Package: tvGarchKF (via r-universe)

June 10, 2026

Date 2025-05-06

Type Package

Title Time-Varying Garch Models Through a State-Space Representation

Version 0.0.1

Maintainer Tomás Arancibia <tarancibia2016@udec.cl>

Description Estimates the time-varying (tv) parameters of the GARCH(1,1) model, enabling the modeling of non-stationary volatilities by allowing the model parameters to change gradually over time. The estimation and prediction processes are facilitated through the application of the Kalman filter and state-space equations. This package supports the estimation of tv parameters for various deterministic functions, which can be identified through exploratory analysis of different time periods or segments of return data. The methodology is grounded in the framework presented by Ferreira et al. (2017) <doi:10.1080/00949655.2017.1334778>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

LinkingTo Rcpp

Imports stats, fGarch, graphics

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Repository https://tomas-udec.r-universe.dev

Date/Publication 2025-06-01 17:59:20 UTC

RemoteUrl https://github.com/tomas-udec/tvgarchkf

RemoteRef HEAD

RemoteSha ed8db55674b6cfa6c14e2bb6576594bd6f5c5c68

Contents

`indipsa` *Selective Stock Price Index*

Description

The data covers the period from March 2000 to October 2014, totaling 3186 observations.

Usage

```
indipsa
```

Format

A time series object with 3186 elements

It's a stock market index that tracks the performance of a select group of Chilean companies.

Source

Yahoo Finance

`tvGarch_Sim` *Generating Simulations using a tv-Garch Model*

Description

Simulate from a tv-Garch(1,1) model.

Usage

```
tvGarch_Sim(
  n,
  gamma,
  alpha,
  beta,
  type = c("polynomial", "NoLineal", "trigonometric"),
  exponentes = NULL,
  trig = NULL,
  arg = NULL
)
```

Arguments

n	integer
gamma	Vector containing coefficents of c.
alpha	Vector containing coefficents of alpha.
beta	Vector containing coefficents of beta.
type	Vector of function type for c, alpha and beta.
exponentes	Vector for exponenets in NoLineal.
trig	Type of trigonometric function.
arg	Value of argument for the trigonometric function.

Value

An object of class 'zoo' with two components: the first component represents returns, while the second component denotes conditional variance.

Examples

```
## Simulate from a tv-GARCH(1,1) model lineal:
alpha_sim <- c(0.2, 0.2)
beta_sim <- c(0.45, 0.5, -0.85)
type_sim <- c("polynomial","polynomial","polynomial")
Sim1 <- tvGarch_Sim(n = 6000, gamma = 0.1, alpha = alpha_sim, beta = beta_sim, type = type_sim)
plot(Sim1[,1], type="l", main="Simulated tvGARCH(1, 1) process",
    ylim=c(-max(Sim1[,2]), max(Sim1[,2])))
lines(Sim1[,2], type="l", col="red")
legend("topright",legend=c("tvGARCH(1,1)",expression(sigma(u))),
      col=c("black","red"),lty=1,bty="n",lwd=1)
## Simulate from a tv-GARCH(1,1) model non linear:
alpha_sim2 <- c(0.75, 0.08)
beta_sim2 <- c(0.05, 0.03, 0.06)
type_sim2 <- c("polynomial","polynomial","NoLineal")
expo <- c(0, 1, 1/2)
Sim2<-tvGarch_Sim(n=6000,gamma=0.05,alpha=alpha_sim2,beta=beta_sim2,type=type_sim2,exponentes=expo)
plot(Sim2[,1], type="l", main="Simulated tvGARCH(1, 1) process",
     ylim=c(-max(Sim2[,2]),max(Sim2[,2])))
```

```
lines(Sim2[,2], type="l", col="red")
legend("topright",legend=c("tvGARCH(1,1)",expression(sigma(u))),
       col=c("black","red"),lty=1,bty="n",lwd=1)
```

`tvGarchKalmanFit` *Fit the time-varying (Tv) parameters of the GARCH model (tv-Garch) by using the Kalman Filter method. The tv-parameters are determined by deterministic functions of either linear or non-linear type.*

Description

The tv-Garch(1,1) model, the parameters vary slowly over time according to linear or non-linear functions. These parameters are denoted by $c(t)$, $\alpha(t)$ and $\beta(t)$ which correspond to the model $\sigma_t = c(t) + \alpha(t) r_{t-1}^2 + \beta(t)\sigma_{t-1}$.

Usage

```
tvGarchKalmanFit(
  series,
  c,
  alpha,
  beta,
  type = c("polynomial", "NoLineal", "trigonometric"),
  exponentes,
  trig,
  arg,
  predict = 0,
  trace.log = FALSE
)
```

Arguments

`series`	Time series.
`c`	Vector containing coefficents of c.
`alpha`	Vector containing coefficents of alpha.
`beta`	Vector containing coefficents of beta.
`type`	Vector of function type for c, alpha and beta.
`exponentes`	Vector for exponenets in NoLineal.
`trig`	Type of trigonometric function.
`arg`	Value of argument for the trigonometric function.
`predict`	Value for time to generate predict.
`trace.log`	Variable to print names of coefficients.

Details

The types of functions for the tv-parameters are: linear, non-linear, trigonometric, and exponential. For the case of the linear model, the tv-parameters follow the following structure:

$$c(t) = c_0 + c_1 u + c_2 u^2 + \ldots + c_p u^p,$$

$$\alpha(t) = a_0 + a_1 u + a_2 u^2 + \ldots + a_p u^p,$$

$$\beta(t) = b_0 + b_1 u + b_2 u^2 + \ldots + b_p u^p,$$

where $u = t/T$, with $t = 1, 2, \ldots, T$. For the non-linear case, it is as follows:

$$c(t) = c_0 + \sum_{j=1}^{k} c_j u_{c,j},$$

$$\alpha(t) = a_0 + \sum_{j=1}^{k} a_j u_{\alpha,j},$$

$$\beta(t) = b_0 + \sum_{j=1}^{k} b_j u_{\beta,j},$$

where k it is positive value and $u_{c,j}$, $u_{\alpha,j}$ and $u_{\beta,j}$ are non linear function set. For the trigonometric case, it is as follows:

$$c(t) = c_0 + c_1 g(u),$$

$$\alpha(t) = a_0 + a_1 g(u),$$

$$\beta(t) = b_0 + b_1 g(u),$$

where $g(u)$ it is a trigonometric function, cos or sin.

Value

Return fit values of omega, alpha and beta

Examples

```
ipsa<-diff(log(indipsa))
c <- c(0.05,0.05)
alpha <- c(0.05,0.05)
beta <- c(0.05,0.05)
type_fit <- c("trigonometric","trigonometric","trigonometric")
fit<-tvGarchKalmanFit(ipsa,c=c,alpha=alpha,beta=beta,type=type_fit,trig="cos",arg="3*(1-log(u))")
```

`tvGarchKalmanLoglike` *Models tv-Garch Filter Kalman LogLikehood.*

Description

It is the function to use in the process to fit coefficients.

Usage

```
tvGarchKalmanLoglike(
  x,
  series,
  c,
  alpha,
  beta,
  nsample = length(series),
  type = c("polynomial", "NoLineal", "trigonometric"),
  exponentes,
  trig,
  arg,
  predict,
  trace.log = FALSE
)
```

Arguments

`x`	Vector of coefficents to fit.
`series`	Time series.
`c`	Vector containing coefficents of c.
`alpha`	Vector containing coefficents of alpha.
`beta`	Vector containing coefficents of beta.
`nsample`	Value of time series length.
`type`	Vector of function type for c, alpha and beta.
`exponentes`	Vector for exponenets in NoLineal.
`trig`	Type of trigonometric function.
`arg`	Value of argument for the trigonometric function.
`predict`	Value for time to generate predict.
`trace.log`	Variable to print names of coefficients.

Value

Value of loglike in model.

`tvGarchKalmanPrint` *Models tv-Garch Filter Kalman print outputs.*

Description

This function is designed to print the outputs of the tv-Garch model, which include the returns, conditional variance, log-likelihood value, and mean squared error (MSE).

Usage

```
tvGarchKalmanPrint(
  x,
  series,
  c,
  alpha,
  beta,
  nsample = length(series),
  type = c("polynomial", "NoLineal", "trigonometric"),
  exponentes,
  trig,
  arg,
  trace.log = FALSE,
  predict
)
```

Arguments

`x`	Vector of coefficents to fit.
`series`	Time series.
`c`	Vector containing coefficents of c.
`alpha`	Vector containing coefficents of alpha.
`beta`	Vector containing coefficents of beta.
`nsample`	Value of time series length.
`type`	Vector of function type for c, alpha and beta.
`exponentes`	Vector for exponenets in NoLineal.
`trig`	Type of trigonometric function.
`arg`	Value of argument for the trigonometric function.
`trace.log`	Variable to print names of coefficients.
`predict`	Value for time to generate predict.

Value

A data frame containing the following columns:

- `X`: State vector of Kalman equations.
- `Fm`: Value of MSE
- `sigma`: Conditional variance.
- `loglike`: Value of the loglike.

Examples

```
data(ipsa)
ipsa<-diff(log(indipsa))
c<-c(0.05,0.05)
alpha<-c(0.05,0.05)
beta<-c(0.05,0.05)
type_fit<-c("trigonometric","trigonometric","trigonometric")
fit<-tvGarchKalmanFit(ipsa,c=c,alpha=alpha,beta=beta,type=type_fit,trig="cos",arg="3*(1-log(u))")
arg_model<-"3*(1-log(u))"
model<-tvGarchKalmanPrint(fit,ipsa,c=c,alpha=alpha,beta=beta,type=type_fit,trig="cos",arg=arg_model)
plot(ipsa,ylab="",xlim=c(2000,2015))
lines(ts(model$sigma, star=2000, freq=225), col="red", lwd=2)
lines(ts(model$sigma*(-1), star=2000, freq=225), col="red", lwd=2)
```

---

tvParameter *Structure of the Time-Varying GARCH(1,1) Parameters*

Description

This function performs an exploratory analysis to uncover the dynamic structure of the time-varying GARCH(1,1) parameters. Specifically, the observation domain $\{1, \ldots, T\}$ is partitioned into M overlapping blocks, each of length N, with a constant shift of size S between consecutive blocks. The relation between these quantities satisfies $T = S(M-1) + N$. The midpoint of the j-th block, for $j = 1, \ldots, M$, is denoted $t_j = S(j-1) + N/2$. For each block, a local estimation of the stationary GARCH(1,1) model is performed using the observations within that block. The resulting sequence of local estimates, evaluated across all blocks, provides an empirical trajectory that reflects the underlying evolution of the time-varying parameters. This trajectory can serve as a guide for selecting flexible function classes capable of capturing their temporal variation.

Usage

```
tvParameter(data, S, N, plot = TRUE)
```

Arguments

data	Represents the financial return series employed to investigate the temporal evolution of the parameters in the tv-GARCH(1,1) model.
S	The number of observations by which the analysis window is shifted to define the starting point of the next block; also known as the step size or shift parameter.
N	The total number of observations contained within each data block, representing the block or window length over which local model estimation is performed.
plot	A Boolean flag indicating whether a graphical representation of the estimation results should be generated.

Value

Data frame who contains omega, alpha, beta of GARCH(1,1) model and midpoint each block.

Examples

```
ipsa<-diff(log(indipsa))*100
S = 100
N = 800
tv <- tvParameter(ipsa,S,N)
```

Index