converged

# We inspect the model summary
summary(model)
```

GARCHX *Fitting a GARCH-X model*

Description

Fits a GARCHX model with given data and estimates the coefficients for omega, alpha, beta, and gamma

Usage

```
GARCHX(
  eps,
  X,
  order = c(1, 1),
  delta = 2,
  covariates = "all",
  optim.method = "NR"
)
```

Arguments

eps	Time series
X	Matrix with exogenous covariates where the number of rows is equal to the length of eps and columns represent the covariates.
order	Order of the GARCH model. Value of p cannot be 0.
delta	Value of the power of the main time series to allow for Power GARCHX, default is 2 for GARCHX.
covariates	Instead of slicing X when building partial models, you can specify a vector of column indices. Defaults to "all".
optim.method	Optimization method for maximizing quasi-likelihood function. Options: "NR", "L-BFGS-B", "GA", "PS", "SA". Default value is "NR".

Details

Uses the GARCHX model

$$\mathcal{E}_t = \sigma_t w_t$$

$$\sigma_t^2 = \omega_0 + \sum_{i=1}^{p} \alpha_i \mathcal{E}_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 + \gamma^T \mathbf{x}_{t-1}$$

To estimate the coefficients for

$$\omega, \alpha, \beta, \gamma$$

.

Value

An object of class GARCHX

Examples

```
# We simulate a time-series as in the simulate.GARCHX example.
set.seed(1)
n <- 200
d <- 4
valinit <- 100
n2 <- n + d + 1
omega <- 0.05
alpha <- 0.05
beta <- 0.05
delta <- 2
gamma <- rep(0.05, d)
e<-rnorm(n2+valinit)
Y<-e
for (t in 2:n2)
 Y[t]<- 0.2*Y[t-1]+e[t]
x<-exp(Y)
X <- matrix(0, nrow = (n+valinit), ncol = length(gamma))
for(j in 1:d)
```

```
 X[, j] <- x[(d+2-j):(n+d+1-j+valinit)]
data <- sim.GARCHX(n = n, omega = omega, alpha = alpha, beta = beta,
                   delta = delta, X = X, gamma = gamma, valinit = valinit)

# Then we fit the model
model <- GARCHX(data$eps, data$X, order = c(1, 1), delta = 2,
                covariates = "all", optim.method = "NR")

# We inspect the model summary
summary(model)
```

`sim.ARCHmX` *Simulating data from an ARCH-m(X) process*

Description

Returns a sample ARCH-m(X) time series given αs, data X, and a function $m(X)$ provided by the user. A vector of simulated ϵs will be returned for every row in X.

Usage

```
sim.ARCHmX(n,
  alpha,
  m,
  X,
  delta=2,
  shock.distr = "Normal",
  valinit=200
)
```

Arguments

`n`	Desired length of simulated time series data
`alpha`	Set of parameters which determines the model's p-step time dependence.
`m`	The function to apply to X. m must accept the matrix X and produce a vector of outputs for every row of X.
`X`	Matrix with exogenous covariates where each row is an observation. Must be compatible with m
`delta`	Value of the power of the main time series for a POWER-GARCH, default is 2
`shock.distr`	The distribution from which the epsilons will be simulated. Default is 'Normal', and must be one of 'Normal', 't5', 't7', or 'Laplace'.
`valinit`	How many cycles the simulation should spend warming up before returning samples. Default is 200

Details

The user's specified αs, δ, function $m(X)$, data X, and shock distribution Z partially define an ARCH-m(X) time series

$$\begin{cases} \epsilon_t = z_t \sigma_t \\ \sigma_t^\delta = \sum_{j=1}^{p} \alpha_j \epsilon_{t-j}^\delta + m(\mathbf{X}_{t-1}) \end{cases}$$

To get the ϵ_t simulations off the ground, a series of burn-in ϵs are generated and used to build each σ_t^δ. From here, the shock distribution is sampled to produce each ϵ_t.

Value

A vector of simulated ϵs corresponding to each observation (row) in X.

Examples

```
set.seed(1)
n <- 2000
alpha <- c(0.01, 0.02, 0.03)
valinit <- 200
X <- matrix(rnorm((n + valinit) * 5), ncol = 5)

m <- function(X) {
  abs(X[, 1])
}

data <- sim.ARCHmX(n = n, alpha = alpha, m = m, X = X, delta = 2, valinit = valinit)

summary(data$eps)
plot(data$eps, type = "l", main = "Simulated ARCH-m(X) series")
```

---

sim.ARCHmXAdditive *Simulate data from an ARCHmXAdditive model*

Description

Returns a sample Additive ARCH-m(X) time series given αs, data X, and list of functions `m` provided by the user. A vector of simulated ϵs will be returned for every row in X.

Usage

```
sim.ARCHmXAdditive(n,
  alpha,
  m,
  X,
  delta = 2,
  valinit = 200,
  shock.distr = "Normal"
)
```

Arguments

n	Desired length of simulated time series data.
alpha	Set of parameters which determines the model's p-step time dependence.
m	List of length ncol(X) containing univariate functions $m_1, \ldots, m_d$. Each m[[ell]] must accept a single numeric input and return a single finite nonnegative numeric value. The additive term is $\sum_{\ell=1}^{d} m_\ell(X_{t-1,\ell})$.
X	Matrix with exogenous covariates where each row is an observation. Must be compatible with m.
delta	Value of the power of the main time series for a POWER-GARCH, default is 2.
valinit	Integer burn-in length used to stabilize the recursion before returning the simulated sample. Default is 200.
shock.distr	Innovation (shock) distribution for z_t. One of "Normal", "t5", "t7", or "Laplace".

Value

A vector of simulated ϵs corresponding to each observation (row) in X.

Examples

```
set.seed(1)
n <- 2000
alpha <- c(0.2, 0.1)
valinit <- 200
X <- matrix(rnorm((n+valinit) * 3), ncol = 3)

m <- list(
  function(x) 4 * abs(sin(x)),
  function(x) 2 * x^2,
  function(x) 1.5
)

data <- sim.ARCHmXAdditive(n = n, alpha = alpha, X = X, m = m)
summary(data$eps)
plot(data$eps, type = "l", main = "Simulated additive ARCH-m(X) series")
```

---

sim.GARCHX *Simulate GARCHX model*

Description

Simulates Time series data from GARCH model with exogenous covariates

Usage

```
sim.GARCHX(n,
  omega,
  alpha,
  beta,
  gamma,
  X,
  delta = 2,
  shock.distr = "Normal",
  valinit = 200
)
```

Arguments

n	Desired length of simulated time series data.
omega	Coefficient value for omega, required to be $\omega_0 > 0$
alpha	ARCH Coefficient value, required to be $\alpha_0 \geq 0$
beta	GARCH Coefficient value, required to be $\beta_0 \geq 0$
gamma	Vector containing coefficients for exogenous covariates.
X	Matrix with exogenous covariates where the number of rows is equal to the length of n + valinit
delta	Value of the power of the time series to allow for Power GARCHX, default is 2 for GARCHX
shock.distr	Distribution of the shock eta_t that multiply w_t in the GARCH-X model eps_ = w_t*eta_t.
valinit	Initialization value, default value is 200

Value

A named list containing vector of Time Series data and X covariates used

Examples

```
set.seed(1)
n <- 200
d <- 4
valinit <- 100
n2 <- n + d + 1
omega <- 0.05
alpha <- 0.05
```

```
beta <- 0.05
delta <- 2
gamma <- rep(0.05, d)
e<-rnorm(n2+valinit)
Y<-e
for (t in 2:n2)
 Y[t]<- 0.2*Y[t-1]+e[t]
x<-exp(Y)
X <- matrix(0, nrow = (n+valinit), ncol = length(gamma))
for(j in 1:d)
 X[, j] <- x[(d+2-j):(n+d+1-j+valinit)]
data <- sim.GARCHX(n = n, omega = omega, alpha = alpha, beta = beta,
                   delta = delta, X = X, gamma = gamma, valinit = valinit)
```

`stepwise` *Performing Stepwise Variable Selection for an ARCH-m(X) or GARCHX Process*

Description

Returns the fitted ARCH-m(X), additive ARCH-m(X), or GARCHX model resulting from the specified stepwise selection algorithm.

Usage

```
stepwise(
  target_model,
  direction="forward",
  criterion="pvalue",
  adjust_method="fdr",
  alpha.level=0.05,
  trace = TRUE
)
```

Arguments

`target_model`	A model object of class `ARCHmX`, `ARCHmXAdditive`, or `GARCHX`. This model must be fit on all available covariates and, in the case of `ARCHmX` and `ARCHmXAdditive`, the covariate hypothesis testing must have been set to `TRUE`. This model defines all the relevant attributes of the candidate models the stepwise selection process will consider.
`direction`	The mode of stepwise search, can be one of "forward", "backward", "both_null" (both directions starting with the empty model), "both_full" (both directions starting with the full model), "bestsubset", and "onepass", with a default of "forward".
`criterion`	The criterion for taking additional steps, default is "pvalue".
`adjust_method`	Correction for multiple testing accumulation of error. See p.adjust.

alpha.level	Defines the adjusted p-value significance level, default is 0.05.
trace	If 'TRUE', information is printed for each step of variable selection. Default is 'FALSE'. Offers summaries of prospective models as each predictor in the scope is added to or removed from the model. 'max_pvalue' indicates the maximum p-value from the multiple tests for each predictor in the model.

Details

The forward selection routine starts with a (null or empty) ARCH model. At each iteration, the current model is compared to an array of all possible models with one additional covariate. A forward step is taken to the candidate model with all significant covariates providing the most improvement in the specified criterion. If no such model exists, or if all covariates are exhausted, the current model is returned.

The backward selection routine starts with the full ARCH-m(X), additive ARCH-m(X), or GARCHX model with all possible covariates. At each iteration, the current model is compared to an array of all possible models with one covariate removed. A backward step is taken to the candidate model providing the most improvement in the specified criterion until a model is found with all significant covariates, or until the logic reaches the empty model.

The both_null and both_full selection routines start with the null and full models, respectively. At each iteration, both forward and backward steps are attempted as outlined above. This continues until no step is taken, and the current model will be returned. The pvalue criterion can not be used for these routines since the p-value contstraints for forward and backward selection are incompatible with one another.

The best_subset selection routine will build models for every possible combination of covariates and return the one with the best value for the specified criterion.

The onepass selection routine builds the full model and returns a new model containing every covariate identified as significant within the full model. The only criterion available for onepass is pvalue.

The user may specify p-value corrections to apply to the p-values used within the desired stepwise selection routine. All options provided by p.adjust are exposed to the user; the default is set to fdr.

Value

The final model chosen by the stepwise-selection process.

Examples

```
# We simulate a time series as in the simulate_ARCHmX example.
set.seed(1)
n <- 2000
alpha <- c(0.01, 0.02, 0.03)
valinit <- 200
X <- matrix(rnorm((n + valinit) * 5), ncol = 5)

m <- function(X) {
  abs(X[, 1])
}
```

```
data <- sim.ARCHmX(n = n, alpha = alpha, m = m, X = X, delta = 2, valinit = valinit)
p <- length(alpha)

# Output will show the trace from the empty model to the model with Covariate 1,
# which is returned as stepwise_model.
target_ARCHmX_model <- ARCHmX(X = data$X, p = p, eps = data$eps)
stepwise_model <- stepwise(target_ARCHmX_model, direction="forward", criterion="pvalue")
```

Index