# Time Series Analysis

Homework of week 14

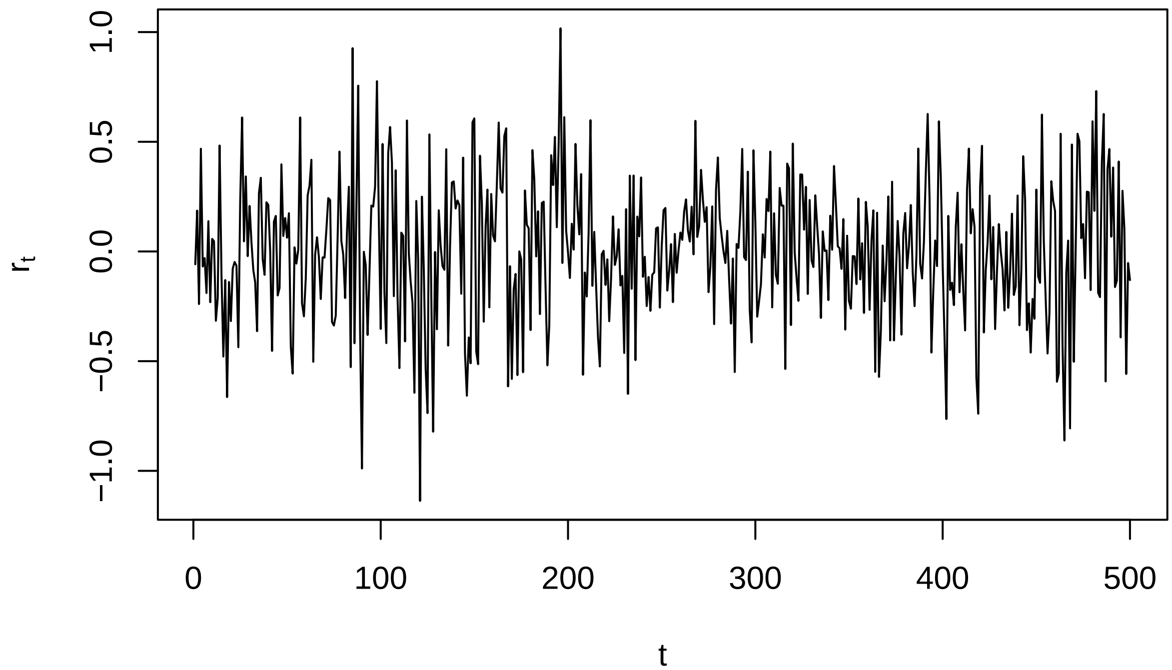
Group member: Hongjin Du 11911132; Simin Du 11912231; Hanbin Liu 11912410

## 12.7

```
library(TSA)
```

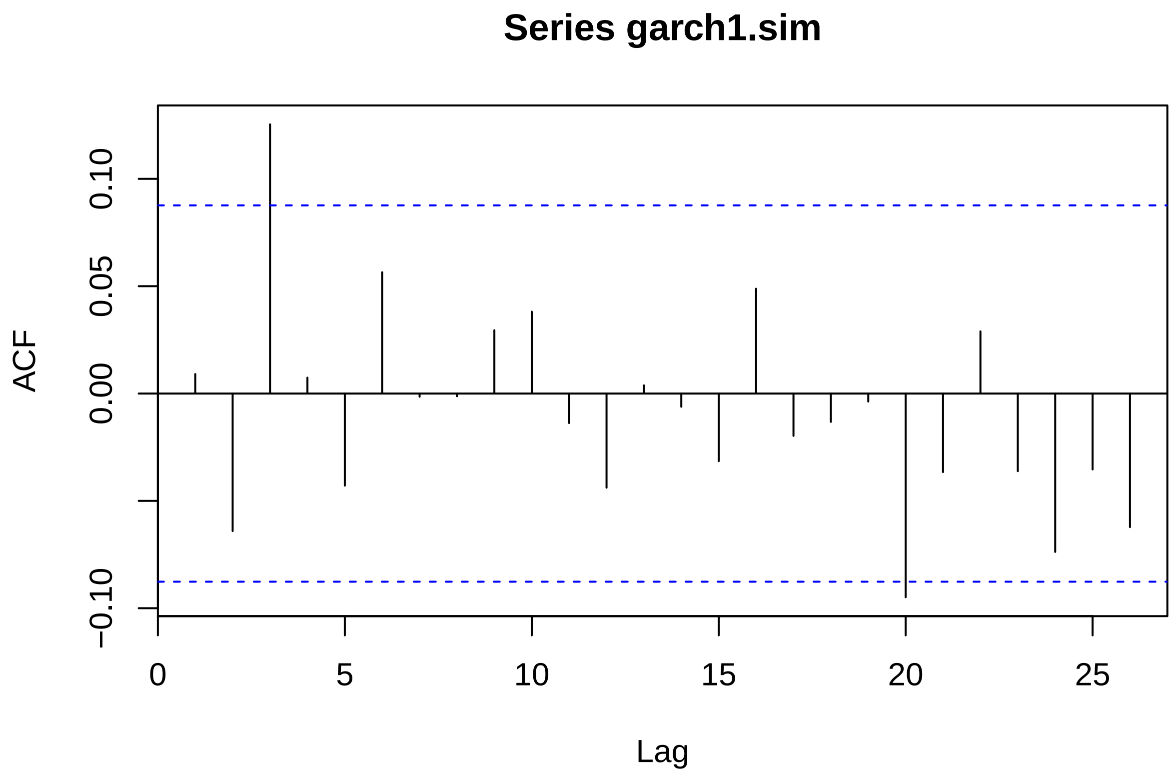
```
##
##    'TSA'

## The following objects are masked from 'package:stats':
##
##     acf, arima

## The following object is masked from 'package:utils':
##
##     tar
```

```
set.seed(1234567)
garch1.sim <- garch.sim(alpha=c(0.01,0.1),beta=0.8,n=500)
plot(garch1.sim,type='l',ylab=expression(r[t]),xlab='t')
```

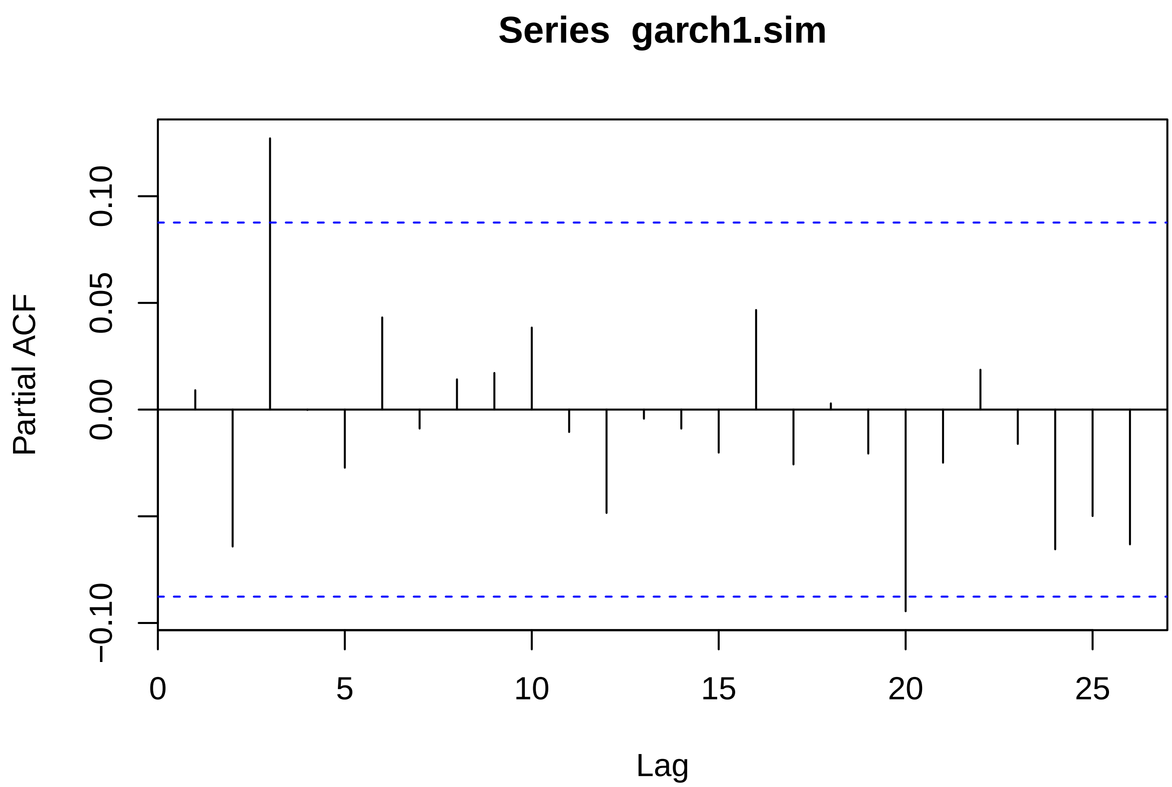



Let us see the sample ACF, PACF, and EACF of the simulated time series.

```
acf(garch1.sim)
```




```
pacf(garch1.sim)
```




```r
eacf(garch1.sim)
```

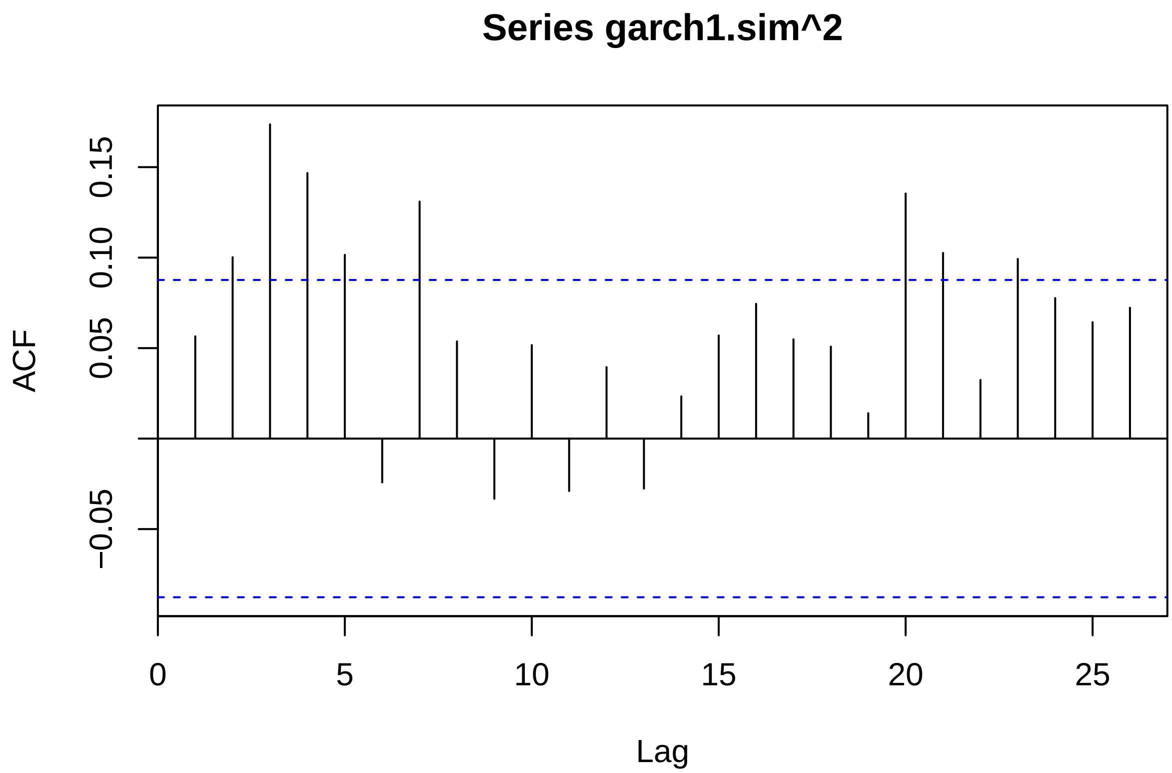
```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o o x o o o o o o o o  o  o  o
## 1 x o x o o o o o o o o  o  o  o
## 2 x x x o o o o o o o o  o  o  o
## 3 o x x o o o o o o o o  o  o  o
## 4 o o x o o o o o o o o  o  o  o
## 5 x x x o o o o o o o o  o  o  o
## 6 x x x o o o o o o o o  o  o  o
## 7 x x o x o o x o o o o  o  o  o
```

Except for lag 3 and 20 which are mildly significant, the sample ACF and PACF of the simulated data do not show significant correlations. Also, the pattern in the EACF table seems suggest an AR(3) model. Hence, the simulated process seems consistent with the assumption of white noise.

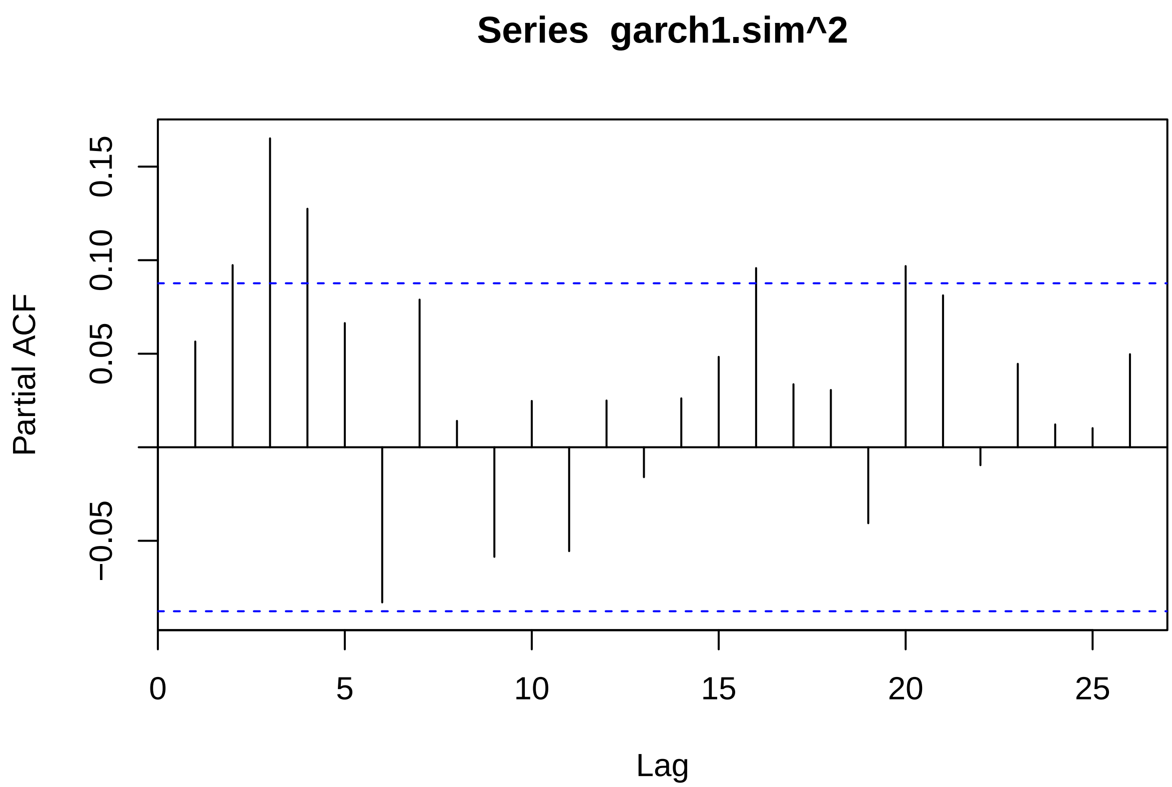
### (a)

Let us see the sample ACF, PACF, and EACF of the squared simulated GARCH(1,1) time series.

```r
acf(garch1.sim ^ 2)
```




```
pacf(garch1.sim ^ 2)
```




```r
eacf(garch1.sim ^ 2)
```

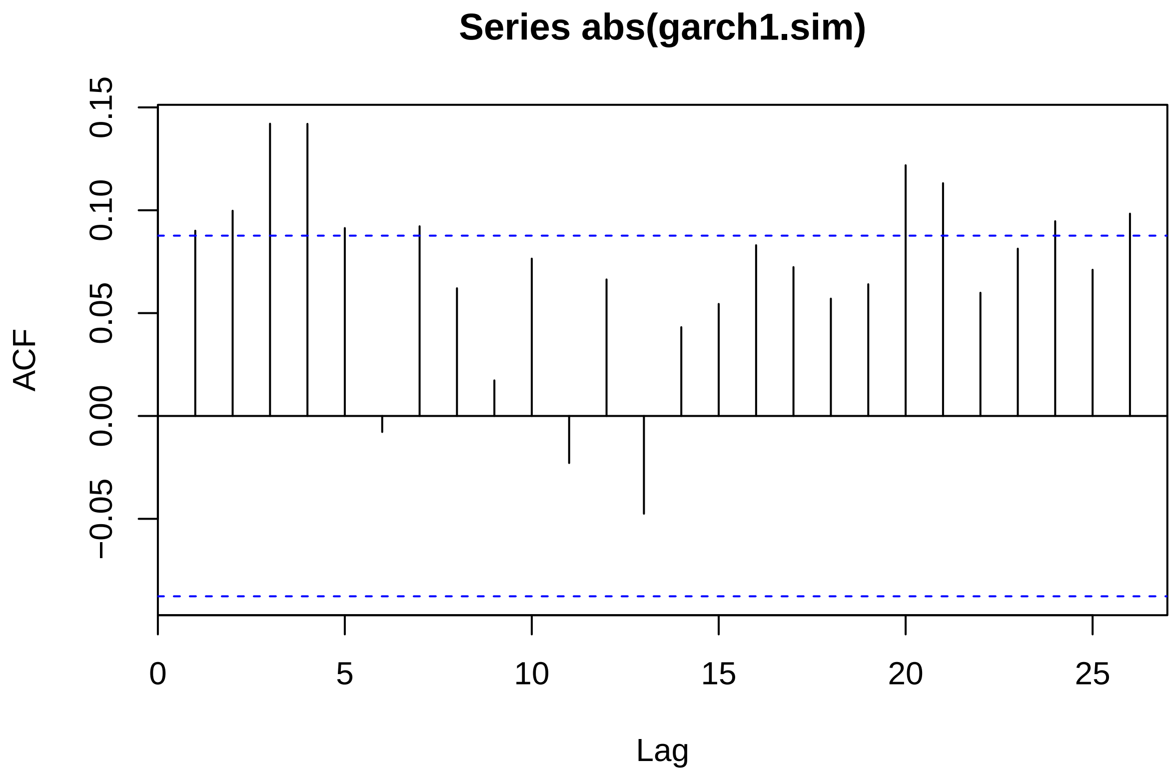
```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o x x x x o x o o o o  o  o  o
## 1 x o o o x o x x o o o  o  o  o
## 2 x o o o o x x o x o o  o  o  o
## 3 x x x o o x o o o o o  o  o  o
## 4 x x x x x o o o o o o  o  o  o
## 5 x o x o o o o o o o o  o  o  o
## 6 x x x x o x o o o o o  o  o  o
## 7 x x o x o x o o o o o  o  o  o
```

The sample ACF and PACF of the squared simulated GARCH(1,1) process show significant autocorrelation pattern in the squared data. Hence the simulated process is serially dependent as it is. But the pattern in the EACF table is not very clear. An ARMA(3,3) model is kind of suggested. As mentioned in the textbook, the fuzziness of the signal in the EACF table is likely caused by the larger sampling variability when we deal with higher moments.

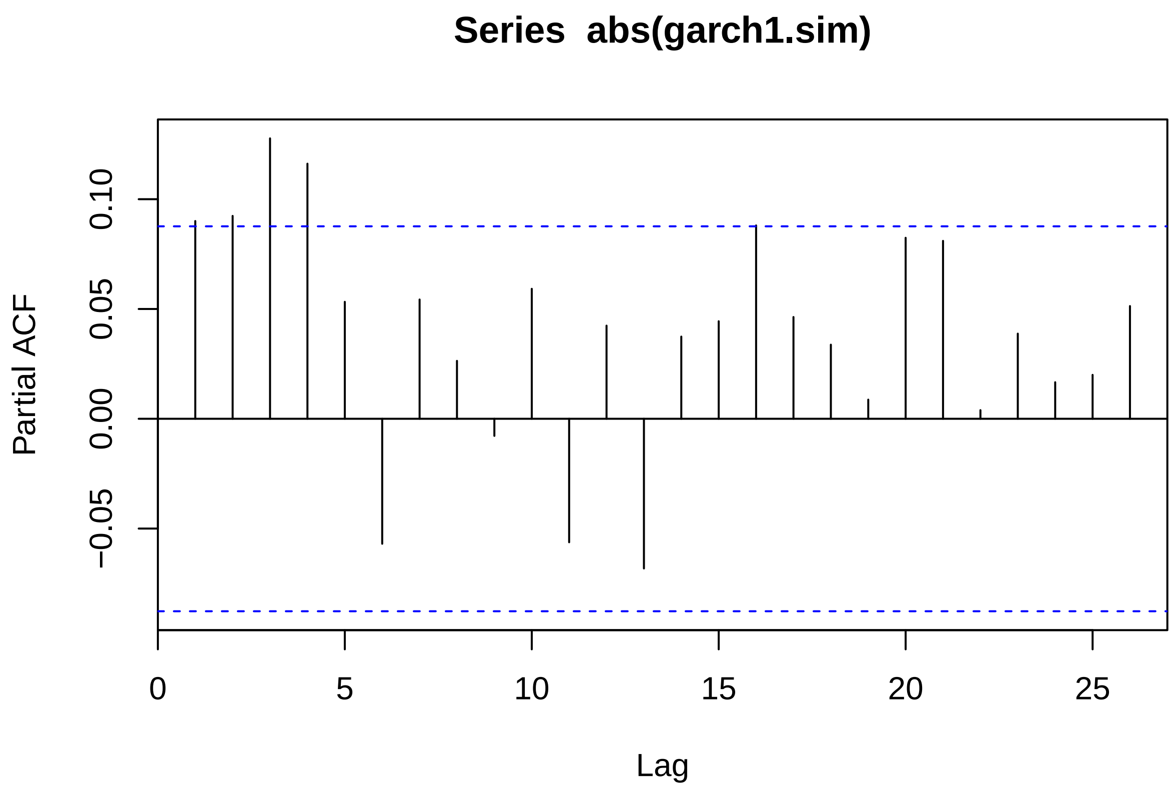
### (b)

Let us see the sample ACF, PACF, and EACF of the absolute values of the simulated GARCH(1,1) time series.

```
acf(abs(garch1.sim))
```




```
pacf(abs(garch1.sim))
```



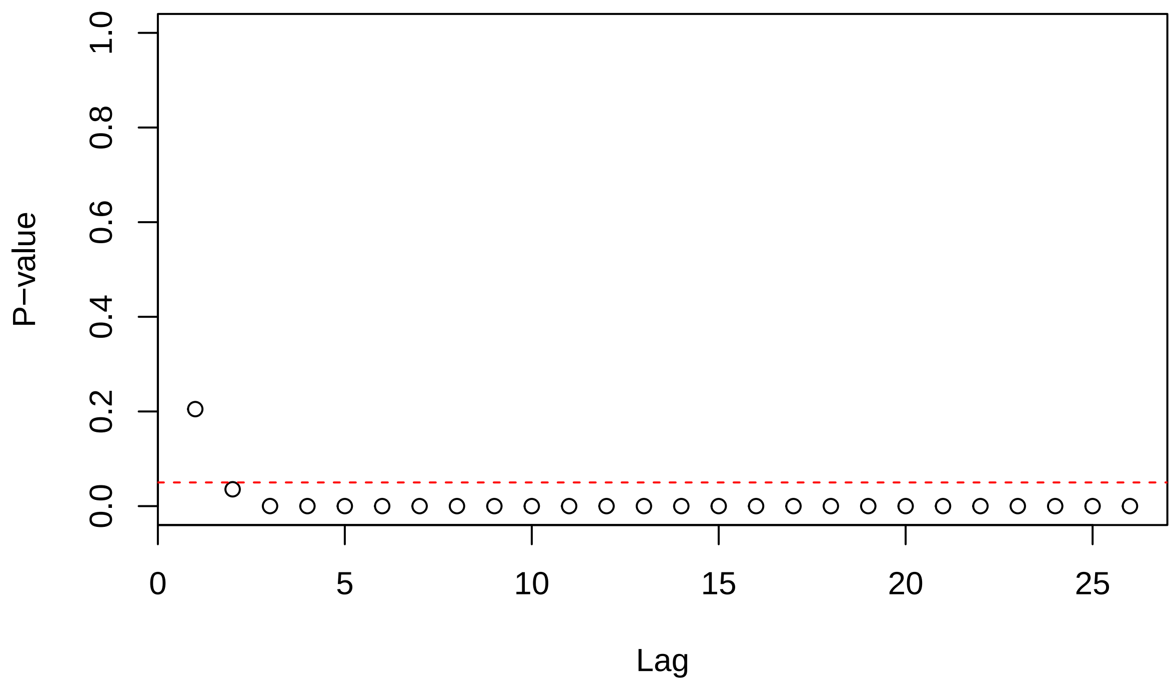

```
eacf(abs(garch1.sim))
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x o x o o o o  o  o  o
## 1 x o o o x o o o o o o  o  o  o
## 2 x x o o o o o o o o o  o  o  o
## 3 x x o o o x o o o o o  o  o  o
## 4 x x o x o x o o o o o  o  o  o
## 5 x o x x x o o o o o o  o  o  o
## 6 x o x o x x o o o o o  o  o  o
## 7 x x x x x o x o o o o  o  o  o
```

The sample EACF table for the absolute simulated process suggests convincingly an ARMA(1,1) model, and therefore a GARCH(1,1) model for the original data.

**(c)**

```
McLeod.Li.test(y = garch1.sim)
```

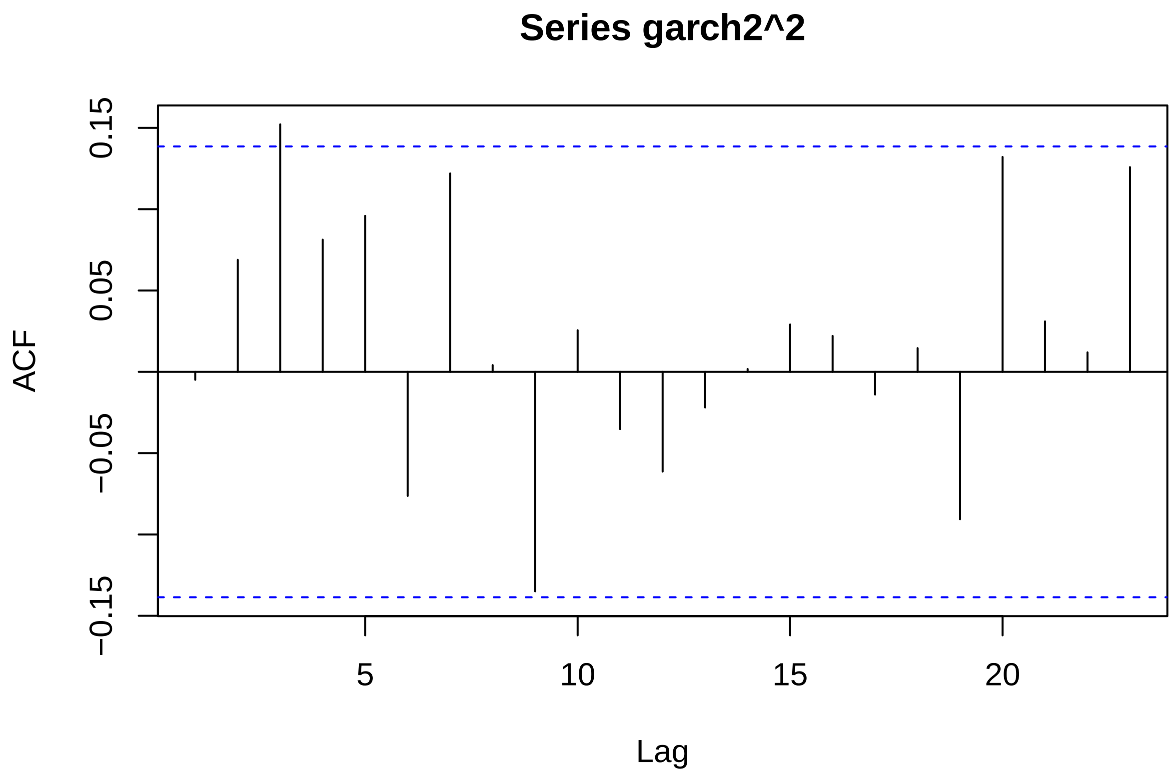



The McLeod-Li test shows the presence of strong ARCH effects in the data, as we know there are.

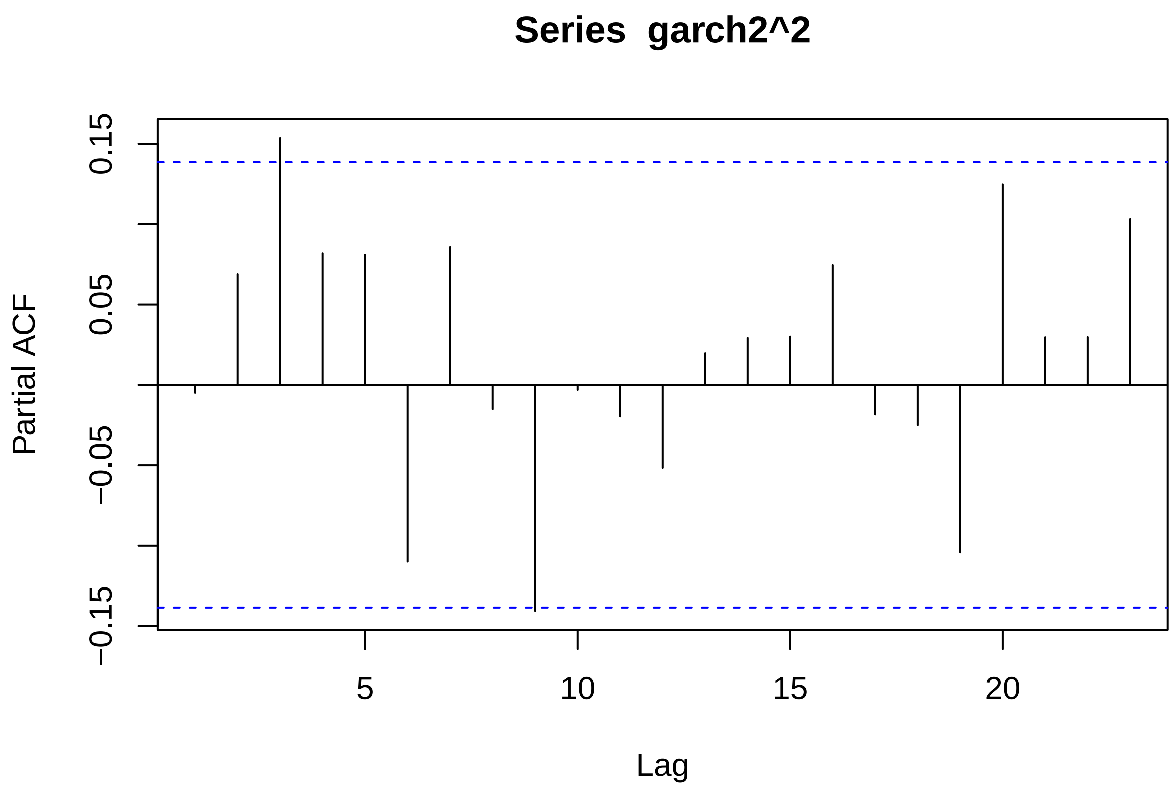
**(d)**

Let us see the sample ACF, PACF, and EACF of the squared GARCH(1,1) time series using only the first 200 simulated data.

```
garch2 <- garch1.sim[1:200]

acf(garch2 ^ 2)
```




```
pacf(garch2 ^ 2)
```



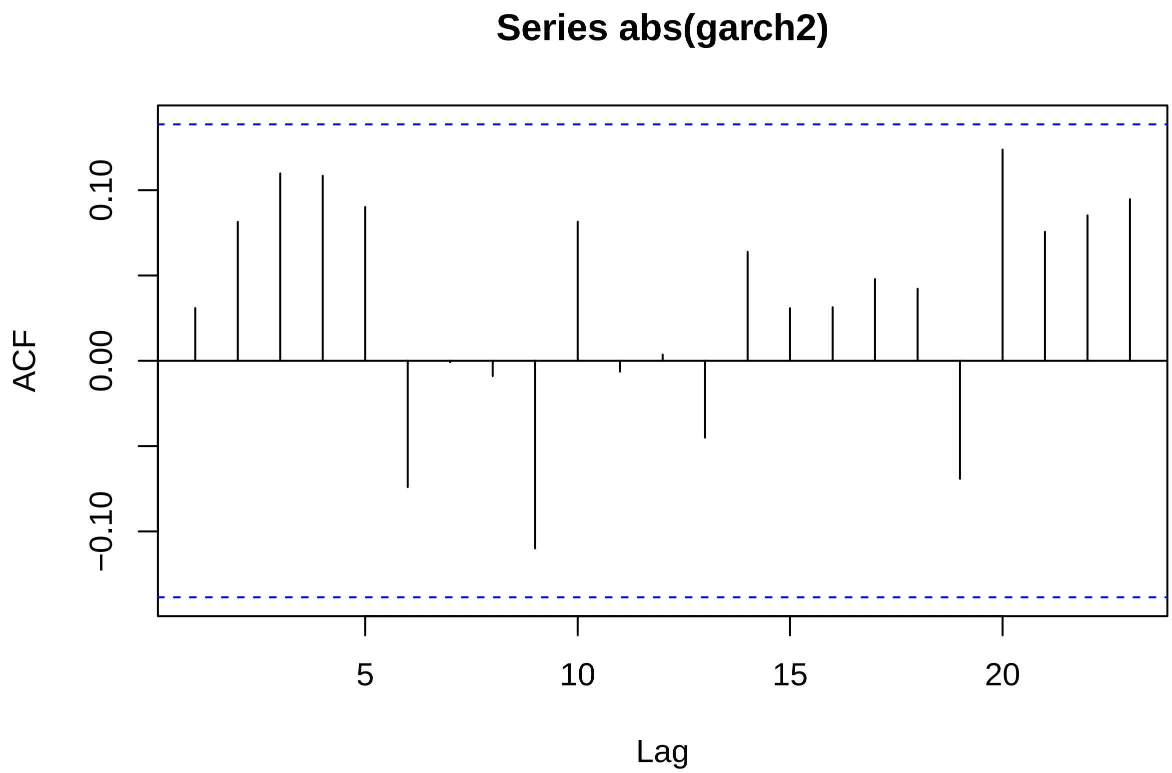

```
eacf(garch2 ^ 2)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o o x o o o o o o o o  o  o  o
## 1 o o o o o o o o o o o  o  o  o
## 2 x x o o o o o o x o o  o  o  o
## 3 x x x o o o o o o o o  o  o  o
## 4 x o x o o o o o o o o  o  o  o
## 5 x x o o x o o o o o o  o  o  o
## 6 x x o o x x o o o o o  o  o  o
## 7 x o o x o x o o o o o  o  o  o
```

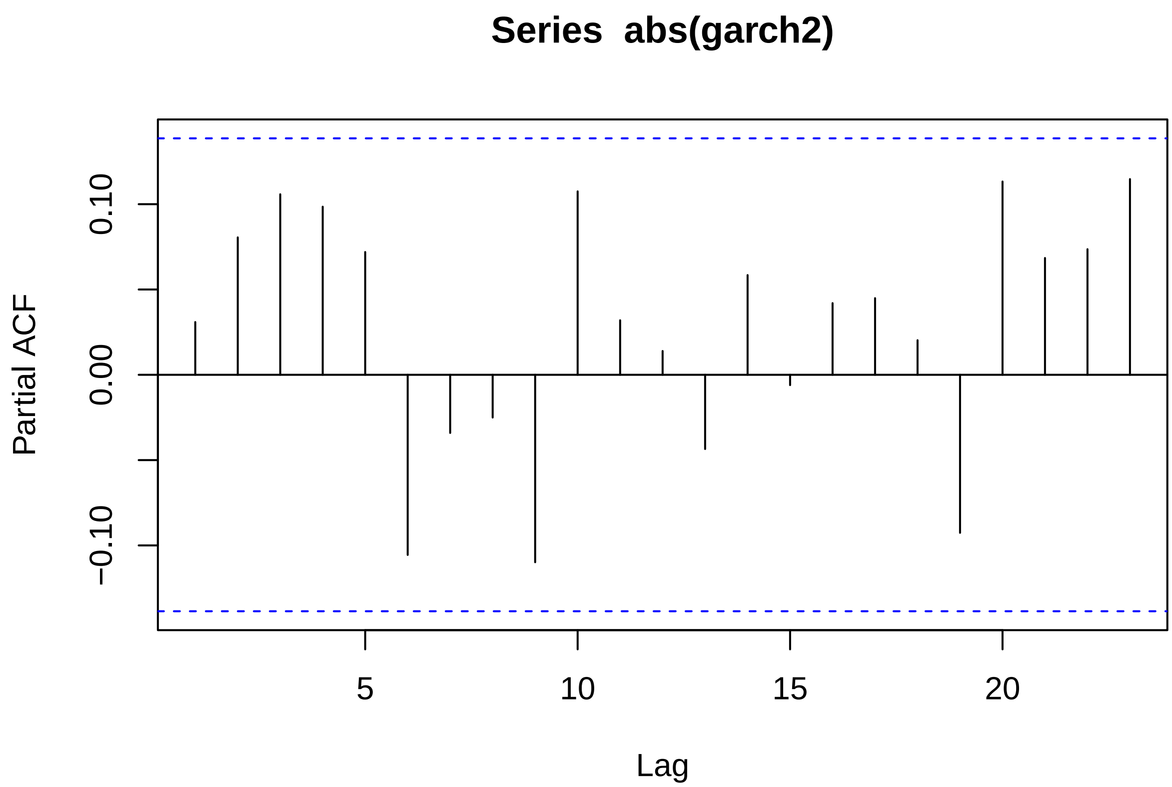
The plots of the ACF and PACF for the first 200 squared simulated data show no significant autocorrelations except for lags 3 and 9. Also, the EACF table seems to suggest the 200 squared data are white noise.

Then let us see the sample ACF, PACF, and EACF of the absolute values of the GARCH(1,1) time series also using the first 200 simulated data.

```
acf(abs(garch2))
```




```
pacf(abs(garch2))
```




```
eacf(abs(garch2))
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o o o o o o o o o o o  o  o  o
## 1 x o o o o o o o o o o  o  o  o
## 2 x x o o o o o o o o o  o  o  o
## 3 x o o o o o o o o o o  o  o  o
## 4 x o o o o o o o o o o  o  o  o
## 5 x x x x x o o o o o o  o  o  o
## 6 x o x x x o o o o o o  o  o  o
## 7 x o x x o o o o o o o  o  o  o
```

The plots of the ACF and PACF for the first 200 absolute simulated data show no significant autocorrelations. In addition, the EACF table convincingly suggests that the 200 squared data are white noise.

So a GARCH($p, q$) time series with size 200 is not enough for us to identify the orders $p$ and $q$ by inspecting its ACF, PACF and EACF.

## 2

### (i)

Reference:
Bollerslev, T. (1987). A Conditionally Heteroskedastic Time Series Model for Speculative Prices and Rates of Return. The Review of Economics and Statistics, 69(3), 542–547. https://doi.org/10.2307/1925546

Based on the reference, here $\epsilon_t$ should follows a standardized $t$-distribution. The pdf of $\epsilon_t$ is

$$\frac{\Gamma(\frac{d+1}{2})}{\sqrt{\pi(d-2)}\Gamma(\frac{d}{2})}\left[1+\frac{\epsilon_t^2}{d-2}\right]^{-\frac{d+1}{2}}$$

so that the pdf of $x_t$ is given by

$$\frac{\Gamma(\frac{d+1}{2})}{\sqrt{\pi(d-2)}\Gamma(\frac{d}{2})}\cdot\frac{1}{\sigma_t}\left[1+\frac{x_t^2}{\sigma_t^2(d-2)}\right]^{-\frac{d+1}{2}}$$

Hence, the log-likelihood function is given by

$$l(\theta)=n\log\left[\frac{\Gamma(\frac{d+1}{2})}{\sqrt{\pi(d-2)}\Gamma(\frac{d}{2})}\right]-\frac{1}{2}\sum_{t=1}^{n}\log\sigma_t^2-\frac{d+1}{2}\sum_{t=1}^{n}\log\left[1+\frac{x_t^2}{\sigma_t^2(d-2)}\right]$$

**(ii)**

```
library(fGarch)
```

```
## Warning:   'fGarch' R 4.1.2
```

```
##      timeDate
```

```
##
##    'timeDate'
```

```
## The following objects are masked from 'package:TSA':
##
##     kurtosis, skewness
```

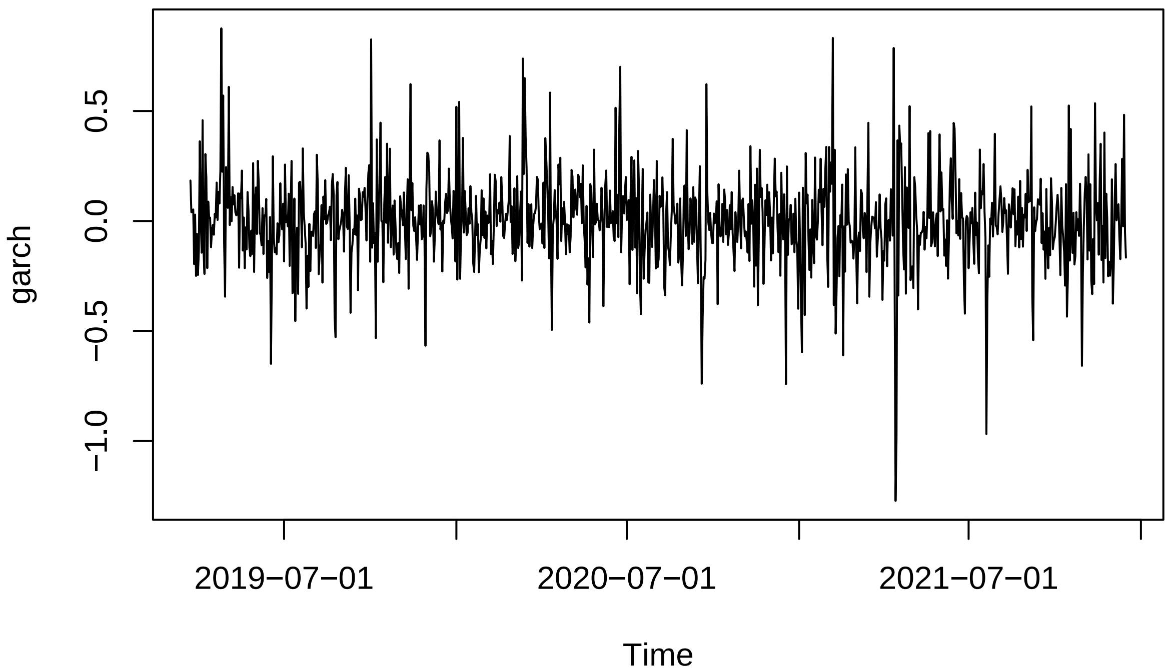
```
##      timeSeries
```

```
## Warning:   'timeSeries' R 4.1.2
```

```
##      fBasics
```

```
## Warning:   'fBasics' R 4.1.2
```

```
set.seed(1234)
spec <- garchSpec(model = list(omega = 0.02, alpha = 0.25, beta = 0.35, shape = 4),cond.dist = "std")
simulate <- garchSim(spec = spec, n = 1000, n.start = 100, extended = FALSE)

#spec default h = 0.05 (\sigma_1^2)
plot(simulate)
```




**(iii)**

```
fit <-
  garchFit( ~ garch(1, 1),
            data = simulate,
            cond.dist = "std")
```

```
##
## Series Initialization:
##  ARMA Model:                arma
##  Formula Mean:              ~ arma(0, 0)
##  GARCH Model:               garch
##  Formula Variance:          ~ garch(1, 1)
##  ARMA Order:                0 0
##  Max ARMA Order:            0
##  GARCH Order:               1 1
##  Max GARCH Order:           1
##  Maximum Order:             1
##  Conditional Dist:          std
##  h.start:                   2
##  llh.start:                 1
##  Length of Series:          1000
##  Recursion Init:            mci
##  Series Scale:              0.2063361
```

```
##
## Parameter Initialization:
##  Initial Parameters:          $params
##  Limits of Transformations:   $U, $V
##  Which Parameters are Fixed?  $includes
##  Parameter Matrix:
##                      U            V      params includes
##     mu     -0.08011218   0.08011218 0.008011218     TRUE
##     omega   0.00000100 100.00000000 0.100000000     TRUE
##     alpha1  0.00000001   0.99999999 0.100000000     TRUE
##     gamma1 -0.99999999   0.99999999 0.100000000    FALSE
##     beta1   0.00000001   0.99999999 0.800000000     TRUE
##     delta   0.00000000   2.00000000 2.000000000    FALSE
##     skew    0.10000000  10.00000000 1.000000000    FALSE
##     shape   1.00000000  10.00000000 4.000000000     TRUE
##  Index List of Parameters to be Optimized:
##     mu  omega alpha1  beta1  shape
##      1      2      3      5      8
##  Persistence:                  0.9
##
##
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
##
##   0:     1336.9901: 0.00801122 0.100000 0.100000 0.800000  4.00000
##   1:     1336.1840: 0.00801122 0.112859 0.103493 0.805075  4.00021
##   2:     1335.2491: 0.00801124 0.117080 0.0951693 0.794294  4.00008
##   3:     1334.3575: 0.00801126 0.143337 0.0924093 0.783512  4.00041
##   4:     1332.3526: 0.00801124 0.162380 0.111477 0.733240  4.00068
##   5:     1331.3055: 0.00801164 0.184981 0.148457 0.696216  4.00280
##   6:     1328.9920: 0.00802175 0.261279 0.137742 0.612170  4.00638
##   7:     1328.3712: 0.00803605 0.277124 0.218424 0.533361  4.01269
##   8:     1328.1652: 0.00803631 0.292452 0.218378 0.539802  4.01329
##   9:     1327.9212: 0.00808192 0.297626 0.210342 0.527533  4.01488
##  10:     1327.7415: 0.00816307 0.316919 0.205314 0.518037  4.03761
##  11:     1327.5870: 0.00846990 0.336608 0.212371 0.488963  4.07859
##  12:     1327.5202: 0.00889075 0.355461 0.213341 0.475037  3.95857
##  13:     1327.4884: 0.00970769 0.370065 0.213344 0.453735  4.04011
##  14:     1327.4849: 0.0101141 0.372006 0.215776 0.452157  4.01205
##  15:     1327.4814: 0.0109320 0.372265 0.215851 0.452880  3.99668
##  16:     1327.4766: 0.0127521 0.371895 0.216184 0.453454  3.99301
##  17:     1327.4755: 0.0135286 0.370885 0.215644 0.453968  4.00556
##  18:     1327.4754: 0.0135971 0.370480 0.215445 0.453991  4.01231
##  19:     1327.4754: 0.0135706 0.370411 0.215409 0.453995  4.01330
##  20:     1327.4754: 0.0135664 0.370406 0.215410 0.453993  4.01332
##
## Final Estimate of the Negative LLH:
##  LLH:  -250.7734    norm LLH:  -0.2507734
##          mu       omega      alpha1       beta1       shape
## 0.002799238 0.015769865 0.215410340 0.453993190 4.013323033
##
## R-optimhess Difference Approximated Hessian Matrix:
```

```
##                  mu       omega       alpha1        beta1        shape
## mu     -36636.258227   -1514.847    169.93347    -75.62777     5.307727
## omega   -1514.847180 -643842.570 -13508.83620 -25486.01344 -1900.015504
## alpha1    169.933470  -13508.836   -696.68578   -671.29857   -53.817248
## beta1     -75.627767  -25486.013   -671.29857  -1145.47683   -79.065247
## shape       5.307727   -1900.016    -53.81725    -79.06525    -9.486147
## attr(,"time")
## Time difference of 0.03690314 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.175498 secs
```

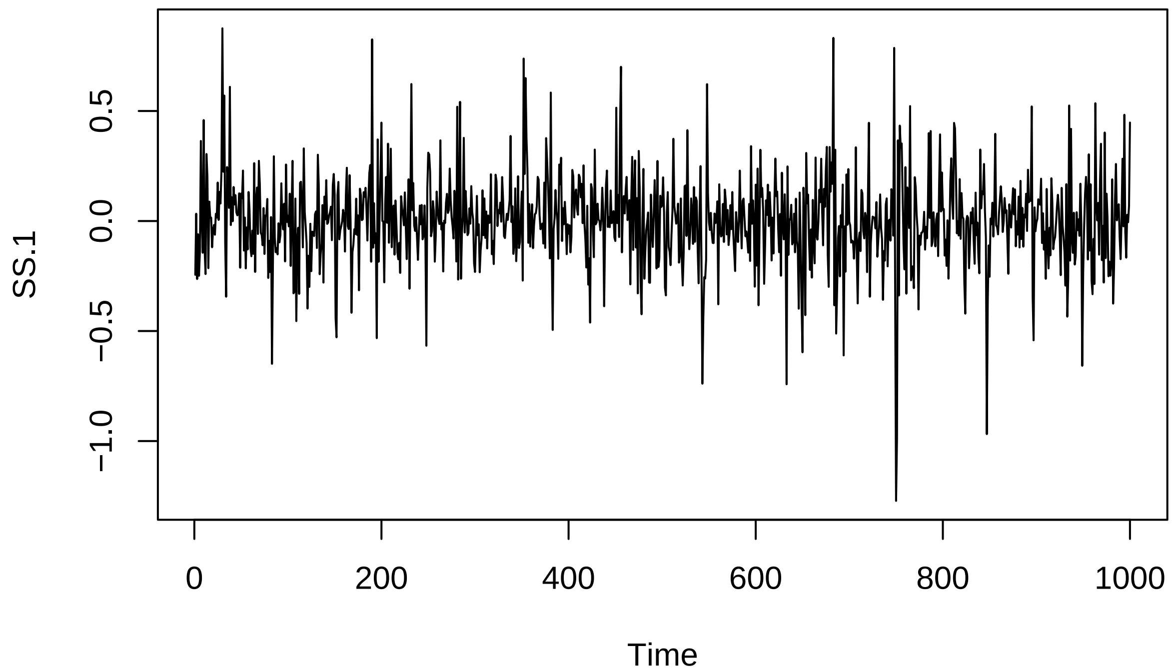
```
fit
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 1), data = simulate, cond.dist = "std")
##
## Mean and Variance Equation:
##  data ~ garch(1, 1)
## <environment: 0x000000001daf2be0>
##  [data = simulate]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##        mu      omega     alpha1      beta1      shape
## 0.0027992  0.0157699  0.2154103  0.4539932  4.0133230
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.002799    0.005237    0.534 0.593002
## omega   0.015770    0.004016    3.926 8.62e-05 ***
## alpha1  0.215410    0.063408    3.397 0.000681 ***
## beta1   0.453993    0.104740    4.334 1.46e-05 ***
## shape   4.013323    0.542193    7.402 1.34e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  250.7734    normalized:  0.2507734
##
## Description:
##  Thu Dec 16 18:46:00 2021 by user: HP
```

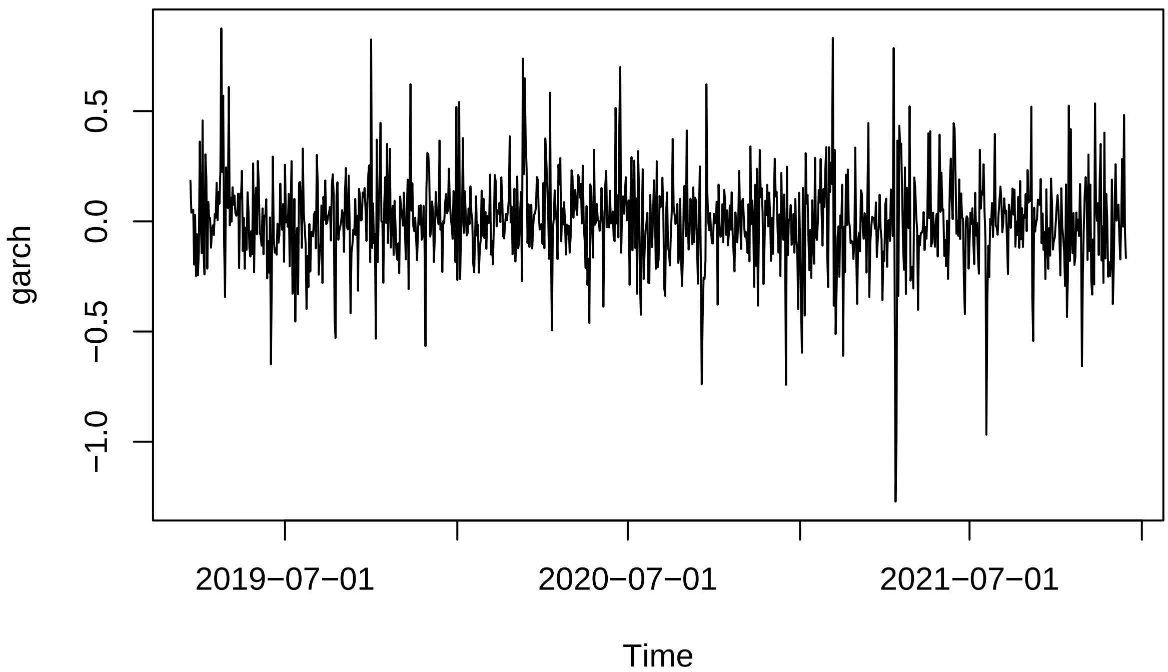
**Our algorithm**

**(ii)**

```
set.seed(1234)
omega <- 0.02
alpha <- 0.25
beta <- 0.35
d <- 4

epsilon <- sqrt((d-2)/d) * rt(1100,d)
epsilon <- epsilon[101:1100]

sigma.square <- rep(0,1001)
sigma.square[1] <- 0.05
x <- rep(0,1000)

for(t in 1:1000){
  x[t] <- sqrt(sigma.square[t]) * epsilon[t]
  sigma.square[t+1] <- omega + alpha * x[t]^2 + beta * sigma.square[t]
}



plot(as.timeSeries(x))
```




```
plot(simulate)
```




**(iii)**

Newton-Raphson algorithm:

```
##
sigma_wa <- 0

##
sigma_wb <- function(xlist, beta,t) {
  x <- xlist[1:t]
  sigma_wb <- 0
  for (i in 3:length(xlist)) {
    sigma_wb <- sigma_wb + (i - 2) * beta ^ (i - 3)
  }
  return(sigma_wb)
}

##
sigma_ab <- function(xlist, beta,t) {
  x <- xlist[1:t]
  sigma_ab <- 0
  for (i in 3:length(x)) {
    sigma_ab <- sigma_ab + (i - 2) * (x[length(x) - i + 1] ^ 2) * (beta ^ (i - 3))
  }
  if(t==2){
```

```
    return(0)
  }
  return(sigma_ab)
}

##
sigma_ww <- 0

##
sigma_aa <- 0

##
sigma_bb <- function(xlist, beta, w, alpha, sigma1, t) {
  x <- xlist[1:t]
  p1 <- p2 <- 0
  for (i in 4:length(x)) {
    p1 <- p1 + w * ((i - 2) * (i - 3) * (beta ^ (i - 4)))
    p2 <-
      p2 + alpha * (x[length(x) - i + 1] ^ 2 * (i - 2) * (i - 3) * beta ^ (i - 4))
  }
  sigma_bb <-
    p1 + p2 + (length(x) - 1) * sigma1 * (length(x) - 2) * beta ^ (length(x) - 3)
  if (t == 2) {
    return(0)
  }
  if (t == 3) {
    return(2 * sigma1)
  }
  return(sigma_bb)
}

##
dldsigma2 <- function(w, alpha, beta, sigma1, xlist, d, t) {
  xlist <- xlist[1:t]
  sigma2 <- c(sigma1)
  for (i in 2:length(xlist)) {
    sigma2 <-
      c(sigma2, w + alpha * xlist[i - 1] ^ 2 + beta * sigma2[i - 1])
  }
  sigma2t <- sigma2[t]
  dldsigma2 <-
    -1 / 2 / sigma2t + (d + 1) / 2 * (xlist[t] ^ 2 / (sigma2[t] * (d - 2) +
                                                        xlist[t] ^ 2) * 1 / sigma2[t])
  return(dldsigma2)
}

##
dl2d2sigma2 <- function(w, alpha, beta, sigma1, xlist, d, t) {
  xlist <- xlist[1:t]
  sigma2 <- c(sigma1)
  for (i in 2:length(xlist)) {
    sigma2 <-
      c(sigma2, w + alpha * xlist[i - 1] ^ 2 + beta * sigma2[i - 1])
```

```
  }
  sigma2t <- sigma2[t]
  dl2d2sigma2 <- 1 / 2 * (1 / (sigma2[t]) ^ 2) -
    (d + 1) / 2 * ((d - 2) * xlist[t] ^ 2 / (sigma2[t] * (d - 2) +
    xlist[t] ^ 2) ^ 2 * 1 / sigma2[t] + xlist[t] ^ 2 / ((sigma2[t] * (d - 2) +
    xlist[t] ^ 2) * sigma2[t] ^ 2))

  return(dl2d2sigma2)
}

##
dsigma2dw <- function(beta, sigma1, xlist, t) {
  xlist <- xlist[1:t]
  dsigma2dw <- (1 - beta ^ (length(xlist) - 1)) / (1 - beta)
  return(dsigma2dw)
}

##
dsigma2da <- function(beta, sigma1, xlist, t) {
  xlist <- xlist[1:t]
  dsigma2da <- 0
  for (i in 2:length(xlist) - 1) {
    dsigma2da <-
      dsigma2da + beta ^ (i - 1) * (xlist[length(xlist) - i]) ^ 2
  }
  if (t == 1) {
    return(0)
  }
  return(dsigma2da)
}

##
dsigma2db <- function(w, alpha, beta, sigma1, xlist, t) {
  xlist <- xlist[1:t]
  dsigma2db <- 0
  for (i in 1:(t - 2)) {
    dsigma2db <-
      dsigma2db + w * i * (beta) ^ (i - 1) +
      alpha * (xlist[i]) ^2 * (t - i - 1) * beta ^ (t - i - 2)
  }
  dsigma2db <- dsigma2db + (t - 1) * beta ^ (t - 2) * sigma1
  if (t == 1) {
    return(0)
  }
  if (t == 2) {
    return(sigma1)
  }
  return(dsigma2db)
}


##
Information <- function(w, alpha, beta, sigma1, xlist, d) {
```

```r
a11 <- a12 <- a13 <- a22 <- a23 <- a33 <- 0

for (t in 2:length(xlist)) {
  ## a11
  a11 <-
    a11 + sigma_ww * dldsigma2(w, alpha, beta, sigma1, xlist, d, t) +
    (dsigma2dw(beta, sigma1, xlist, t)) ^ 2 *
    dl2d2sigma2(w, alpha, beta, sigma1, xlist, d, t)

  ## a12
  a12 <-
    a12 + sigma_wa * dldsigma2(w, alpha, beta, sigma1, xlist, d, t) +
    dsigma2dw(beta, sigma1, xlist, t) *
    dl2d2sigma2(w, alpha, beta, sigma1, xlist, d, t) *
    dsigma2da(beta, sigma1, xlist, t)

  ## a13
  a13 <-
    a13 + sigma_wb(xlist, beta, t) *
    dldsigma2(w, alpha, beta, sigma1, xlist, d, t) +
    dsigma2dw(beta, sigma1, xlist, t) *
    dl2d2sigma2(w, alpha, beta, sigma1, xlist, d, t) *
    dsigma2db(w, alpha, beta, sigma1, xlist, t)

  # a22
  a22 <-
    a22 + sigma_aa * dldsigma2(w, alpha, beta, sigma1, xlist, d, t) +
    dsigma2da(beta, sigma1, xlist, t) *
    dl2d2sigma2(w, alpha, beta, sigma1, xlist, d, t) *
    dsigma2da(beta, sigma1, xlist, t)

  ## a23
  a23 <-
    a23 + sigma_ab(x, beta, t) * dldsigma2(w, alpha, beta, sigma1, xlist, d, t) +
    dsigma2da(beta, sigma1, xlist, t) *
    dl2d2sigma2(w, alpha, beta, sigma1, xlist, d, t) *
    dsigma2db(w, alpha, beta, sigma1, xlist, t)

  ## a33
  a33 <-
    a33 + sigma_bb(xlist, beta, w, alpha, sigma1, t) *
    dldsigma2(w, alpha, beta, sigma1, xlist, d, t) +
    dsigma2db(w, alpha, beta, sigma1, xlist, t) *
    dl2d2sigma2(w, alpha, beta, sigma1, xlist, d, t) *
    dsigma2db(w, alpha, beta, sigma1, xlist, t)
}

a21 <- a12
a31 <- a13
a32 <- a23

M <- matrix(
  c(-a11, -a12, -a13, -a21, -a22, -a23, -a31, -a32, -a33),
```

```r
    nrow = 3,
    byrow = TRUE
  )
  return(M)
}


##
dl <- function(w, alpha, beta, sigma1, xlist, d) {
  b1 <- b2 <- b3 <- 0

  for (t in 2:length(xlist)) {
    b1 <-
      b1 + dsigma2dw(beta, sigma1, xlist, t) *
      dldsigma2(w, alpha, beta, sigma1, xlist, d, t)
    b2 <-
      b2 + dsigma2da(beta, sigma1, xlist, t) *
      dldsigma2(w, alpha, beta, sigma1, xlist, d, t)
    b3 <-
      b3 + dsigma2db(w, alpha, beta, sigma1, xlist, t) *
      dldsigma2(w, alpha, beta, sigma1, xlist, d, t)
  }

  M <- matrix(c(b1, b2, b3), nrow = 3, byrow = TRUE)
  return(M)
}

##
library(MASS)
literation_NR <- function(w, alpha, beta, sigma1, xlist, d) {
  theta <- matrix(c(w, alpha, beta), nrow = 3, byrow = TRUE)
  theta1 <-
    theta + ginv(Information(w, alpha, beta, sigma1, xlist, d)) %*%
    dl(w, alpha, beta, sigma1, xlist, d)
  return(theta1)
}

##
GetEst <- function(w, alpha, beta, sigma1, xlist, d) {
  thetaT <- matrix(c(w, alpha, beta), nrow = 3, byrow = TRUE)
  while (TRUE) {
    thetai <-
      literation_NR(thetaT[1], thetaT[2], thetaT[3], sigma1, xlist, d)
    if (abs((thetai - thetaT)[1]) <= 0.000001 &&
        abs((thetai - thetaT)[2]) <= 0.000001 &&
        abs((thetai - thetaT)[3]) <= 0.000001) {
      thetaT <- thetai
      break
    }
    thetaT <- thetai
  }
  return(thetaT)
}
```

Estimate:

```r
GetEst(0.02, 0.3, 0.3, 0.05, x, 4)
```

```
##           [,1]
## [1,] 0.0158711
## [2,] 0.2109344
## [3,] 0.4568039
```

```r
GetEst(0.015, 0.3, 0.4, 0.05, x, 4)
```

```
##           [,1]
## [1,] 0.0158711
## [2,] 0.2109344
## [3,] 0.4568039
```

```r
GetEst(0.02, 0.2, 0.3, 0.05, x, 4)
```

```
##           [,1]
## [1,] 0.0158711
## [2,] 0.2109344
## [3,] 0.4568039
```

```r
GetEst(0.02, 0.25, 0.35, 0.05, x, 4)
```

```
##           [,1]
## [1,] 0.0158711
## [2,] 0.2109344
## [3,] 0.4568039
```

Standard error:

```r
I <- Information(0.0158711, 0.2109344, 0.4568039, 0.05, x, 4)
SE <- sqrt(diag(ginv(I)))
SE
```

```
## [1] 0.003869506 0.059891588 0.106483172
```