# Calibrating and Forecasting Financial Portfolios with GARCH(1, 1) Model

Mohan Wu and Zhiyang Zhang, University of Waterloo

**Abstract**

Forecasting future stock prices with the assumption that the volatility is constant has proven to be poor in the past. The solution to this is stochastic volatility and GARCH(1,1) is often used because it is easy to calibrate. Sometimes fund managers would like to forecast volatility that is weeks ahead of time to manage their portfolio. Currently, the GARCH(1,1) is calibrated using weekly or daily data. The problem with calibrating the model using weekly data is the lack of observation data which leads to higher variance in the estimation. There is also a problem with calibrating using daily data in that the dynamics of the change in volatility over weeks would be lost. To address these problem, I propose the rolling method where daily data is transformed into 5 sets of weekly data. This fixes the lack of observation problem and the dynamic of the weekly change in volatility. An investing strategy using the rolling method to forecast volatility is tested on real stocks and its performance is compared against the return of the S&P 500.

## 1. Introduction

The most important feature of any stochastic volatility model is the ability to accurately forecast volatility. The forecast can then help predict future asset prices. The GARCH(1,1) model is studied extensively for its speed and ease in calibration. From Namugaya, Weke and Charles (2014), we know that the GARCH(1,1) model does better than most GARCH(p, q) models. This give us the motivation to find the best calibration method with the GARCH(1,1) model. As in Engle and Patton (2001), we look at the performance of the GARCH(1,1) model on the financial market using various methods in calibration.

Often, portfolio managers would like to forecast stock prices that are a few days or weeks ahead to make a profitable portfolio. It might seem reasonable at first to use weekly data to capture the dynamic of the volatility change over several days. The problem arises when there are too few data for the model to calibrate with, increasing the variability in its volatility predictions. This problem would be solved if we were to calibrate the model using daily data but we would lose the weekly volatility change effect that we are interested in. The solution to both problems is to use a combination of the two methods: the rolling-window method studied by Ardia and Hoogerheide (2013).

In this method, we start with daily data and then split the data into five groups of weekly data. We then use the 5 sets of weekly data to calibrate the GARCH(1,1) model. To measure the predictability of the rolling-window method, we compare models calibrated with rolling data, weekly data and daily data. Specifically, we use the absolute prediction distance on a 5-day forecast for each of the models. In this paper, we look at the four year stock prices from

2012-2015 of 5 stocks. These stocks belong to various market cap sizes which is a good representation of the market as a whole. Calibrating our models with various sized companies will test their effectiveness in the general market. We employ a portfolio management strategy that considers both the forecasted return and variance of the stocks. The final return of the portfolio over a year's time will determine the effectiveness of the rolling-window method in forecasting volatility.

The rest of the paper is followed by calibration and estimation of the parameters of the GARCH(1,1) using the max-likelihood method presented in Reider (2009). The different calibration methods are then tested on a 5-day forecast of stock prices. Following that, the rolling method is tested over a year using a portfolio management strategy of five stocks. Finally, the results are examined and analyzed to arrive at a conclusion.

## 2. Methodology

### 2.1 Parameter Estimation

We first introduce the GARCH(1,1) model that we use to forecast the volatilities.

$$\begin{aligned} x_{ti} &= \mu_i + \epsilon_{ti} \\ \epsilon_{ti} &= \sigma_{ti} z_{ti} \\ \sigma_{ti}^2 &= \omega_i + \alpha_i \epsilon_{t-1,i}^2 + \beta_i \sigma_{t-1,i}^2 \\ i.i.d.\ z_t &\sim \mathcal{N}(0, R) \end{aligned} \qquad (1)$$

We let $x_{ti}$ be the log return of stock *i* on day t and $\mu_i$ be the average of all $x_{ti}$. We begin by calculating $\varepsilon_{ti}$ at any day t. Now the question remains with estimating $\sigma_{ti}$. To do so, we need to

find optimal parameters $\omega_i$, $\alpha_i$ and $\beta_i$. To simplify the notation, we let $\theta = (\omega, \alpha, \beta)$. We use the max-likelihood to estimate our parameter $\theta$. We further assume that our likelihood function is Gaussian implying that $z_t$ are i.i.d. Gaussian. The log-likelihood can then be simplified according to Posedel (2005) to:

$$L_i(\theta|\varepsilon_i,\ \varepsilon_{i0}^2,\ \sigma_{i0}^2) = \frac{1}{2N}\sum_{t=1}^{N} l_i(\theta) \text{ where } l_i(\theta) = -log\,\sigma_{ti}^2(\theta) + \frac{\varepsilon_{ti}^2}{\sigma_{ti}^2(\theta)} \quad (2)$$

Here $\varepsilon_i = (\varepsilon_{i1},\ \varepsilon_{i2} \ldots \varepsilon_{iN})$ and $\varepsilon_{i0}^2,\ \sigma_{i0}^2$ are initial parameters to estimate $\sigma_{i1}^2$. We use the generally accepted estimates $\varepsilon_{i0}^2 = \frac{1}{N}\sum_{t=1}^{N} \varepsilon_{it}^2$ and $\sigma_{i0}^2 = 1$.

Since we are using GARCH(1,1) we can simplify the optimization problem from three parameters down to two. Finding the max likelihood of two parameters is a lot more computationally efficient than finding the max likelihood of three. We can do this by profiling out two parameters and letting $\eta = \frac{\alpha}{\omega}$. The log-likelihood in equation 2 can then be modified as follows:

$$l_i(\theta) = -log\,\omega \cdot \hat{\sigma}_{ti}^2(\theta) + \frac{\varepsilon_{ti}^2}{\omega \cdot \hat{\sigma}_{ti}^2(\theta)} \text{ where } \hat{\sigma}_{ti} = 1 + \eta\varepsilon_{t-1,\,i}^2 + \beta\hat{\sigma}_{t-1,\,i}^2 \quad (3)$$

To make it easier for optim to find the optimal values for the parameters, we use closed-form estimates of the parameters proposed by Kristensen and Linton (2006). After obtaining the estimates for $\beta$ and $\eta$, we can convert $\eta$ back to $\alpha$ and $\omega$. This method works well on daily data because all of the parameters converges to the max-likelihood.

Unfortunately, the closed-form estimate fails for weekly data as $\alpha$ is sometimes estimated to be negative when $\alpha$ is positive by assumption. This is likely due to the lack of data since we are looking weekly data over three years. In order to solve this issue, we use a two step optimization process. First, we choose an arbitrary $\beta_0 < 1$ which satisfies the assumption that $\alpha + \beta_0 < 1$. We then use optimize to do an 1-d optimization over $\eta$ to find the optimal $\eta$ for our choice of $\beta_0$. Now, we use our estimates for $\eta$ and $\beta_0$ to find the max-likelihood parameters using optim and they do converge to the max-likelihood. Details for checking the parameters are shown in the Appendix.

For our last method, the rolling-window, we calibrate our parameter $\theta = (\omega, \alpha, \beta)$ with five separate univariate time series of weekly data. We use five different estimates of $\varepsilon_{i0}^2 = \frac{1}{N}\sum_{t=1}^{N} \varepsilon_{it}^2$ with each corresponding to the data in the series. We assume each univariate series have the same $\theta = (\omega, \alpha, \beta)$ and optimize the parameters similar to our daily data method. We first profile out $\alpha$ and $\omega$ in terms of $\eta$ and then we use optim to find the maximum of the sum of the five log-likelihood functions. Again details are in the Appendix but the parameters do converge to the max likelihood.

From our fit, we can back out the innovations $z_t$ and calculate their sample correlation R. Using R we can generate new innovations for forecast with innovations $z_t$ following a Gaussian distribution with mean 0 and variance R. In this way, we can use our estimated parameters $\theta$ to forecast $\sigma^2$ and ultimately, the price of the stock.

We look at the predictability of the three calibration methods from section 2.1 on the price of a company with ticker symbol LEA.

| Daily APD | Weekly APD | Rolling APD |
|---|---|---|
| 9.797345 | 6.646181 | 5.368854 |

We see that indeed, we are getting better results with weekly and rolling calibration methods than daily on an absolute prediction distance metric.

**2.2 Portfolio Management Strategy**

In order to test the GARCH(1,1) models, a portfolio management process is simulated and the performance of the portfolio is used to determine the effectiveness of the model. Since the rolling-method is the best in our simulation results, we use this method in our portfolio management strategy. We look at three different models based on this calibration method:

1. An SSV model with empirical innovations described in Appendix B
2. The SSV model with $R = I$d
3. The SSV model with a freely estimated $R$

Denote $C0$ as the amount of cash at time t=0 and $y0i$ as the market price of asset i at time 0. We start off with no assets and only cash. To simplify the process, we ignore transaction fees in trading, which means we can values all assets in terms of their current price at any given time. Using the principle of Modern Portfolio Theory (MPT), we developed the following

strategy(Elton, Gruber, Brown & Goetzmann, 2009) for day t=n, market price of the asset *yn* and current amount of cash *Cn*.

Fit the GARCH(1,1) model from day n-N to n (where N is user-defined) and then we have the forecast for day t+1 with the mean and variance:

$$\mu = E[\, yn+1 \,], \Sigma = \text{var}(\, yn+1 \,)$$

To determine the desired $qn = (qn1,..., qn5)$, the weight of the portfolio, we use an "expected return rate" approach. In this strategy, we set the expected return rate to be Ω, and minimize the risk, i.e., has the smallest variance. Then the problem becomes solving *q* such that:

$$\min q' \Sigma q \mid qp = \Omega, p = (\mu - yt)/ yt$$

which can be solved with Lagrange multipliers to be $q = \lambda\Sigma\text{-}1\, p$, where $\lambda = \Omega/(\, p' \Sigma p \,)$.

In our assumption, shorting is allowed in the model and acts the same as buying but with negative amount of shares. We constraint that the total short selling amount to be less than or equal to the amount of cash after buying shares. In this situation, suppose: $qn^+ = \max(qn\ , 0)$, $qn^- = -\min(qn\ , 0)$ then the constraint would be $Cn - qn^+ yn > qn^- yn$.

On the day t = n + 1, we have the actual stock price *yn+1*. We calculate the changes in assets and liquidate all our assets: $Cn+1 = Cn + qn\ (\, yn+1 - yn\, )$ . We continue this process for a year which we assume to be T = 252 days.

## 3. Results

We now explore portfolio management by forecasting five companies of different size on the NYSE. In order to avoid potential errors, we choose stocks with relatively longer histories and no stock splits during the period with which we are interested. The five chosen companies are based on Greenblatt's magic formula (2010) which ranks all stocks on their return on capital and price to enterprise value. We decide to choose large market cap: BIIB, mid market cap: MAN, LEA, DHI and small market cap: NHTC because volatility is significantly different between these three categories. This will truly test GARCH(1,1) in how well it can forecast volatility.

We apply our strategy over year 2016, with a three year data for model fitting purposes and initial cash $C0$ = 1,000,000. We choose three years because the historical price of the stock becomes less relevant after three years due to market changes and management changes. The three year data also avoids using prices from 2011 when the stock market crashed in August which would hurt the model's predictability.

We test several different expected return rates Ω ranging from 0.001 to 0.02. From our results, we find that for any return rate below 0.004, the portfolio does not change very much over the year; for any rate above 0.011, the portfolio becomes very volatile which is a sign of a lot of risk. The simulation results are as follow:

| Ω | 0.004 | 0.005 | 0.006 | 0.007 | 0.008 | 0.009 | 0.010 | 0.011 |
|---|---|---|---|---|---|---|---|---|
| Portfolio 1 | 6.2% | 8.4% | -0.8% | -0.9% | -3.0% | -3.4% | -3.8% | -8.3% |
| Portfolio 2 | 1.8% | 2.1% | 2.4% | 2.6% | 2.9% | 3.0% | 3.3% | 3.6% |
| Portfolio 3 | -9.8% | -1.0% | 7.7% | 6.3% | 9.6% | 0.5% | -2.9% | -0.3% |

All portfolios have a reasonable performance with a particular time period and return rate. We see that Portfolio 2, the portfolio using identity R in the SSV model, has a relatively steady positive performance overall. Portfolio 3, the one using a freely estimated R has the best performance of 9.6% among all our simulations with 0.008 return rate, but it requires a relatively precise return rate.

Since our stock selection, with 1 large market cap stock, 3 medium market cap stock and 1 small market cap stock, is a valid representation of the actual stock market in terms of market cap, it is fair to compare our return to the S&P 500. Sadly, none of them beat the S&P 500 Index in the year 2016, which is 9.84% (PK, 2016).

Therefore, our portfolio management could not be considered as a successful management. However, the result from our simulation, especially the result from Portfolio 2, do show the potential usage of a GARCH(1,1) model in a real-world stock market for stochastic forecasting.

## 4. Discussions

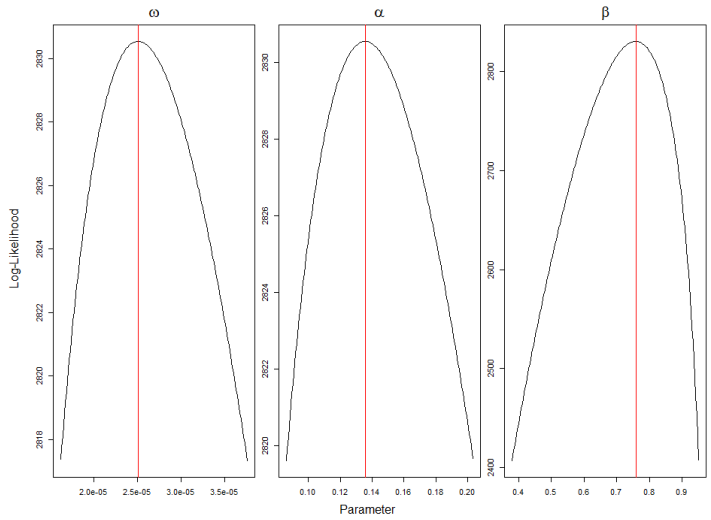
In this project, we implement and examine the GARCH(1,1) model in stochastic volatility forecasting in the stock market. We calibrate the model with daily data, weekly data and rolling data using max-likelihood. We examine the model by applying the model to the stock market with modern portfolio theory. From our results, it is clear that the GARCH(1,1) model is able to forecast the price changes in market to some degree, but there are a few shortcomings with the model.

One issue in the GARCH(1,1) model in portfolio management is tolerance. For the stock market, events such as policy changes make significant effects on the prices of stocks in a relatively short time, i.e, Brexit. Our GARCH(1,1) model have a hard time calibrating to such events. As for portfolio management, the code efficiency of our approach would not be able to handle larger portfolios with more stocks. The solution is to code in more efficient languages such as C++ and Julia. Besides efficiency, another shortcoming in our approach is the value of expected return rate. Currently we use brutal force testing methods to achieve a relatively optimal result, which cannot be applied to real-life situation. From our results, a huge difference can be observed when using different return rate under the same GARCH(1,1) model. For further research, the relationship between GARCH(1,1) models and the ideal expected return rate would be a valuable topic.

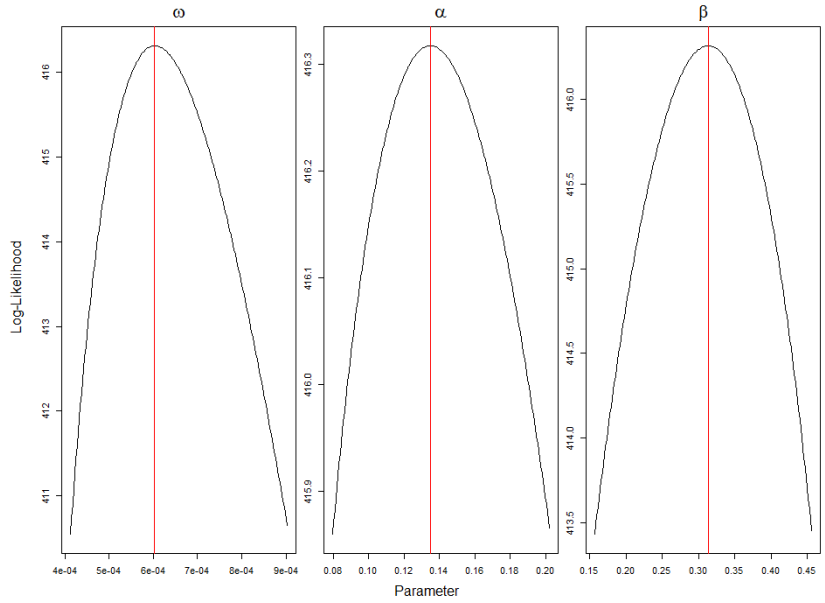
## Appendix

## A. MLE Convergence Check of the Parameters

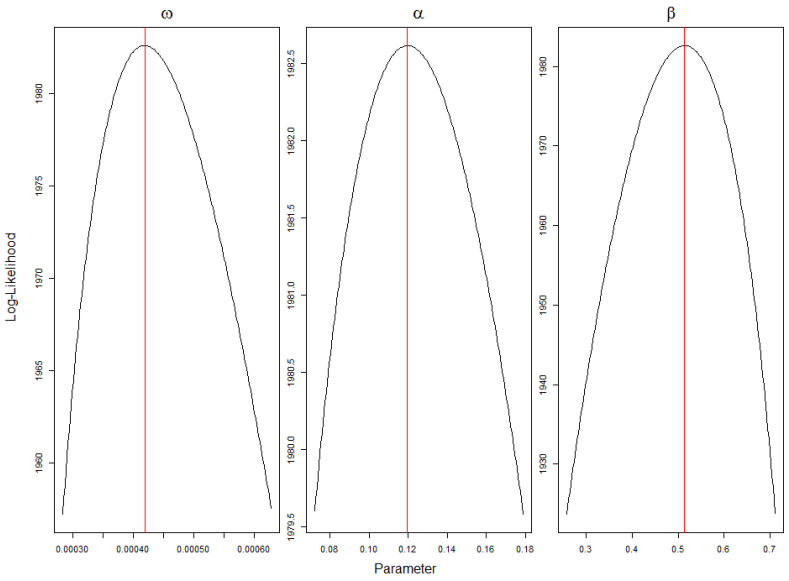
Daily Data MLE:




Weekly Data MLE:



Rolling Data MLE:




**B. SSV Model with Empirical Returns (Credit to Prof Martin Lysy)**

In the original formulation of the SSV model (1), the innovations are modeled as multivariate normals: i.i.d. $z_t \sim N(0, R)$. A simple extension to this model is perhaps most easily described by the estimation procedure. That is:

1. Start by fitting univariate GARCH(1, 1) models to each asset series $Y_{ti}^{(W)}$.

2. Back out the estimates innovations $\hat{Z}_t^{(W)}$.

3. Instead of directly taking the sample correlation of $\hat{Z}_t^{(W)}$, first transform the innovations in each asset to look very close to normal. Here is an R function to do this for an arbitrary vector x:

4. After applying the normalizing transformation to each innovation series $\hat{Z}_{ti}^{(W)}$, calculate the sample correlation of the result.

To simulate one step from the Empirical SSV model, do the following:

1. Generate i.i.d. $\tilde{z}_t \sim N(0, \hat{R})$, where $\hat{R}$ is the sample correlation of the normalized innovations.

2. Map $\tilde{z}_{ti}$ to the element of $\hat{Z}_{ti}(W)$ having rank closest to pnorm $(\tilde{z}_{ti})$.

3. Do this to each asset i, thus obtaining $z_t = (z_{t1}, \ldots, z_{td})$, for which each element z ti is an element of the corresponding sample $\hat{Z}_{ti}^{(W)}$.

4. Use this $z_{ti}$ in the GARCH simulation step.

**C. R code**

1. garch.R

```
#' Simulate Data from a GARCH(1,1) Model
#'
#' @param N total number of observations.
#' @param nsim number of time series to generate.
#' @param omega,alpha,beta GARCH(1,1) parameters.
#' @param eps0,sig0 Initial values, included as first observation.
#' @param mu vector of means of the stocks
#' @return Return
garch.sim <- function(N, omega, alpha, beta, mu, R, sig20, eps0, y0) {
  library(MASS)
  #set.seed(100)
  if(N==1){
    z <- as.matrix(mvrnorm(N, rep(0, length(mu)), R))
  } else{
    z <- t(mvrnorm(N, rep(0, length(mu)), R))
  }
  # memory allocation
  eps <- matrix(NA, length(mu), N+1)
  sig2 <- matrix(NA, length(mu), N+1)
  eps[,1] <- eps0
  sig2[,1] <- sig20
  for(ii in (1:N)+1) {
    sig2[,ii] <- omega + alpha * eps[,ii-1]^2 + beta * sig2[,ii-1]
    eps[,ii] <- sqrt(sig2[,ii]) *z[,ii-1]
  }
  # remove starting values
  sig2 <- sig2[,-1,drop=FALSE]
  eps <- eps[,-1,drop=FALSE]
  xt <- mu + eps
  logp<- log(y0)+apply(xt, 1, cumsum)
  p <- exp(logp)
  p
}

#' Fit a GARCH model for daily stock prices
#'
#' @param yt prices of stocks
#' @return MLE and zt (innovations).
garch.fit <- function(yt) {
  xt <- diff(log(yt))[-1] # log returns
  n <- length(xt)
  # convert to numeric format
```

```
  xt <- as.numeric(xt)
  muh <- mean(xt)

  #intial parameters
  eps <- xt - muh
  theta0 <- garch.clo(eps)
  eta0 <- theta0["alpha"]/theta0["omega"]
  beta0 <- theta0["beta"]
  eps20 <- mean(eps*eps)
  sig20 <- 1
  param0 <- c(eta0, beta0)
  gpfit <- optim(par = log(param0),
                 fn = function(param) {
                   -garch.profll(eta = exp(param[1]),
                                 beta = exp(param[2]),
                                 eps = eps, eps20 = eps20,
                                 sig20 = sig20)
                 })
  if(gpfit$convergence != 0) {
    warning("optim did not converge.")
  }

  # convert back to original GARCH parameters
  theta.mle <- garch.profmle(eta = exp(gpfit$par[1]),
                             beta = exp(gpfit$par[2]),
                             eps = eps, eps20 = eps20, sig20 = sig20)

  sig2 <- garch.sig2(theta.mle['omega'], theta.mle['alpha'], theta.mle['beta'], eps, eps20, sig20)
  zt <- eps/sig2

  c(list(theta = theta.mle,zt = zt, mu = muh, epsN = eps[n], sig2N =sig2[n]))
}

#' Fit a GARCH model for weekly stock prices
#'
#' @param yt prices of stocks
#' @return MLE and zt (innovations).
garch.fitw <- function(yt) {
  xt <- diff(log(yt))[-1] # log returns
  n <- length(xt)
  # convert to numeric format
  xt <- as.numeric(xt)
  muh <- mean(xt)

  #intial parameters
```

```
  eps <- xt - muh
  eps20 <- mean(eps*eps)
  sig20 <- 1
  beta0 <- 0.7
  opfit <- optimize(f=garch.profll, c(0,1000),
                    beta= beta0, eps=eps, eps20 =eps20,
                    sig20=sig20, maximum = TRUE)

  eta0 <- opfit$maximum
  param0 <- c(eta0, beta0)

  gpfit <- optim(par = log(param0),
                 fn = function(param) {
                   -garch.profll(eta = exp(param[1]),
                                 beta = exp(param[2]),
                                 eps = eps, eps20 = eps20,
                                 sig20 = sig20)
                 })
  if(gpfit$convergence != 0) {
    warning("optim did not converge.")
  }

  # convert back to original GARCH parameters
  theta.mle <- garch.profmle(eta = exp(gpfit$par[1]),
                             beta = exp(gpfit$par[2]),
                             eps = eps, eps20 = eps20, sig20 = sig20)

  sig2 <- garch.sig2(theta.mle['omega'], theta.mle['alpha'], theta.mle['beta'], eps, eps20, sig20)
  zt <- eps/sig2

  c(list(theta = theta.mle,zt = zt, mu = muh, epsN = eps[n], sig2N =sig2[n]))
}


#' Recover full parameter estimates from profile likelihood estimates
#' Credit to Prof Lysy
#' @param eta profile parameter equal to
#' @param beta GARCH parameter.
#' @param eps20,sig20 initial standardized volatility
#' @return The named vector
garch.profmle <- function(eta, beta, eps, eps20, sig20) {
  sig2 <- garch.sig2(1, eta, beta, eps, eps20, sig20) # tsig^2
  omegah <- mean(eps^2/sig2) # omega.hat(eta, beta)
  # transform back to original scale
  c(omega = as.numeric(omegah), alpha = as.numeric(eta*omegah),
```

```
    beta = as.numeric(beta))
}

#' GARCH Profile Loglikelihood
#' Credit to Prof Lysy
#' eps_t = omega * sig_t * z_t
#' z_t ~iid N(0,R)
#' sig_t^2 = 1 + eta * eps^2_t-1 + beta * sig^2_t-1,
#'
#' @param eta corresponds to alpha/beta from the original GARCH model
#' @param beta parameter of original GARCH model
#' @param eps vector of GARCH observations
#' @param eps2, sig20 initial value
#' @export
garch.profll <- function(eta, beta, eps, eps20, sig20, debug = FALSE) {
  n <- length(eps)
  sig2 <- garch.sig2(1, eta, beta, eps, eps20, sig20) #Profiled out alpha, omega
  if(any(sig2 < 0)) return(-Inf) #Make sure all sigt2 is positive
  omegah <- mean(eps^2/sig2)
  -.5 * (n + n*log(omegah) + sum(log(sig2)))
}


#' Computes Sigma^2 (volatilities) at time t from GARCH observations
#' Credit to Prof Lysy
#' @param omega, alpha, beta GARCH parameters.
#' @param eps vector of GARCH observations
#' @param eps20, sig20 initial values.
#' @return sig^2
garch.sig2 <- function(omega, alpha, beta, eps, eps20, sig20) {
  n <- length(eps)
  eps2 <- eps^2
  mu <- mean(eps2)
  if(missing(eps20)) eps20 <- mu
  if(missing(sig20)) sig20 <- mu
  fsig2 <- omega + alpha * c(eps20, eps2[-n]) #First part of the sigma^2 equation
  sig2 <-as.numeric(filter(fsig2, beta, "recursive", init = sig20))
  sig2
}

#' Closed-form estimator by Kristensen & Linton (using notation from the paper)
#'
#' @param eps GARCH observation parameter
#' @return vector of estimated GARCH parameters (alpha, beta, omega).
garch.clo <- function(eps) {
```

```
  n <- length(eps)
  xt <- eps*eps
  sigma2 <- mean(xt)
  rho <- cov(xt[1:(n-1)],xt[2:n])
  phi <- cov(xt[1:(n-2)], xt[3:n])/rho
  b <-  (phi^2 + 1 - 2*rho*phi)/(phi - rho)
  theta <- (-b + sqrt(b^2 -4))/2
  alpha <- theta + phi
  if (alpha <0 ){
    alpha <- abs(alpha)
  }
  beta <- -theta
  omega <- sigma2*(alpha+beta)
  c(omega = omega, alpha = alpha, beta = beta)
}
```

2. garchRoll.R

```
#Transform sample to normal looking histogram
ztrans <- function(x) {
  n <- length(x)
  z <- rank(x)/(n+1) # approximately have z[i] = Pr(x <= x[i])
  qnorm(z) # corresponding z-scores
}

#Get rank of sample
zrank <- function(x){
  n <- length(x)
  rank <- rank(x)/(n)
}

#' Simulate Data from a GARCH(1,1) Model
#'
#' @param N total number of observations.
#' @param nsim number of time series to generate.
#' @param omega,alpha,beta GARCH(1,1) parameters.
#' @param eps0,sig0 Initial values, included as first observation.
#' @param mu vector of means of the stocks
#' @return Return
garch.rsim <- function(N, omega, alpha, beta, mu, R, sig20, eps0, y0, emp, Zt) {
  library(MASS)
  set.seed(100)
  if(N==1){
    z <- as.matrix(mvrnorm(N, rep(0, length(mu)*5), R))
  } else{
    z <- t(mvrnorm(N, rep(0, length(mu)*5), R)) #5-day innovations
```

```
}
if(emp){
  pz <- pnorm(z)
  empzrank <- apply(Zt, 2, zrank)
  stockrank <- matrix(NA,length(mu)*5,N)
  empz <- matrix(NA, length(mu)*5,N)
  for(i in 1:(length(mu)*5)){
    stockrank[i,] <-vapply(pz[i,],function(x) which.min(abs(empzrank[,i]-x)),1)
    empz[i,] <- Zt[stockrank[i,],i]
    z <- empz
  }
}
# memory allocation for each the 5-day innovations
eps1 <- matrix(NA, length(mu), N+1)
eps2 <- matrix(NA, length(mu), N+1)
eps3 <- matrix(NA, length(mu), N+1)
eps4 <- matrix(NA, length(mu), N+1)
eps5 <- matrix(NA, length(mu), N+1)
sig21 <- matrix(NA, length(mu), N+1)
sig22 <- matrix(NA, length(mu), N+1)
sig23 <- matrix(NA, length(mu), N+1)
sig24 <- matrix(NA, length(mu), N+1)
sig25 <- matrix(NA, length(mu), N+1)
#z <- cbind(eps0/sig0, z) # first innovation determined by initial values
eps1[,1] <- eps0[,1] # 1st of 5-day epsilon (not epsilon of stock 1)
eps2[,1] <- eps0[,2]
eps3[,1] <- eps0[,3]
eps4[,1] <- eps0[,4]
eps5[,1] <- eps0[,5]
sig21[,1] <- sig20[,1]
sig22[,1] <- sig20[,2]
sig23[,1] <- sig20[,3]
sig24[,1] <- sig20[,4]
sig25[,1] <- sig20[,5]

for(ii in (1:N)+1) {
  #Simulate 5-day eps and sig2, order is z^(W,1)_1, z^(W,2)_1, ...
  sig21[,ii] <- omega + alpha * eps1[,ii-1]^2 + beta * sig21[,ii-1]
  eps1[,ii] <- sqrt(sig21[,ii]) *z[seq(1, length(mu), 5),ii-1]
  sig22[,ii] <- omega + alpha * eps2[,ii-1]^2 + beta * sig22[,ii-1]
  eps2[,ii] <- sqrt(sig22[,ii]) *z[seq(2, length(mu), 5),ii-1]
  sig23[,ii] <- omega + alpha * eps3[,ii-1]^2 + beta * sig23[,ii-1]
  eps3[,ii] <- sqrt(sig23[,ii]) *z[seq(3, length(mu), 5),ii-1]
  sig24[,ii] <- omega + alpha * eps4[,ii-1]^2 + beta * sig24[,ii-1]
  eps4[,ii] <- sqrt(sig24[,ii]) *z[seq(4, length(mu), 5),ii-1]
```

```
    sig25[,ii] <- omega + alpha * eps5[,ii-1]^2 + beta * sig25[,ii-1]
    eps5[,ii] <- sqrt(sig25[,ii]) *z[seq(5, length(mu), 5),ii-1]
  }
  # remove starting values
  sig21 <- sig21[,-1,drop=FALSE]
  eps1 <- eps1[,-1,drop=FALSE]
  sig22 <- sig22[,-1,drop=FALSE]
  eps2 <- eps2[,-1,drop=FALSE]
  sig23 <- sig23[,-1,drop=FALSE]
  eps3 <- eps3[,-1,drop=FALSE]
  sig24 <- sig24[,-1,drop=FALSE]
  eps4 <- eps4[,-1,drop=FALSE]
  sig25 <- sig25[,-1,drop=FALSE]
  eps5 <- eps5[,-1,drop=FALSE]
  mu <- c(mu)
  xt1 <- mu + eps1
  xt2 <- mu + eps2
  xt3 <- mu + eps3
  xt4 <- mu + eps4
  xt5 <- mu + eps5
  xt <- (xt1 + xt2 + xt3 + xt4 + xt5)/5
  logp<- log(y0)+t(apply(xt, 1, cumsum))
  p <- exp(logp)
  if (N == 1) {
    sig2 <- cbind(sig21, sig22, sig23, sig24, sig25)
    list(price = p, var = sig2)
  } else {
    p
  }
}

#' Fit a GARCH model for rolling stock prices
#'
#' @param yt prices of stocks
#' @return MLE and zt (innovations).
garch.fitr <- function(yt1, yt2, yt3, yt4, yt5) {
  xt1 <- diff(log(yt1))[-1]
  xt2 <- diff(log(yt2))[-1]
  xt3 <- diff(log(yt3))[-1]
  xt4 <- diff(log(yt4))[-1]
  xt5 <- diff(log(yt5))[-1]# log returns
  n <- length(xt1)
  # convert to numeric format
  xt1 <- as.numeric(xt1)
  xt2 <- as.numeric(xt2)
```

```
  xt3 <- as.numeric(xt3)
  xt4 <- as.numeric(xt4)
  xt5 <- as.numeric(xt5)
  muh1 <- mean(xt1)
  muh2 <- mean(xt2)
  muh3 <- mean(xt3)
  muh4 <- mean(xt4)
  muh5 <- mean(xt5)
  muh <- mean(muh1, muh2, muh3, muh4, muh5)

  #intial parameters
  eps1 <- xt1 - muh
  eps2 <- xt2- muh
  eps3 <- xt3 - muh
  eps4 <- xt4- muh
  eps5 <- xt5 - muh

  theta0 <- garch.clo(eps1)+ garch.clo(eps2)+ garch.clo(eps3)+ garch.clo(eps4)+ garch.clo(eps5)
  theta0 <- theta0/5
  eta0 <- abs(theta0["alpha"]/theta0["omega"])
  beta0 <- 0.65

  eps20 <- rowMeans(rbind(eps1*eps1, eps2*eps2, eps3*eps3, eps4*eps4, eps5*eps5),
na.rm=TRUE)
  sig20 <- 1
  param0 <- c(eta0, beta0)


  gpfit <- optim(par = log(param0),
                 fn = function(param) {
                   -garch.rprofll(eta = exp(param[1]),
                                  beta = exp(param[2]),
                                  eps1 = eps1, eps2=eps2, eps3=eps3, eps4=eps4,
                                  eps5 = eps5,eps20 = eps20,
                                  sig20 = sig20)
                 })
  if(gpfit$convergence != 0) {
    warning("optim did not converge.")
  }

  # convert back to original GARCH parameters
  theta.mle <- garch.rprofmle(eta = exp(gpfit$par[1]),
                              beta = exp(gpfit$par[2]),
                              eps = eps1,eps2=eps2, eps3=eps3, eps4=eps4,
                              eps5=eps5, eps20 = eps20, sig20 = sig20)
```

```
  sig21 <- garch.sig2(theta.mle['omega'], theta.mle['alpha'],
                      theta.mle['beta'], eps1, eps20[1], sig20)
  sig22 <- garch.sig2(theta.mle['omega'], theta.mle['alpha'],
                      theta.mle['beta'], eps2, eps20[2], sig20)
  sig23 <- garch.sig2(theta.mle['omega'], theta.mle['alpha'],
                      theta.mle['beta'], eps3, eps20[3], sig20)
  sig24 <- garch.sig2(theta.mle['omega'], theta.mle['alpha'],
                      theta.mle['beta'], eps4, eps20[4], sig20)
  sig25 <- garch.sig2(theta.mle['omega'], theta.mle['alpha'],
                      theta.mle['beta'], eps5, eps20[5], sig20)

  zt1 <- eps1/sig21
  zt2 <- eps2/sig22
  zt3 <- eps3/sig23
  zt4 <- eps4/sig24
  zt5 <- eps5/sig25

  eps <- c(eps1[n], eps2[n], eps3[n], eps4[n], eps5[n])
  sig2 <- c(sig21[n], sig22[n], sig23[n], sig24[n], sig25[n])
  c(list(theta = theta.mle,zt1 = zt1, zt2 =zt2, zt3=zt3, zt4=zt4, zt5=zt5,
         mu = muh, epsN = eps, sig2N =sig2))
}

#' Recover full parameter estimates from profile likelihood estimates
#' Credit to Prof Lysy
#' @param eta profile parameter equal to
#' @param beta GARCH parameter.
#' @param eps20,sig20 initial standardized volatility
#' @return The named vector
#'
garch.rprofmle <- function(eta, beta, eps1, eps2, eps3, eps4, eps5, eps20, sig20) {
  sig21 <- garch.sig2(1, eta, beta, eps1, eps20[1], sig20) # tsig^2
  sig22 <- garch.sig2(1, eta, beta, eps2, eps20[2], sig20)
  sig23 <- garch.sig2(1, eta, beta, eps3, eps20[3], sig20)
  sig24 <- garch.sig2(1, eta, beta, eps4, eps20[4], sig20)
  sig25 <- garch.sig2(1, eta, beta, eps5, eps20[5], sig20)

  omegah1 <- mean(eps1^2/sig21)
  omegah2 <- mean(eps2^2/sig22)
  omegah3 <- mean(eps3^2/sig23)
  omegah4 <- mean(eps4^2/sig24)
  omegah5 <- mean(eps5^2/sig25)
  omegah <- mean(c(omegah1,omegah2,omegah3, omegah4, omegah5))
```

```
  # omega.hat(eta, beta)
  # transform back to original scale
  c(omega = as.numeric(omegah), alpha = as.numeric(eta*omegah),
    beta = as.numeric(beta))
}

#' GARCH Profile Loglikelihood
#' Credit to Prof Lysy
#' eps_t = omega * sig_t * z_t
#' z_t ~iid N(0,R)
#' sig_t^2 = 1 + eta * eps^2_t-1 + beta * sig^2_t-1,
#'
#' @param eta corresponds to alpha/beta from the original GARCH model
#' @param beta parameter of original GARCH model
#' @param eps vector of GARCH observations
#' @param eps2, sig20 initial value
#' @export
garch.rprofll <- function(eta, beta, eps1, eps2, eps3, eps4, eps5, eps20, sig20, debug = FALSE) {
  n <- length(eps1)
  sig21 <- garch.sig2(1, eta, beta, eps1, eps20[1], sig20) #Profiled out alpha, omega
  sig22 <- garch.sig2(1, eta, beta, eps2, eps20[2], sig20)
  sig23 <- garch.sig2(1, eta, beta, eps3, eps20[3], sig20)
  sig24 <- garch.sig2(1, eta, beta, eps4, eps20[4], sig20)
  sig25 <- garch.sig2(1, eta, beta, eps5, eps20[5], sig20)

  if(any(sig21 < 0)) return(-Inf) #Make sure all sigt2 is positive
  if(any(sig22 < 0)) return(-Inf)
  if(any(sig23 < 0)) return(-Inf)
  if(any(sig24 < 0)) return(-Inf)
  if(any(sig25 < 0)) return(-Inf)

  omegah1 <- mean(eps1^2/sig21)
  omegah2 <- mean(eps2^2/sig22)
  omegah3 <- mean(eps3^2/sig23)
  omegah4 <- mean(eps4^2/sig24)
  omegah5 <- mean(eps5^2/sig25)
  d1 <- -.5 * (n + n*log(omegah1) + sum(log(sig21)))
  d2 <- -.5 * (n + n*log(omegah2) + sum(log(sig22)))
  d3 <- -.5 * (n + n*log(omegah3) + sum(log(sig23)))
  d4 <- -.5 * (n + n*log(omegah4) + sum(log(sig24)))
  d5 <- -.5 * (n + n*log(omegah5) + sum(log(sig25)))
  d1+d2+d3+d4+d5
}

#' Computes Sigma^2 (volatilities) at time t from garch observations
```

```r
#' Credit to Prof Lysy
#' @param omega, alpha, beta GARCH parameters.
#' @param eps vector of GARCH observations
#' @param eps20, sig20 initial values.
#' @return sig^2
garch.sig2 <- function(omega, alpha, beta, eps, eps20, sig20) {
  n <- length(eps)
  eps2 <- eps^2
  mu <- mean(eps2)
  if(missing(eps20)) eps20 <- mu
  if(missing(sig20)) sig20 <- mu
  fsig2 <- omega + alpha * c(eps20, eps2[-n]) #First part of the sigma^2 equation
  sig2 <-as.numeric(filter(fsig2, beta, "recursive", init = sig20))
  sig2
}


#' Closed-form estimator by Kristensen & Linton (using notation from the paper)
#'
#' @param eps GARCH observation parameter
#' @return vector of estimated GARCH parameters (alpha, beta, omega).
garch.clo <- function(eps) {
  n <- length(eps)
  xt <- eps*eps
  sigma2 <- mean(xt)
  rho <- cov(xt[1:(n-1)],xt[2:n])
  phi <- cov(xt[1:(n-2)], xt[3:n])/rho
  b <-  (phi^2 + 1 - 2*rho*phi)/(phi - rho)
  theta <- (-b + sqrt(b^2 -4))/2
  alpha <- theta + phi
  beta <- -theta
  omega <- sigma2*(alpha+beta)
  c(omega = omega, alpha = alpha, beta = beta)
}
```

3. garchtest.R

```r
garch.dforecast <- function(yt1, yt2, yt3, yt4, yt5, R, N){
  daily1 <- garch.fit(yt1)
  daily2 <- garch.fit(yt2)
  daily3 <- garch.fit(yt3)
  daily4 <- garch.fit(yt4)
  daily5 <- garch.fit(yt5)

  theta.mle <- daily1$theta
  omega <- c(daily1$theta['omega'],daily2$theta['omega'], daily3$theta['omega'],
```

```
          daily4$theta['omega'], daily5$theta['omega'])
  alpha <- c(daily1$theta['alpha'], daily2$theta['alpha'], daily3$theta['alpha'],
          daily4$theta['alpha'], daily5$theta['alpha'])
  beta <- c(daily1$theta['beta'], daily2$theta['beta'], daily3$theta['beta'],
         daily4$theta['beta'], daily5$theta['beta'])

  mu <- c(daily1$mu, daily2$mu, daily3$mu, daily4$mu, daily5$mu)
  if (R == 'I'){
    R <- diag(length(mu)) #Assuming Z ~ MVN(0, I)
  } else{
    Zt <- cbind(daily1$zt, daily2$zt, daily3$zt, daily4$zt, daily5$zt)
    R <- cor(Zt)
  }
  sig20 <- c(daily1$sig2N, daily2$sig2N, daily3$sig2N, daily4$sig2N, daily5$sig2N)
  eps0 <- c(daily1$epsN, daily2$epsN, daily3$epsN, daily4$epsN, daily5$epsN)
  y <-  cbind(yt1, yt2, yt3, yt4, yt5)
  y0 <- as.numeric(tail(y,n=1))

  price <- garch.sim(N, omega, alpha, beta, mu, R, sig20, eps0, y0)
  price
}

garch.wforecast <- function(wt1, wt2, wt3, wt4, wt5, R, N){
  weekly1 <- garch.fitw(wt1)
  weekly2 <- garch.fitw(wt2)
  weekly3 <- garch.fitw(wt3)
  weekly4 <- garch.fitw(wt4)
  weekly5 <- garch.fitw(wt5)

  omega <- c(weekly1$theta['omega'],weekly2$theta['omega'], weekly3$theta['omega'],
          weekly4$theta['omega'], weekly5$theta['omega'])
  alpha <- c(weekly1$theta['alpha'], weekly2$theta['alpha'], weekly3$theta['alpha'],
          weekly4$theta['alpha'], weekly5$theta['alpha'])
  beta <- c(weekly1$theta['beta'], weekly2$theta['beta'], weekly3$theta['beta'],
         weekly4$theta['beta'], weekly5$theta['beta'])

  mu <- c(weekly1$mu, weekly2$mu, weekly3$mu, weekly4$mu, weekly5$mu)
  if (R == 'I'){
    R <- diag(length(mu)) #Assuming Z ~ MVN(0, I)
  } else{
    Zt <- cbind(weekly1$zt, weekly2$zt, weekly3$zt, weekly4$zt, weekly5$zt)
    R <- cor(Zt)
  }
  sig20 <- c(weekly1$sig2N, weekly2$sig2N, weekly3$sig2N, weekly4$sig2N, weekly5$sig2N)
  eps0 <- c(weekly1$epsN, weekly2$epsN, weekly3$epsN, weekly4$epsN, weekly5$epsN)
```

```
  y <-  cbind(yt1, yt2, yt3, yt4, yt5)
  y0 <- as.numeric(tail(y,n=1))

  price <- garch.sim(N, omega, alpha, beta, mu, R, sig20, eps0, y0)
  price
}

garch.rforecast <- function(yt1, yt2, yt3, yt4, yt5, R, N){
  n <- length(yt1)
  rt11 <- yt1[seq(1, n, 5)]
  rt12 <- yt1[seq(2, n, 5)]
  rt13 <- yt1[seq(3, n, 5)]
  rt14 <- yt1[seq(4, n, 5)]
  rt15 <- yt1[seq(5, n, 5)]

  rt21 <- yt2[seq(1, n, 5)]
  rt22 <- yt2[seq(2, n, 5)]
  rt23 <- yt2[seq(3, n, 5)]
  rt24 <- yt2[seq(4, n, 5)]
  rt25 <- yt2[seq(5, n, 5)]

  rt31 <- yt3[seq(1, n, 5)]
  rt32 <- yt3[seq(2, n, 5)]
  rt33 <- yt3[seq(3, n, 5)]
  rt34 <- yt3[seq(4, n, 5)]
  rt35 <- yt3[seq(5, n, 5)]

  rt41 <- yt4[seq(1, n, 5)]
  rt42 <- yt4[seq(2, n, 5)]
  rt43 <- yt4[seq(3, n, 5)]
  rt44 <- yt4[seq(4, n, 5)]
  rt45 <- yt4[seq(5, n, 5)]

  rt51 <- yt5[seq(1, n, 5)]
  rt52 <- yt5[seq(2, n, 5)]
  rt53 <- yt5[seq(3, n, 5)]
  rt54 <- yt5[seq(4, n, 5)]
  rt55 <- yt5[seq(5, n, 5)]

  roll1 <- garch.fitr(rt11, rt12, rt13, rt14, rt15)
  roll2 <- garch.fitr(rt21, rt22, rt23, rt24, rt25)
  roll3 <- garch.fitr(rt31, rt32, rt33, rt34, rt35)
  roll4 <- garch.fitr(rt41, rt42, rt43, rt44, rt45)
  roll5 <- garch.fitr(rt51, rt52, rt53, rt54, rt55)
```

```
omega <- c(roll1$theta['omega'],roll2$theta['omega'], roll3$theta['omega'],
         roll4$theta['omega'], roll5$theta['omega'])
alpha <- c(roll1$theta['alpha'], roll2$theta['alpha'], roll3$theta['alpha'],
         roll4$theta['alpha'], roll5$theta['alpha'])
beta <- c(roll1$theta['beta'], roll2$theta['beta'], roll3$theta['beta'],
        roll4$theta['beta'], roll5$theta['beta'])

mu <- cbind(roll1$mu, roll2$mu, roll3$mu, roll4$mu, roll5$mu)
if (R == 'I'){
  R <- diag(length(mu)*5) #Assuming Z ~ MVN(0, I)
  emp <- FALSE
} else if(R=='F'){
  Zt <- cbind(roll1$zt1, roll1$zt2, roll1$zt3, roll1$zt4, roll1$zt5,
              roll2$zt1, roll2$zt2, roll2$zt3, roll2$zt4, roll2$zt5,
              roll3$zt1, roll3$zt2, roll3$zt3, roll3$zt4, roll3$zt5,
              roll4$zt1, roll4$zt2, roll4$zt3, roll4$zt4, roll4$zt5,
              roll5$zt1, roll5$zt2, roll5$zt3, roll5$zt4, roll5$zt5)
  R <- cor(Zt)
  emp <-FALSE
}else{
  Zt <- cbind(roll1$zt1, roll1$zt2, roll1$zt3, roll1$zt4, roll1$zt5,
              roll2$zt1, roll2$zt2, roll2$zt3, roll2$zt4, roll2$zt5,
              roll3$zt1, roll3$zt2, roll3$zt3, roll3$zt4, roll3$zt5,
              roll4$zt1, roll4$zt2, roll4$zt3, roll4$zt4, roll4$zt5,
              roll5$zt1, roll5$zt2, roll5$zt3, roll5$zt4, roll5$zt5)
  Ztrans <- apply(Zt, 2, ztrans)
  R <- cor(Ztrans)
  emp <- TRUE
}
sig20 <- rbind(roll1$sig2N, roll2$sig2N, roll3$sig2N, roll4$sig2N, roll5$sig2N)
eps0 <- rbind(roll1$epsN, roll2$epsN, roll3$epsN, roll4$epsN, roll5$epsN)
y <-  cbind(yt1, yt2, yt3, yt4, yt5)
y0 <- as.numeric(tail(y,n=1))


if (N == 1) {
  simresult <- garch.rsim(N, omega, alpha, beta, mu, R, sig20, eps0, y0, emp, Zt)
  price <- simresult[["price"]]
  var <- simresult[["var"]]
  list(forecast = price, var = var)
} else {
  price <- garch.rsim(N, omega, alpha, beta, mu, R, sig20, eps0, y0, emp, Zt)
  price[,N]
```

```
  }
}
```

4.mle-check.R (Credit to Prof Lysy)

```
#' Numerical check for MLE

#'

#' Given a log-likelihood function and a potential MLE, checks whether each one-dimensional version of the log-likelihood which varies one parameter at a time with all others at the MLE is indeed maximized at the MLE value.

#' @param loglik loglikelihood function.  Takes a single vector argument.

#' @param theta.mle The potential MLE.

#' @param itheta indices of one dimensional functions to evaluate and plot.  Defaults to all parameters.

#' @param theta.names Optional vector of parameter names for plotting.

#' @param theta.rng Optional two-row matrix giving the plotting limits for each parameter. Defaults to theta.mle +/- .5 * abs(theta.mle)

#' @param refit If \code{TRUE}, recalculates the range so that drop is more or less the same on either side of \code{theta.mle}.

#' @param layout Optional vector giving the number of rows and columns in the plot.  For \code{p} parameters, defaults to \code{c(nr, nc)}, where \code{nr = floor(p)} and \code{nc = ceiling(p/nr)}.

#' @return Invisibly returns \code{NULL}.  The purpose of this function is to plot the one-dimensional log-likelihoods.

mle.check <- function(loglik, theta.mle, itheta, theta.names, theta.rng,
                      refit = FALSE, layout) {
  ntheta <- length(theta.mle) # number of parameters
  if(missing(itheta)) itheta <- 1:ntheta
  if(is.logical(itheta)) itheta <- which(itheta) # convert T/F's to indices
  if(missing(theta.names)) {
    theta.names <- paste0("theta[",1:ntheta,"]")
    # converts to expression so symbol "theta_i" is plotted
```

```
  theta.names <- parse(text = theta.names)
}
if(missing(theta.rng)) {
  theta.rng <- rbind(theta.mle - .5 * abs(theta.mle),
                     theta.mle + .5 * abs(theta.mle))
}
# shorten theta.names and theta.rng if necessary
ntheta2 <- length(itheta)
if(length(theta.names) > ntheta2) theta.names <- theta.names[itheta]
if(ncol(theta.rng) > ntheta2) theta.rng <- theta.rng[,itheta]
# set up plot
opar <- par(no.readonly = TRUE) # save specs of current plot
# plot size
if(missing(layout)) {
  layout <- floor(sqrt(ntheta2))
  layout <- c(layout, ceiling(ntheta2/layout))
}
# for loop for plotting
par(mfrow = layout, mar = c(2,2.5,2.5,0), oma = c(3, 3, .5, .5))
for(ii in 1:ntheta2) {
  ith <- itheta[ii]
  theta.seq <- seq(from = theta.rng[1,ii],
                   to = theta.rng[2,ii], len = 200)
  for(jj in 1:2) {
    # evaluate likelihood fixing all components except one
    theta.ll <- sapply(theta.seq, function(thetai) {
```

```
    theta <- theta.mle
    theta[ith] <- thetai
    loglik(theta)
  })
  if(refit) {
    # browser()
    vth <- !is.na(theta.ll) & theta.ll > -Inf # valid values
    lth <- theta.seq < theta.mle[ith] # on the left of mle
    rth <- theta.seq > theta.mle[ith] # on the right
    # larger of the min value on each size
    lbd <- max(min(theta.ll[vth & lth]), min(theta.ll[vth & rth]))
    # rescale theta.seq to be on this range
    ibd <- c(which.min(ifelse(vth & lth, abs(theta.ll-lbd), Inf)),
             which.min(ifelse(vth & rth, abs(theta.ll-lbd), Inf)))
    theta.seq <- seq(theta.seq[ibd[1]], theta.seq[ibd[2]], len = 200)
  } else break
 }
 # plot loglikelihood and add estimated value
 plot(theta.seq, theta.ll, type = "l",
      xlab = "", ylab = "")
 title(main = theta.names[ii], cex.main = 2)
 abline(v = theta.mle[ith], col = "red")
}
# labels in margin
mtext(side = 2, text = "Log-Likelihood",
      line = 1, outer = TRUE)
```

```
  mtext(side = 1, text = "Parameter",
        line = 1, outer = TRUE)
  invisible(par(opar)) # restore plot parameters
}
```

5.garchcheck.R

```
#MLE Checks

source("mle-check.R")
source("garch.R")
source("garchRoll.R")

require(quantmod)

getSymbols(Symbols = c("LEA"),
           src = "yahoo", from = "1990-01-01")
dates <- c("2012-01-04", "2015-12-31")
yt <- LEA[paste0(dates, collapse = "/")]$LEA.Close


#Get eps from price
geteps <- function(yt){
  xt <- diff(log(yt))[-1] # log returns
  n <- length(xt)
  # convert to numeric format
  xt <- as.numeric(xt)
  muh <- mean(xt)
  #intial parameters
  eps <- xt - muh
  eps20 <- mean(eps^2)
  list(eps20=eps20, eps = eps)
}

#Rolling Geteps
rgeteps <-function(yt1, yt2, yt3, yt4, yt5){
  xt1 <- diff(log(yt1))[-1]
  xt2 <- diff(log(yt2))[-1]
  xt3 <- diff(log(yt3))[-1]
  xt4 <- diff(log(yt4))[-1]
  xt5 <- diff(log(yt5))[-1]# log returns
  n <- length(xt1)
```

```
  # convert to numeric format
  xt1 <- as.numeric(xt1)
  xt2 <- as.numeric(xt2)
  xt3 <- as.numeric(xt3)
  xt4 <- as.numeric(xt4)
  xt5 <- as.numeric(xt5)
  muh1 <- mean(xt1)
  muh2 <- mean(xt2)
  muh3 <- mean(xt3)
  muh4 <- mean(xt4)
  muh5 <- mean(xt5)
  muh <- mean(muh1, muh2, muh3, muh4, muh5)

  #intial parameters
  eps1 <- xt1 - muh
  eps2 <- xt2- muh
  eps3 <- xt3 - muh
  eps4 <- xt4- muh
  eps5 <- xt5 - muh
  eps20 <- rowMeans(rbind(eps1*eps1, eps2*eps2,
                    eps3*eps3, eps4*eps4, eps5*eps5), na.rm=TRUE)
  list(eps1=eps1, eps2=eps2, eps3=eps3, eps4 =eps4, eps5=eps5, eps20=eps20)
}

#GARCH Loglikihood function for weekly/daily data
loglik <- function(theta) {
  garch.loglik(omega = theta[1], alpha = theta[2], beta = theta[3],
            eps = eps, eps20 = eps20,
            sig20 = theta[1])
}

#GARCH Loglikihood function for rolling data
rloglik <- function(theta) {
  garch.rloglik(omega = theta[1], alpha = theta[2], beta = theta[3],
             eps1 = eps1, eps2=eps2, eps3=eps3, eps4=eps4, eps5=eps5,
             eps20,sig20 = theta[1])
}

#Daily data
eps <- geteps(yt)$eps
eps20 <- geteps(yt)$eps20
theta.mle <- garch.fit(yt)$theta

tnames <- expression(omega, alpha, beta)
#plot check
```

```
mle.check(loglik = loglik, theta.mle = theta.mle,
        refit = TRUE, theta.names = tnames) # pass

#Weekly Data
wt <- to.weekly(yt)$yt.Close
eps <- geteps(wt)$eps
eps20 <- geteps(wt)$eps20
theta.mle <- garch.fitw(wt)$theta

tnames <- expression(omega, alpha, beta)
#plot check
mle.check(loglik = loglik, theta.mle = theta.mle,
        refit = TRUE, theta.names = tnames) # pass

#Rolling data

n <- length(yt)
rt1 <- yt[seq(1, n, 5)]
rt2 <- yt[seq(2, n, 5)]
rt3 <- yt[seq(3, n, 5)]
rt4 <- yt[seq(4, n, 5)]
rt5 <- yt[seq(5, n, 5)]

eps <- rgeteps(rt1, rt2, rt3, rt4, rt5)
eps1 <- eps$eps1
eps2 <- eps$eps2
eps3 <- eps$eps3
eps4 <- eps$eps4
eps5 <- eps$eps5
eps20 <- eps$eps20

theta.mle <- garch.fitr(rt1, rt2, rt3, rt4, rt5)$theta

tnames <- expression(omega, alpha, beta)
#plot check
mle.check(loglik = rloglik, theta.mle = theta.mle,
        refit = TRUE, theta.names = tnames) # pass
```

6. SimulationCalibrate.R

```
#Daily 5-day Simulation
require(quantmod)
source("garchtest.R")
source("garch.R")
source("garchRoll.R")
```

```
getSymbols(Symbols = c("BIIB","MAN","LEA","DHI","NHTC"),
          src = "yahoo", from = "1990-01-01")
dates <- c("2012-01-01", "2016-02-29")
yt1 <- BIIB[paste0(dates, collapse = "/")]$BIIB.Close
yt2 <- MAN[paste0(dates, collapse = "/")]$MAN.Close
yt3 <- LEA[paste0(dates, collapse = "/")]$LEA.Close
yt4 <- DHI[paste0(dates, collapse = "/")]$DHI.Close
yt5 <- NHTC[paste0(dates, collapse = "/")]$NHTC.Close

#Real Prices
dates2 <- c("2016-03-01", "2016-03-30")
pt1 <- BIIB[paste0(dates2, collapse = "/")]$BIIB.Close
pt2 <- MAN[paste0(dates2, collapse = "/")]$MAN.Close
pt3 <- LEA[paste0(dates2, collapse = "/")]$LEA.Close
pt4 <- DHI[paste0(dates2, collapse = "/")]$DHI.Close
pt5 <- NHTC[paste0(dates2, collapse = "/")]$NHTC.Close

set.seed(100)
rprice <- pt3[5]
terror <- 0
sim <- 50
for (i in 1:sim){
  dayforecast <- garch.dforecast(yt1, yt2, yt3, yt4, yt5, 'F', 5)[3]
  terror <- terror + sum(abs(dayforecast - rprice))
}
merror <- terror/sim

#Weekly Forecast
wt1 <- to.weekly(yt1)$yt1.Close
wt2 <- to.weekly(yt2)$yt2.Close
wt3 <- to.weekly(yt3)$yt3.Close
wt4 <- to.weekly(yt4)$yt4.Close
wt5 <- to.weekly(yt5)$yt5.Close

set.seed(100)
terrorw <-0
for (i in 1:sim){
  weekforecast <- garch.wforecast(wt1, wt2, wt3, wt4, wt5, 'F', 1)[3]
  terrorw <- terrorw + sum(abs(weekforecast - rprice))
}
merrorw <- terrorw/sim

#Rolling

set.seed(100)
```

```
terrorr <- 0

for (i in 1:sim){
  rollforecast <- garch.rforecast(yt1, yt2, yt3, yt4, yt5, 'F', 5)[3]
  terrorr <- terrorr + sum(abs(rollforecast - rprice))
}
merrorr <- terrorr/sim
```

7. PortfolioManagement.R

```
require(quantmod)
source("garch.R")
source("garchRoll.R")
source("garchtest.R")

# Helpers
find_q0 <- function(garchresult_r, p0_r, cost, returnrate) {
  forecast_r <- garchresult_r[["forecast"]][1,]
  var_r <- garchresult_r[["var"]]
  prediction_r <- (forecast_r - cost) / cost

  # Get optim q0
  lambda_r <- returnrate / (t(prediction_r) %*% var_r %*% prediction_r)
  q_r <- c(t(var_r) %*% prediction_r) * coredata(lambda_r) # Expected trade percentage of
money on t0
  assets_r <- q_r * p0_r
  q0_r <- assets_r / cost

  # Data generation
  q0plus_r <- sapply(q0_r, function (x)  max(x, 0))
  q0minus_r <- sapply(q0_r, function (x) -min(x, 0))
  a0_r <- sum(q0plus_r * cost)
  if (a0_r > p0_r) {
    # If such situation happened, liquidate all assets
    q0_r <- rep(0, 5)
  } else if (sum(q0minus_r * cost) > (p0_r - a0_r)) {
    # If such situation happened, liquidate all assets
    q0_r <- q0plus_r
  }
  q0_r
}

update_p0 <- function(q0, cost, new_cost, p0) {
  diff <- q0 * (new_cost - cost)
  # Update portfolio info and data
```

```
  p0 <- p0 + sum(diff)
  p0
}

simulation <- function(returnrate) {
  start_date = "2012-01-04"
  end_date = "2015-12-31"
  getSymbols(Symbols = c("BIIB","MAN","LEA","DHI","NHTC"),
             src = "yahoo", from = start_date)
  dates <- c(start_date, end_date)

  # Stock Data Init
  yt1 <- BIIB[paste0(dates, collapse = "/")]$BIIB.Close
  yt2 <- MAN[paste0(dates, collapse = "/")]$MAN.Close
  yt3 <- LEA[paste0(dates, collapse = "/")]$LEA.Close
  yt4 <- DHI[paste0(dates, collapse = "/")]$DHI.Close
  yt5 <- NHTC[paste0(dates, collapse = "/")]$NHTC.Close
  N <- length(yt1)
  cost <- sapply(c(yt1[N,],yt2[N,],yt3[N,],yt4[N,],yt5[N,]), coredata)[,1] # Price at t0

  # Init portfolio and assets
  q0_r <- q0_i <- q0_f <- rep(0, 5)
  p0_r <- p0_i <- p0_f <- 1000000
  for (t in 1:252) {
    # Predict the new price and get the trading amount
    garchresult_r <- garch.rforecast(yt1, yt2, yt3, yt4, yt5, 'R', 1)
    garchresult_i <- garch.rforecast(yt1, yt2, yt3, yt4, yt5, 'I', 1)
    garchresult_f <- garch.rforecast(yt1, yt2, yt3, yt4, yt5, 'F', 1)


    q0_r <- find_q0(garchresult_r, p0_r, cost, returnrate)
    q0_i <- find_q0(garchresult_i, p0_i, cost, returnrate)
    q0_f <- find_q0(garchresult_f, p0_f, cost, returnrate)

    # Get market value
    newtime <- N+t
    actualprice1 <- BIIB[newtime,]$BIIB.Close
    actualprice2 <- MAN[newtime,]$MAN.Close
    actualprice3 <- LEA[newtime,]$LEA.Close
    actualprice4 <- DHI[newtime,]$DHI.Close
    actualprice5 <- NHTC[newtime,]$NHTC.Close
    new_cost <-
sapply(c(actualprice1,actualprice2,actualprice3,actualprice4,actualprice5),coredata)[,1]

    # Update portfolio info and data
```

```
    p0_r <- update_p0(q0_r, cost, new_cost, p0_r)
    p0_i <- update_p0(q0_i, cost, new_cost, p0_i)
    p0_f <- update_p0(q0_f, cost, new_cost, p0_f)
    cost <- new_cost

    # Update data
    yt1 <- rbind(yt1[2:N,], actualprice1)
    yt2 <- rbind(yt2[2:N,], actualprice2)
    yt3 <- rbind(yt3[2:N,], actualprice3)
    yt4 <- rbind(yt4[2:N,], actualprice4)
    yt5 <- rbind(yt5[2:N,], actualprice5)
  }
  print(returnrate)
  print(p0_r/1000000 - 1)
  print(p0_i/1000000 - 1)
  print(p0_f/1000000 - 1)
}

for (i in seq(from=0.04, to=0.011, by=0.001)) {
  simulation(i)
}
```

**Reference**

Jalira Namugaya, Patrick G. O. Weke, W. M. Charles (2014). Modelling Volatility of Stock Returns: Is GARCH(1,1) enough? International Journal of Sciences: Basic and Applied Research (IJSBAR), Vol 16, No 2.

Robert F Engle and Andrew J Patton (2001). What good is a volatility model? Quantitative Finance, Volume 1,237–245

David Ardia and Lennart Hoogerheideb (2013). GARCH Models for Daily Stock Returns: Impact of Estimation Frequency on Value-at-Risk and Expected Shortfall Forecasts. Tinbergen Institute Discussion Paper, TI 2013-047/III.

Rob Reider (2009). Volatility Forecasting I: GARCH Models.

Petra Posedel (2005). Properties and Estimation of GARCH(1,1) Model. Metodoloˇski zvezki, Vol. 2, No. 2, 2005, 243-257.

Dennis Kristensen and Oliver Linton (2006). A Closed-Form Estimator for the GARCH(1,1) Model. Econometric Theory, Vol. 22, No. 2, pp. 323-337

Elton, E. J., Gruber, M. J., Brown, S. J., & Goetzmann, W. N. (2009). Modern portfolio theory and investment analysis. John Wiley & Sons.

Greenblatt, J. (2010). The Little Book that Beats the Market. Hoboken, NJ: Wiley.

PK. (2016, December 31). 2016 S&P 500 Return. Retrieved from https://dqydj.com/2016-sp-500-return/