Package: tvGarchKF (via r-universe)

October 21, 2026

Date 2026-08-13

Type Package

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

Version 1.0.0

Maintainer Tomás Arancibia <tarancibia2016@udec.cl>

Depends R (>= 3.5.0)

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, numDeriv, timeSeries, methods, zoo, utils, locfit, npreg, withr, mgcv, splines

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests testthat (>= 3.5.0)

Config/testthat/edition 3

Collate 'NonParam-BS.R' 'NonParam-LP.R' 'NonParam-SS.R' 'Parametric-function.R' 'auto.tvGarchKF.R' 'class-tvGarchKF.R' 'class-tvGarchKFNonParam.R' 'class-tvGarchKFNonParam_Fit.R' 'helpers_gamSS.R' 'helpers.R' 'indipsa.R' 'methods-fitted.R'

'methods-nobs.R' 'methods-plot.R' 'methods-predict.R'
'methods-residuals.R' 'methods-summary.R' 'methods-vcov.R'
'tvCoef.R' 'tvGarchKalman.R' 'tvGarchKalmanFit.R'
'tvGarchKalmanLoglike.R' 'tvGarchKalmanPrint.R'
'tvGarchNonParamFit.R' 'tvGarchNonParamKF.R' 'tvGarch_Sim.R'
'tvParameter.R'

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

Date/Publication 2026-08-22 16:53:47 UTC

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

RemoteRef HEAD

RemoteSha 08ebfa907aec9941233b481f81d7689fe59082e9

Contents

auto.tvGarchKF 2
fitted,tvGarchKF-method 5
indipsa 5
nobs,tvGarchKF-method 6
plot,tvGarchKF,ANY-method 6
predict,tvGarchKF-method 11
residuals,tvGarchKF-method 13
summary,tvGarchKF-method 14
tvGarch_Sim 14
tvGarchKalmanFit 16
tvGarchKalmanLoglike 18
tvGarchKalmanPrint 19
tvGarchKF-class 20
tvGarchKFNonParam-class 21
tvGarchKFNonParam_Fit-class 22
tvGarchNonParamFit 23
tvGarchNonParamKF 25
tvParameter 26
vcov,tvGarchKF-method 28

Index **29**

`auto.tvGarchKF` *Automatic selection of tuning parameters for non-parametric tv-GARCH estimation*

Description

This function provides an automatic procedure for selecting tuning parameters for non-parametric estimation of the tv-GARCH parameter curves. Generalized additive models (GAMs) are used as reference smoothers to estimate the effective degrees of freedom (EDF) associated with each time-varying parameter curve. The resulting EDF values are then used as a common complexity criterion to calibrate comparable tuning parameters for smoothing spline, local polynomial and B-spline estimators.

Usage

```
auto.tvGarchKF(
  object,
  data,
  k_grid = 3:8,
  bs_gam = "ps",
  gam_method = "REML",
  spar_grid = seq(0.3, 1.3, by = 0.01),
  span_grid = seq(0.5, 2.5, by = 0.05),
  loess_degree = c(1, 2, 2),
  bs_degree_grid = c(2, 2, 2),
  tol = 0.001
)
```

Arguments

`object`	An object class "tvGarchKF" containing the initial estimates of the time-varying GARCH parameters obtained from the local estimation procedure.
`data`	A numeric vector containing the observed time series used to fit the tv-GARCH model.
`k_grid`	An integer vector specifying the candidate basis dimensions (k) evaluated when fitting the GAM reference models. The default is 3:8.
`bs_gam`	A character string indicating the spline basis used in the GAM reference models. The default "ps" corresponding to penalized B-splines.
`gam_method`	A character string specifying the criterion used to estimate the smoothing parameters of the GAM reference models. The default is "REML". Other available choices include "ML" and "GCV.Cp".
`spar_grid`	A numeric vector containing candidate smoothing parameters used to select the smoothing spline whose effective degrees of freedom most closely match those of the corresponding GAM reference models. The default is seq(0.3, 1.3, by = 0.01).
`span_grid`	A numeric vector containing candidate span values used to select the local polynomial smoother whose effective degrees of freedom most closely match those of the corresponding GAM reference models. The default is seq(0.5, 2.5, by = 0.05).
`loess_degree`	An integer vector specifying the polynomial degree used for the local polynomial approximation of each time-varying parameter curve. The elements correspond, respectively, to $c(u)$, $\alpha(u)$ and $\beta(u)$. The default is c(1,2,2).

`bs_degree_grid` An integer vector specifying the degree of the B-spline basis used for each time-varying parameter curve. The elements correspond, respectively, to $c(u)$, $\alpha(u)$ and $\beta(u)$. The default is c(2,2,2).

`tol` A positive numeric value specifying the tolerance used in the EDF-matching procedure. The default is 0.001.

Details

The automatic selection procedure first fits a GAM reference model to each initial time-varying parameter curve and extracts its effective degrees of freedom. These EDF values provide a common measure of curve complexity. For each non-parametric estimator, the function then searches over the corresponding tuning-parameter grid and selects the specification whose effective degrees of freedom most closely match those of the GAM reference model.

Consequently, the smoothing spline, local polynomial and B-spline estimators are calibrated to have approximately comparable levels of smoothness, despite being bases on different smoothing mechanisms.

Value

An object of class "auto.tvGarchKF" containing the fitted GAM reference models and the tuning parameters automatically selected for smoothing spline, local polynomial, and B-spline estimators.

The returned object includes the estimated effective degrees of freedom (edf), smoothing parameters (spar), local polynomial spans (span), polynomial and spline degrees (degree), basis dimensions (k), numbers of knots (nknots), and measures of agreement between the effective degrees of freedom of each non-parametric estimator and those of the corresponding GAM reference model.

The selected tuning parameters are organised by estimation method and time-varying parameter curve, c(u), $\alpha(u)$ and $\beta(u)$, and can subsequently be supplied to `tvGarchNonParamFit()` for non-parametric estimation of the time-varying GARCH parameter curves.

Examples

```
library("tvGarchKF")
n = 2000
seed = 321
type = c( "polynomial","trigonometric","polynomial")
model<-tvGarch_Sim(n=n,c=c(0.18,0.10),alpha=c(0.45,0.4),beta=c(0,1.05,-1.05,0.1),
                      type=type,trig="cos",arg="2*pi*u",seed=seed)
tv_curve<-tvParameter(model[,1],S=120,N=600,plot=TRUE,cex.axis=3,cex.main=3,cex.points=3,
                        mar=c(5,6,4,2),mfrow=c(1,3))
fit_par<-auto.tvGarchKF(object=tv_curve,data=model[,1],k_grid=3:8,bs_gam="ps",gam_method="REML",
                        spar_grid=seq(0.3,1.3,by=0.01),span_grid=seq(0.50,2.50,by=0.05),
                          loess_degree=c(1,2,2), bs_degree_grid=c(2,2,2),tol=1e-3)
```

fitted,tvGarchKF-method *Fitted method.*

Description

Fitted method.

Usage

```
## S4 method for signature 'tvGarchKF'
fitted(object)

## S4 method for signature 'tvGarchKFNonParam'
fitted(object)

## S4 method for signature 'tvGarchKFNonParam_Fit'
fitted(object)
```

Arguments

object	Object class tvGarchKF or tvGarchKFNonParam or tvGarchKFNonParam_Fit.

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

---

nobs,tvGarchKF-method *Number of Observations*

Description

Generic function to extract the number of observations from an object.

Usage

```
## S4 method for signature 'tvGarchKF'
nobs(object, ...)
```

Arguments

object	Object class tvGarchKF.
...	Others.

Value

Number of observations.

---

plot,tvGarchKF,ANY-method

Plot method for tvGarchKF, tvGarchKFNonParam *and* tvGarchKFNonParam_Fit *objects.*

Description

This method provides graphical summaries for fitted tv-GARCH models. By default, it displays the estimated conditional volatility. Alternatively, setting which = "tvparameters" produces a three-panel plot of the estimated time-varying GARCH parameters $c(u)$, $\alpha(u)$ and $\beta(u)$. The function also allows the user to superimpose fitted parametric curves and stationary GARCH benchmarks, and to customize the main graphical elements.

Usage

```
## S4 method for signature 'tvGarchKF,ANY'
plot(
  x,
  y,
  which = c("volatility", "tvparameters"),
  x.index = NULL,
  tvParameter = NULL,
  type = NULL,
  trig = NULL,
```

```
  arg = NULL,
  col.points = "black",
  col.fit = NULL,
  col.stationary = "black",
  lwd.y = 3,
  lwd.fit = 3,
  lwd.stationary = 2,
  lty.fit = 1,
  lty.stationary = 2,
  pch = 20,
  cex.points = 3,
  cex.axis = 2,
  cex.lab = 2,
  cex.main = 2,
  cex.legend = 1.2,
  mfrow = c(1, 3),
  mar = c(5, 5, 4, 2),
  xlab = "Time",
  ylab = "Log-Returns",
  main = NULL,
  legend = TRUE,
  legend.pos = "bottomleft",
  legend.label = NULL,
  bty = "n",
  las = 1,
  ann = FALSE,
  ...
)

## S4 method for signature 'tvGarchKFNonParam_Fit,ANY'
plot(
  x,
  y,
  which = c("volatility", "tvparameters"),
  x.index = NULL,
  tvParameter = NULL,
  type = NULL,
  trig = NULL,
  arg = NULL,
  col.points = "black",
  col.fit = NULL,
  col.stationary = "black",
  lwd.y = 3,
  lwd.fit = 3,
  lwd.stationary = 2,
  lty.fit = 1,
  lty.stationary = 2,
  pch = 20,
```

```
  cex.points = 3,
  cex.axis = 2,
  cex.lab = 2,
  cex.main = 2,
  cex.legend = 1.2,
  mfrow = c(1, 3),
  mar = c(5, 5, 4, 2),
  xlab = "Time",
  ylab = "Log-Returns",
  main = NULL,
  legend = TRUE,
  legend.pos = "bottomleft",
  legend.label = NULL,
  bty = "n",
  las = 1,
  ann = FALSE,
  ...
)

## S4 method for signature 'tvGarchKFNonParam,ANY'
plot(
  x,
  y,
  which = c("volatility", "tvparameters"),
  x.index = NULL,
  tvParameter = NULL,
  type = NULL,
  trig = NULL,
  arg = NULL,
  col.points = "black",
  col.fit = NULL,
  col.stationary = "black",
  lwd.y = 3,
  lwd.fit = 3,
  lwd.stationary = 2,
  lty.fit = 1,
  lty.stationary = 2,
  pch = 20,
  cex.points = 3,
  cex.axis = 2,
  cex.lab = 2,
  cex.main = 2,
  cex.legend = 1.2,
  mfrow = c(1, 3),
  mar = c(5, 5, 4, 2),
  xlab = "Time",
  ylab = "Log-Returns",
  main = NULL,
```

```
  legend = TRUE,
  legend.pos = "bottomleft",
  legend.label = NULL,
  bty = "n",
  las = 1,
  ann = FALSE,
  ...
)
```

Arguments

x	An object class tvGarchKF or tvGarchKFNonParam_Fit or tvGarchKFNonParam.
y	No use.
which	A character string indicating the type of plot. By default "volatility", the function displays the estimated conditional volatility $\widehat{\sigma}_t$. Setting which = "tvparameters" produces a three-panel plot of the estimated time-varying GARCH parameters $c(u)$, $\alpha(u)$, and $\beta(u)$.
x.index	An optional vector defining the horizontal axis. If NULL, the observations are plotted against their sequential index. This argument is useful for displaying the series against calendar dates or another user-defined time scale.
tvParameter	An optional object containing the block estimates of the time-varying GARCH parameters. This argument is used only when which = "tvparameters", in which case the fitted parametric curves are superimposed on the corresponding block estimates.
type	A character vector specifying the functional form assigned to each time-varying parameter. Available specifications include "polynomial", "NoLineal" and "trigonometric".
trig	A character vector indicating the trigonometric function, "cos" or "sin", used when type = "trigonometric".
arg	A character vector specifying the argument of the trigonometric function. This argument is only used when type = "trigonometric".
col.points	Color used for the points representing the estimated time-varying parameters. Default is "black".
col.fit	Color used for the fitted time-varying parameter curves. If NULL, colors are assigned internally.
col.stationary	Color used for the stationary GARCH reference line. Default is "black".
lwd.y	Line width used for the estimated conditional volatility. Default is 3.
lwd.fit	Line width used for the fitted time-varying parameter curves. Default is 3.
lwd.stationary	Line width used for the stationary GARCH reference line. Default is 2.
lty.fit	Line type used for the fitted time-varying parameter curves. Default is 1.
lty.stationary	Line type used for the stationary GARCH reference line. Default is 2.
pch	Plotting symbol used for the points representing the estimated time-varying parameters. Default is 20.
cex.points	Expansion factor controlling the size of the plotted points. Default is 3.

cex.axis	Expansion factor controlling the size of the axis annotations. Default is 2.
cex.lab	Expansion factor controlling the size of the axis labels Default is 2.
cex.main	Expansion factor controlling the size of the main title Default is 2.
cex.legend	Expansion factor controlling the size of the legend text. Default is 1.2.
mfrow	Numeric vector of length two specifying the graphical layout. The default, c(1,3), arranges the plots of $c(u)$, $\alpha(u)$ and $\beta(u)$ in a single row.
mar	Numeric vector specifying the plot margins in the order bottom, left, top and right. Default is c(5,5,4,2).
xlab	Label for the horizontal axis. Default is "Time".
ylab	Label for the vertical axis. Default is "Log-Returns".
main	Optional main title for the plot. If NULL, no user-specified title is added.
legend	Logical. If TRUE, a legend is added to the plot. Default is TRUE.
legend.pos	Position of the legend. Default is "bottomleft".
legend.label	Optional character vector specifying the legend labels. If NULL, default labels are generated internally.
bty	Character specifying the box type. Default is "n", so no box is drawn around the plot.
las	Numeric value controlling the orientation of axis labels. Default is 1, corresponding to horizontal labels.
ann	Logical. If False, the default plot annotation is suppressed.
...	Additional graphical arguments passed to the underlying base R plotting functions.

Examples

```
library("tvGarchKF")
n = 2000
seed = 321
type = c("polynomial","trigonometric","polynomial")
model<-tvGarch_Sim(n=n,c=c(0.18,0.10),alpha=c(0.45,0.4),beta=c(0,1.05,-1.05,0.1),
                     type=type,trig="cos",arg="2*pi*u",seed=seed)
tv_curve<-tvParameter(model[,1],S=120,N=600,plot=TRUE,cex.axis=3,cex.main=3,cex.points=3,
                       mar=c(5,6,4,2),mfrow=c(1,3))
# Parametric
c0     <- c(0.20,0.08)
alpha0 <- c(0.40,0.45)
beta0  <- c(0.00,0.95,-0.95,0.1)
fit<-tvGarchKalmanFit(data=model[,1],c=c0,alpha=alpha0,beta=beta0,type=type,trig="cos",arg="2*pi*u")
plot(fit, which = "volatility", mar = c(5, 4, 4, 2))
plot(fit,which = "tvparameters",tvParameter = tv_curve,
        type = c("linear", "trigonometric", "cubic"),trig = "cos",arg = "2*pi*u")

# Smoothing Spline
FitNonParam_ss<-tvGarchNonParamFit(data=model[,1],tv_curve,smoothing=list(spar=c(0.5,0.68,0.8),
                                   nknots=c(3,3,3)),plot=TRUE,method="Smoothing Spline")
plot(FitNonParam_ss,which = "tvparameters",tvParameter = tv_curve)
```

```
TVparameter_ss<-predict(FitNonParam_ss)
FitNonParamKF_ss<-tvGarchNonParamKF(model[, 1], TVparameter_ss)
plot(FitNonParamKF_ss)

# Local Polynomial
FitNonParam_lp<-tvGarchNonParamFit(data=model[,1],tvparameter=tv_curve,
                                   localPoly=list(span=c(0.70,2.5,1.4),degree=c(1,2,2)),
                                    plot=TRUE,method="Local Polynomial")
plot(FitNonParam_lp,which = "tvparameters",tvParameter = tv_curve)

TVparameter_lp<-predict(FitNonParam_lp)
FitNonParamKF_lp<-tvGarchNonParamKF(model[, 1], TVparameter_lp)
plot(FitNonParamKF_lp)

# B-Spline
FitNonParam_bs<-tvGarchNonParamFit(data=model[,1],tvparameter=tv_curve,
                                     bspline=list(degree=c(2,2,2),df=c(3,4,2)),
                                     plot=TRUE,method="B-Spline")
plot(FitNonParam_bs,which = "tvparameters",tvParameter = tv_curve)

TVparameter_bs<-predict(FitNonParam_bs)
FitNonParamKF_bs<-tvGarchNonParamKF(model[,1],TVparameter_bs)
plot(FitNonParamKF_bs)
```

`predict,tvGarchKF-method`

Forecast conditional volatility from a tv-GARCH model.

Description

This method computes out-of-sample forecasts of the conditional volatility from a fitted tv-GARCH model using the Kalman Filter prediction recursions. Approximate confidence intervals for the forecasted volatility are constructed from the state prediction error variance.

Usage

```
## S4 method for signature 'tvGarchKF'
predict(
  object,
  n.ahead = 10,
  conf = 0.95,
  crit_val = NULL,
  interval = TRUE,
  Plot = TRUE,
  ...
)
```

```
## S4 method for signature 'tvGarchKFNonParam'
predict(object, n.ahead = 10, ...)

## S4 method for signature 'tvGarchKFNonParam_Fit'
predict(
  object,
  n.ahead = 10,
  conf = 0.95,
  crit_val = NULL,
  interval = TRUE,
  Plot = TRUE,
  ...
)
```

Arguments

`object`	An object class `tvGarchKF` or `tvGarchKFNonParam` or `tvGarchKFNonParam_Fit`.
`n.ahead`	A positive integer specifying the number of steps ahead for which the condition volatility is forecast. The default is 10.
`conf`	Confidence level used to construct prediction intervals for the forecasted volatility. The default is 0.95.
`crit_val`	Optional critical value used to construct the confidence intervals. If NULL, the critical value is automatically obtained from the standard normal distribution according to the confidence level specified by `conf`.
`interval`	Logical. If TRUE, confidence intervals for the forecasted conditional volatility are computed and returned. The default is TRUE.
`Plot`	Logical. If TRUE, the forecasted conditional volatility and the corresponding confidence intervals are displayed. The default is TRUE.
`...`	Additional graphical arguments passed to the plot method for objects of class "tvGarchKF_predict".

Value

An object of class "tvGarchKF_predict" containing the forecast of the conditional volatility and the quantities required to asses their uncertainty. The returned object includes:

- `n.ahead`: Forecast horizon.
- `nsample`: Number of observations in the fitted sample.
- `sigma_forecast`: Forecasted conditional volatility ($\widehat{\sigma}_{t+h}$, for $h = 1, \ldots,$ `n.ahead`).
- `se_sigma`: Standard errors of the forecasted conditional volatility.
- `X_forecast`: Predicted state values.
- `P_forecast`: State prediction error variances.
- `method`: Estimation method used in the fitted tv-GARCH model.
- `conf`: Confidence level used to construct intervals.
- `crit_val`: Critical values used to construct the confidence intervals.

- interval: A data frame containing the forecast horizon, forecasted volatility, standard error, and lower and upper confidence limits. This components is included only when interval = TRUE.

If Plot = TRUE, the forecasted volatility and its confidence intervals are also displayed.

Examples

```
library("tvGarchKF")
n = 2000
seed = 321
type = c("polynomial","trigonometric","polynomial")
model<-tvGarch_Sim(n=n,c=c(0.18,0.10),alpha=c(0.45,0.4),beta=c(0,1.05,-1.05,0.1),
                      type=type,trig="cos",arg="2*pi*u",seed=seed)
tv_curve<-tvParameter(model[,1],S=120,N=600,plot=TRUE,cex.axis=3,cex.main=3,cex.points=3,
                        mar=c(5,6,4,2),mfrow=c(1,3))
# Parametric
c0     <- c(0.20,0.08)
alpha0 <- c(0.40,0.45)
beta0  <- c(0.00,0.95,-0.95,0.1)
fit<-tvGarchKalmanFit(data=model[,1],c=c0,alpha=alpha0,beta=beta0,type=type,trig="cos",arg="2*pi*u")
predict(fit,n.ahead=30,Plot=TRUE,legend.cex=1.5,legend.pos="topleft")
# Smoothing Spline
FitNonParam_ss<-tvGarchNonParamFit(data=model[,1],tv_curve,smoothing=list(spar=c(0.5,0.68,0.8),
                                   nknots=c(3,3,3)),plot=TRUE,method="Smoothing Spline")
TVparameter_ss<-predict(FitNonParam_ss)
FitNonParamKF_ss<-tvGarchNonParamKF(model[, 1], TVparameter_ss)
predict(FitNonParamKF_ss, n.ahead = 20, Plot=TRUE)
# Local Polynomial
FitNonParam_lp<-tvGarchNonParamFit(data=model[,1],tvparameter=tv_curve,
                                   localPoly=list(span=c(0.70,2.5,1.4),degree=c(1,2,2)),
                                    plot=TRUE,method="Local Polynomial")
TVparameter_lp<-predict(FitNonParam_lp)
FitNonParamKF_lp<-tvGarchNonParamKF(model[, 1], TVparameter_lp)
predict(FitNonParamKF_lp, n.ahead = 20, Plot=TRUE)
# B-Spline
FitNonParam_bs<-tvGarchNonParamFit(data=model[,1],tvparameter=tv_curve,
                                     bspline=list(degree=c(2,2,2),df=c(3,4,2)),
                                     plot=TRUE,method="B-Spline")
TVparameter_bs<-predict(FitNonParam_bs)
FitNonParamKF_bs<-tvGarchNonParamKF(model[,1],TVparameter_bs)
predict(FitNonParamKF_bs,n.ahead=20,Plot=TRUE)
```

---

residuals,tvGarchKF-method

Residuals method.

Description

Residuals method.

Usage

```
## S4 method for signature 'tvGarchKF'
residuals(object, as.zoo = TRUE, ...)
```

Arguments

object	Object class tvGarchKF
as.zoo	Boolean flag.
...	Others.

---

summary,tvGarchKF-method *Summary method.*

Description

Summary method.

Usage

```
## S4 method for signature 'tvGarchKF'
summary(object, ...)

## S4 method for signature 'tvGarchKFNonParam'
summary(object, ...)
```

Arguments

object	Object class tvGarchKF or tvGarchKFNonParam.
...	Others.

---

tvGarch_Sim *Generating Simulations using a tv-Garch Model*

Description

Simulate from a tv-Garch(1,1) model, using specific functions for the tv-parameters.

Usage

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

Arguments

n	This is the value of the length of the series to be simulated.
c	A vector for the coefficients for the function $c(u)$ in the model.
alpha	A vector for the coefficients for the function $\alpha(u)$ in the model.
beta	A vector for the coefficients for the function $\beta(u)$ in the model.
type	A character type=c("polynomial","NoLineal","trigonometric"), which indicates what type of function will be assigned to each tv-parameters.
exponentes	A vector for the exponents for the "NoLinear" case.
trig	A character parameter indicating the trigonometric function to be used, "cos" or "sin".
arg	This is the value of the argument to be used in the trigonometric function.
seed	This value is the seed to simulate series.

Value

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

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, c = 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,c=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` *Estimate a tv-GARCH(1,1) by using the Kalman Filter.*

Description

This function estimates a time-varying GARCH(1,1) (tv-GARCH(1,1)) model using the Kalman Filter (KF). The model parameters are allowed to evolve smoothly over rescaled time $(u = t/T)$ according to user-specified functional forms. The time-varying parameters $c(t)$, $\alpha(t)$ and $\beta(t)$ determine the dynamics of the condition variance through $c(u_t) + \alpha(u_t) r_{t-1}^2 + \beta(u_t)\sigma_{t-1}, u_t = \frac{t}{T}$. The functional forms of the time-varying parameters are specified by the user and the corresponding coefficients are estimated using the KF-based procedure.

Usage

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

Arguments

`data`	A univariate time series, which can be an object of class "ts" or "zoo".
`c`	A numeric vector containing the initial values of the coefficients defining the function $c(u)$. These values are used to initialize the estimation procedure.
`alpha`	A numeric vector containing the initial values of the coefficients defining the function $\alpha(u)$. These values are used to initialize the estimation procedure.
`beta`	A numeric vector containing the initial values of the coefficients defining the function $\beta(u)$. These values are used to initialize the estimation procedure.

type	A character vector specifying the functional form assigned to each time-varying parameter. Available options are "polynomial", "NoLineal" and "trigonometric". The three elements correspond to $(c(u)), \alpha(u), \beta(u)$.
exponentes	A numeric vector specifying the exponents used in the nonlinear specification when type = "NoLineal".
trig	A character value specifying the trigonometric function used when type = "trigonometric". Available options are "cos" and "sin".
arg	A character string specifying the argument of the trigonometric function used when type = "trigonometric".
predict	A non-negative integer specifying the number of steps ahead for volatility forecasting. The default is 0, in which case no out-of-sample forecasts are computed during model fitting.
trace.log	Logical. If TRUE, additional information about the estimation procedure is printed. The default is FALSE.
title	An optional character string specifying a title associated with the fitted object. The default is NULL.
description	An optional character string providing a user-defined description of the fitted model. The default is NULL.

Details

The time-varying parameters can be represented using polynomial, nonlinear or trigonometric functional forms. For the polynomial specification,

$$c(t) = c_0 + c_1 u + c_2 u^2 + \ldots + c_p u^p,$$
$$\alpha(t) = a_0 + a_1 u + a_2 u^2 + \ldots + a_p u^p,$$
$$\beta(t) = b_0 + b_1 u + b_2 u^2 + \ldots + b_p u^p,$$

where $u = t/T$, with $t = 1, 2, \ldots, T$. For the nonlinear specification,

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

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

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

where the exponents $q_{c,j}$, $q_{\alpha,j}$ and $q_{\beta,j}$ are specified through the exponentes argument For the trigonometric specification,

$$c(t) = c_0 + c_1 g(u),$$
$$\alpha(t) = a_0 + a_1 g(u),$$
$$\beta(t) = b_0 + b_1 g(u),$$

where $(g(u))$ denotes the trigonometric function specified through trig, with its argument defined by arg.

Value

A list of class "tvGarchKF".

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

Arguments

x	A vector of coefficents to fit.
data	A univariate time series, where could be an object of class "ts" or "zoo".
c	A vector for the coefficients for the function $c(u)$ in the model.
alpha	A vector for the coefficients for the function $\alpha(u)$ in the model.
beta	A vector for the coefficients for the function $\beta(u)$ in the model.
nsample	A value of data length.

type	A character type=c("polynomial","NoLineal","trigonometric"), which indicates what type of function will be assigned to each tv-parameters.
exponentes	A vector for the exponents for the "NoLinear" case.
trig	A character parameter indicating the trigonometric function to be used, "cos" or "sin".
arg	This is the value of the argument to be used in the trigonometric function.
predict	A numeric value that indicates the h-steps of volatility prediction.
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,
  data,
  c,
  alpha,
  beta,
  nsample = length(data),
  type = c("polynomial", "NoLineal", "trigonometric"),
  exponentes,
  trig,
  arg,
  trace.log = FALSE,
  predict
)
```

Arguments

x	Vector of coefficents to fit.
data	A univariate time series, where could be an object of class "ts" or "zoo".
c	A vector for the coefficients for the function $c(u)$ in the model.
alpha	A vector for the coefficients for the function $\alpha(u)$ in the model.
beta	A vector for the coefficients for the function $\beta(u)$ in the model.

nsample	A value of data length.
type	A character type=c("polynomial","NoLineal","trigonometric"), which indicates what type of function will be assigned to each tv-parameters.
exponentes	A vector for the exponents for the "NoLinear" case.
trig	A character parameter indicating the trigonometric function to be used, "cos" or "sin".
arg	This is the value of the argument to be used in the trigonometric function.
trace.log	Variable to print names of coefficients.
predict	A numeric value that indicates the h-steps of volatility prediction.

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<-"3*(1-log(u))"
model<-tvGarchKalmanPrint(fit@coef,ipsa,c=c,alpha=alpha,beta=beta,type=type_fit,trig="cos",arg=arg)
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)
```

---

tvGarchKF-class *Clase tvGarchKF*

Description

Define principal S4 class of parametrics methods.

Slots

`call` User call.

`coef` Estimation of coefficients c, α and $beta$.

`var.coef` Matrix of covariance.

`residuals` Residuals

`sigma2` Conditional variance.

`data` A univariate time series, where could be an object of class "ts" or "zoo".

`loglik` Log-likelihood.

`aic` AIC value.

`title` Title.

`description` Description.

`method` A character string specifying the parametric estimation approach used to the tv-parameter curves.

`type` A charactertype=c("polynomial","NoLineal","trigonometric"), which indicates what type of function will be assigned to each tv-parameters.

`pars` A vector with initial coefficients (c, α and $beta$).

`exponentes` A vector for the exponents for the "NoLinear" case.

`trig` A character parameter indicating the trigonometric function to be used, "cos" or "sin".

`arg` This is the value of the argument to be used in the trigonometric function.

`stationary_coef` A vector with stationary coefficients (c, α and $beta$).

---

tvGarchKFNonParam-class

Clase tvGarchKFNonParam

Description

Define principal S4 class of non-parametrics methods.

Slots

`call` User call.

`Check_condition` Stationarity condition

`fitted.values` A data frame containing the estimated non-parametric curves $c(u)$, $\alpha(u)$ and $\beta(u)$.

`tvparameter` The original time-varying parameter structure used as input.

`data` A univariate time series, where could be an object of class "ts" or \"zoo".

`ParamStationary` Estimation of stationary coefficients.

`Omega` Estimation of c parameter.

`Alpha` Estimation of α parameter.

`Beta` Estimation of β parameter.

`title` Title.

`HattvParameters` Predicted values of the non-parametric parameter curves over the complete time domain.

`Param_model` This variable it's the stationary model.

`best_degree` The degree of the spline basis functions.

`best_df` The degrees of freedom used to estimate the tv-parameter curves $c(u)$, $\alpha(u)$ and $\beta(u)$.

`local_used` A list containing local polynomial tuning parameters.

`smoothing_used` A list containing the smoothing spline tuning parameters.

`method` The selected non-parametric estimation method.

`description` Description.

`tvGarchKFNonParam_Fit-class`
Clase tvGarchKFNonParam

Description

Define S4 class of non-parametrics methods applied Filter Kalman.

Slots

`title` Title.

`call` User call.

`fitted.values` This value it's the conditional variance.

`tvparameter` The original time-varying parameter structure used as input.

`data` A univariate time series, where could be an object of class "ts" or "zoo".

`Omega` Estimation of c parameter.

`Alpha` Estimation of α parameter.

`Beta` Estimation of β parameter.

`Param_model` This variable it's the stationary model.

`method` The selected non-parametric estimation method.

`description` Description.

`tvGarchNonParamFit` *EStimate tv-GARCH(1,1) parameter curves using non-parametric smoothing methods.*

Description

Given a set of initial local estimates of the time-varying parameters, this function applies smoothing spline, local polynomial, or B-spline to obtain smooth estimates tv-GARCH(1,1) parameter curves ($c(u)$, $\alpha(u)$, $\beta(u)$). The tuning parameters can be specified directly by the user or selected automatically using `auto.tvGarchKF()`.

Usage

```
tvGarchNonParamFit(
  data,
  tvparameter,
  smoothing = list(spar = c(0.5, 0.5, 0.5), nknots = c(12L, 12L, 12L), methods = NULL,
    control.spar = list(lower = 0.5, upper = 1.2), max_knots = NULL),
  bspline = list(degree = 1:3, df = c(3, 3, 4)),
  localPoly = list(span = seq(0.1, 0.45, by = 0.05), degree = 1:2),
  plot = TRUE,
  method = c("Smoothing Spline", "Local Polynomial", "B-Spline"),
  title = NULL,
  description = NULL
)
```

Arguments

`data`	A univariate time series, which can be an object of class "ts", "zoo" or "xts".
`tvparameter`	An object containing the initial estimates of the time-varying GARCH parameters $c(u)$, $\alpha(u)$ and $\beta(u)$, typically obtained from `tvParameter()` or extracted from an object of class "tvGarchKF".
`smoothing`	A list specifying the tuning parameters for the smoothing spline method. The component spar = $(spar_c, spar_\alpha, spar_\beta)$ controls the amount of smoothing for $c(u)$, $\alpha(u)$ and $\beta(u)$, respectively, while nknots = $(nknots_c, nknots_\alpha, nknots_\beta)$ specifies the corresponding number of knots. The optional components methods, control.spar and max_nknots provide additional control over the smoothing procedure.
`bspline`	A list specifying the tuning parameters for the B-spline method. The component degree specifies the degree of the B-spline basis, whereas df specifies the degrees of freedom used to estimate the parameter curves $c(u)$, $\alpha(u)$ and $\beta(u)$. Curve-specific values can be supplied for each time-varying parameter.
`localPoly`	A list specifying the tuning parameters for the local polynomial method. The component span controls the proportion of observations used in each local fit, whereas degree specifies the polynomial degree. Curve-specific values can be supplied for $c(u)$, $\alpha(u)$ and $\beta(u)$.

plot	Logical. If TRUE, the estimated time-varying parameter curves are displayed. The default is TRUE.
method	A character string specifying the non-parametric smoothing method. Available options are "Smoothing Spline", "Local Polynomial", and "B-Spline". A single method is applied to all three curves $c(u)$, $\alpha(u)$, and $\beta(u)$, although curve-specific tuning parameters can be specified.
title	An optional character string specifying a title associated with the fitted object. The default is NULL.
description	An optional character string providing a user-defined description of the fitted model. The default is NULL.

Value

An object of class "tvGarchKFNonParam" containing the non-parametric estimates of the tv-GARCH parameter curves. The returned object includes:

- `tvparameter`: The initial estimates of the time-varying GARCH parameters used as input.
- `fitted.values`: A data frame containing the fitted non-parametric curves $c(u)$, $\alpha(u)$ and $\beta(u)$ at the observed time points.
- `HattvParameters`: The estimated non-parametric curves evaluated over the complete time domain.
- `method`: The selected non-parametric smoothing method.
- `C`, `Alpha`, `Beta`: Fitted objects associated with the non-parametric estimation of ($c(u)$, $\alpha(u)$, $\beta(u)$), respectively.
- `Method-specific tuning parameters`: The tuning parameters associated with the selected non-parametric method, such as spar, nknots, span, degree or df.
- `Check_condition`: Diagnostic information for the local stationarity condition $(\alpha(u)+\beta(u) < 1)$, including the maximum value of $(\alpha(u) + \beta(u))$ over the observed time domain and an indicator of whether the condition is satisfied.

The resulting object can be used with `predict()` to obtain the estimated time-varying parameter curves required as input to `tvGarchNonParamKF()`.

Examples

```
library("tvGarchKF")
n = 2000
seed = 321
type = c("polynomial","trigonometric","polynomial")
model<-tvGarch_Sim(n=n,c=c(0.18,0.10),alpha=c(0.45,0.4),beta=c(0,1.05,-1.05,0.1),
                       type=type,trig="cos",arg="2*pi*u",seed=seed)
tv_curve<-tvParameter(model[,1],S=120,N=600,plot=TRUE,cex.axis=3,cex.main=3,cex.points=3,
                         mar=c(5,6,4,2),mfrow=c(1,3))
fit_par<-auto.tvGarchKF(object=tv_curve,data=model[,1],k_grid=3:8,bs_gam="ps",gam_method="REML",
                         spar_grid=seq(0.3,1.3,by=0.01),span_grid=seq(0.50,2.50,by=0.05),
                           loess_degree=c(1,2,2), bs_degree_grid=c(2,2,2),tol=1e-3)
# Smoothing Spline
FitNonParam_ss<-tvGarchNonParamFit(data=model[,1],tv_curve,smoothing=list(spar=c(0.5,0.68,0.8),
```

```
                                    nknots=c(3,3,3)),plot=TRUE,method="Smoothing Spline")
# Local Polynomial
FitNonParam_lp<-tvGarchNonParamFit(data=model[,1],tvparameter=tv_curve,
                                    localPoly=list(span=c(0.70,2.5,1.4),degree=c(1,2,2)),
                                     plot=TRUE,method="Local Polynomial")
# B-Spline
FitNonParam_bs<-tvGarchNonParamFit(data=model[,1],tvparameter=tv_curve,
                                      bspline=list(degree=c(2,2,2),df=c(3,4,2)),
                                      plot=TRUE,method="B-Spline")
```

---

tvGarchNonParamKF *Kalman Filter estimation with non-parametric tv-GARCH parameters.*

Description

This function applies the Kalman Filter (KF) to a tv-GARCH(1,1) model using non-parametric estimates ofthe time-varying parameter curves $c(u)$, $\alpha(u)$ and $\beta(u)$. These curves are typically obtained by applying predict() to an object returned by `tvGarchNonParamFit()`. The resulting parameter estimates are then incorporated into the KF recursion to update the state vector (X_{t+1} and obtain estimates of the conditional variance ($\widehat{\sigma}^2_{t+1}$).

Usage

```
tvGarchNonParamKF(data, TVparameter, title = NULL, description = NULL)
```

Arguments

`data`	A univariate time series of returns used to construct the squared-return innovation representation.
`TVparameter`	An object containing the non-parametric estimates of the tv-GARCH parameter curves $c(u)$, $\alpha(u)$, $\beta(u)$. It is typically obtained by applying `predict()` to an object fitted with `tvGarchNonParamFit()`, and is subsequently used as input to the Kalman Filter.
`title`	An optional character string specifying a title associated with the fitted object. The default is NULL.
`description`	An optional character string providing a user-defined description of the fitted model. The default is NULL.

Value

An object of class "tvGarchKFNonParam_Fit" containing the Kalman-filter-based estimates of the conditional variance obtained using non-parametric estimates of the tv-GARCH parameters. The returned object includes the fitted conditional variance, the original time series, the estimated time-varying parameter curve, the selected non-parametric smoothing method and the corresponding fitted model objects.

The returned objectinherits from the general tv-GARCH Kalman Filter class and can therefore be used with the S3/S4 methods provided by the package. In particular, `fitted()` extracts the fitted conditional variance, `plot()` displays either the estimated conditional volatility forecasts and their corresponding confidence intervals and `summary()` provides a summary of the fitted model and the selected non-parametric specification.

Examples

```
library("tvGarchKF")
n = 2000
seed = 321
type = c("polynomial","trigonometric","polynomial")
model<-tvGarch_Sim(n=n,c=c(0.18,0.10),alpha=c(0.45,0.4),beta=c(0,1.05,-1.05,0.1),
                       type=type,trig="cos",arg="2*pi*u",seed=seed)
tv_curve<-tvParameter(model[,1],S=120,N=600,plot=TRUE,cex.axis=3,cex.main=3,cex.points=3,
                        mar=c(5,6,4,2),mfrow=c(1,3))
fit_par<-auto.tvGarchKF(object=tv_curve,data=model[,1],k_grid=3:8,bs_gam="ps",gam_method="REML",
                        spar_grid=seq(0.3,1.3,by=0.01),span_grid=seq(0.50,2.50,by=0.05),
                          loess_degree=c(1,2,2), bs_degree_grid=c(2,2,2),tol=1e-3)
# Smoothing Spline
FitNonParam_ss<-tvGarchNonParamFit(data=model[,1],tv_curve,smoothing=list(spar=c(0.5,0.68,0.8),
                                    nknots=c(3,3,3)),plot=TRUE,method="Smoothing Spline")
TVparameter_ss<-predict(FitNonParam_ss)
FitNonParamKF_ss<-tvGarchNonParamKF(model[, 1], TVparameter_ss)
# Local Polynomial
FitNonParam_lp<-tvGarchNonParamFit(data=model[,1],tvparameter=tv_curve,
                                    localPoly=list(span=c(0.70,2.5,1.4),degree=c(1,2,2)),
                                     plot=TRUE,method="Local Polynomial")
TVparameter_lp<-predict(FitNonParam_lp)
FitNonParamKF_lp<-tvGarchNonParamKF(model[, 1], TVparameter_lp)
# B-Spline
FitNonParam_bs<-tvGarchNonParamFit(data=model[,1],tvparameter=tv_curve,
                                      bspline=list(degree=c(2,2,2),df=c(3,4,2)),
                                      plot=TRUE,method="B-Spline")
TVparameter_bs<-predict(FitNonParam_bs)
FitNonParamKF_bs<-tvGarchNonParamKF(model[, 1],TVparameter_bs)
```

`tvParameter` *Estimate the local structure of tv-GARCH(1,1) parameters.*

Description

This function performs an exploratory analysis of 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. These quantities satisfy $T = S(M-1) + N$. The midpoint of the j-th block, for $j = 1, \ldots, M$, is defined as $t_j = S(j-1) + N/2$. For each block, a stationary GARCH(1,1) model is fitted using the obervations contained in that block. The resulting sequence of local estimates provides an empirical trajectory of the time-varying parameters $(c(u))$, $\alpha(u)$ and $\beta(u)$. These trajectories can can be used as an exploratory tool for identifying suitable functional forms or as initial estimates for subsequent parametric or non-parametric estimation of the tv-GARCH parameter curves.

Usage

```
tvParameter(
  data,
  S,
  N,
  plot = TRUE,
  title = NULL,
  description = NULL,
  cex.axis = 2,
  cex.main = 2.5,
  cex.points = 2,
  mar = c(5, 6, 4, 2),
  mfrow = c(1, 3)
)
```

Arguments

data	A univariate time series, where could be an object of class "ts" or "zoo".
S	A positive integer specifying the step size or shift between consecutive overlapping blocks.
N	A positive integer specifying the number of observations contained in each block.
plot	A logical value indicating whether the local estimates of the time-varying parameter curves should be displayed. The default is TRUE.
title	An optional character string specifying a title associated with the fitted object. The default is NULL.
description	An optional character string providing a user-defined description. The default is NULL.
cex.axis	A numeric expansion factor controlling the size of the axis annotations. The default is 2.
cex.main	A numeric expansion factor controlling the size of the main title. The default is 2.5.
cex.points	A numeric expansion factor controlling the size of the points representing the local estimates of the time-varying parameters. The default is 2.
mar	A numeric vector specifying the plot margins in the order bottom, left, top and right. The default is c(5,6,4,2).
mfrow	A numeric vector of length two specifying the graphical layout. The default, c(1,3), arranges the local estimates of $c(u)$, $\alpha(u)$ and $\beta(u)$ in a single row.

Value

Data frame who contains c, 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)
```

---

vcov,tvGarchKF-method *Vcov method.*

Description

Vcov method.

Usage

```
## S4 method for signature 'tvGarchKF'
vcov(object, ...)
```

Arguments

object	Object class tvGarchKF
...	Others.

Index

∗ **datasets**
    indipsa, 5

auto.tvGarchKF, 2

fitted,tvGarchKF-method, 5
fitted,tvGarchKFNonParam-method
        *(*fitted,tvGarchKF-method*)*, 5
fitted,tvGarchKFNonParam_Fit-method
        *(*fitted,tvGarchKF-method*)*, 5

indipsa, 5

nobs,tvGarchKF-method, 6

plot,tvGarchKF,ANY-method, 6
plot,tvGarchKF,missing-method
        *(*plot,tvGarchKF,ANY-method*)*, 6
plot,tvGarchKFNonParam,ANY-method
        *(*plot,tvGarchKF,ANY-method*)*, 6
plot,tvGarchKFNonParam,missing-method
        *(*plot,tvGarchKF,ANY-method*)*, 6
plot,tvGarchKFNonParam_Fit,ANY-method
        *(*plot,tvGarchKF,ANY-method*)*, 6
plot,tvGarchKFNonParam_Fit,missing-method
        *(*plot,tvGarchKF,ANY-method*)*, 6
predict,tvGarchKF-method, 11
predict,tvGarchKFNonParam-method
        *(*predict,tvGarchKF-method*)*, 11
predict,tvGarchKFNonParam_Fit-method
        *(*predict,tvGarchKF-method*)*, 11

residuals,tvGarchKF-method, 13

summary,tvGarchKF-method, 14
summary,tvGarchKFNonParam-method
        *(*summary,tvGarchKF-method*)*, 14

tvGarch_Sim, 14
tvGarchKalmanFit, 16
tvGarchKalmanLoglike, 18
tvGarchKalmanPrint, 19
tvGarchKF-class, 20
tvGarchKFNonParam-class, 21
tvGarchKFNonParam_Fit-class, 22
tvGarchNonParamFit, 23
tvGarchNonParamKF, 25
tvParameter, 26

vcov,tvGarchKF-method, 28