GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

05/02/2021

Input

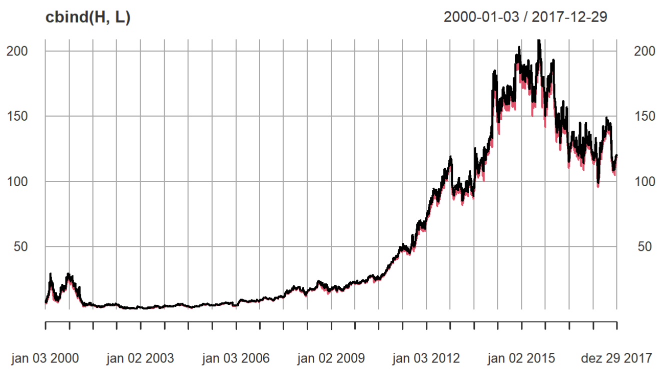
```
symbol = "ALXN"
from=as.Date('2000-01-01')
to=as.Date('2017-12-31')
C_Trend = 0.95
C_Reaction = 0.50
```

Data download

```
x <- getSymbols.yahoo(symbol,auto.assign = FALSE, from=from, to=to)
```

High and Low

```
H <- Hi(x)
L <- Lo(x)
C <- Cl(x)
plot(cbind(H,L))
```

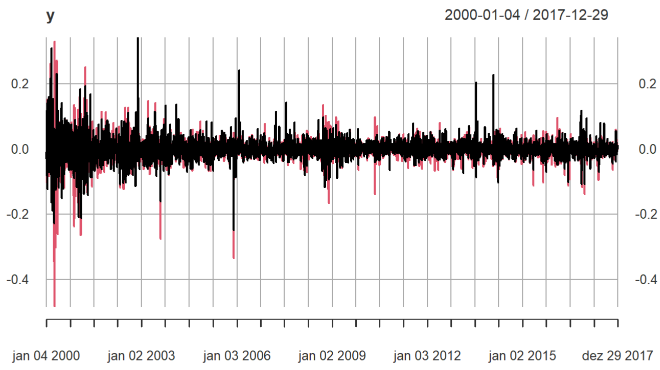


Returns

```
y <- cbind( diff(log(H)),  diff(log(L)) )
y <- na.omit(y)
y %>% cor() # Returns correlation
```

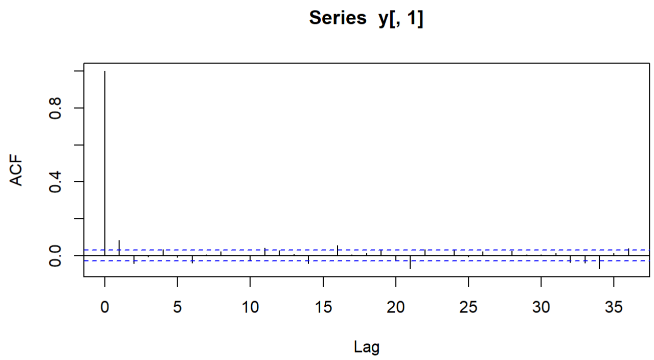
```
##           ALXN.High  ALXN.Low
## ALXN.High 1.0000000 0.5575223
## ALXN.Low  0.5575223 1.0000000
```

```
plot(y)
```

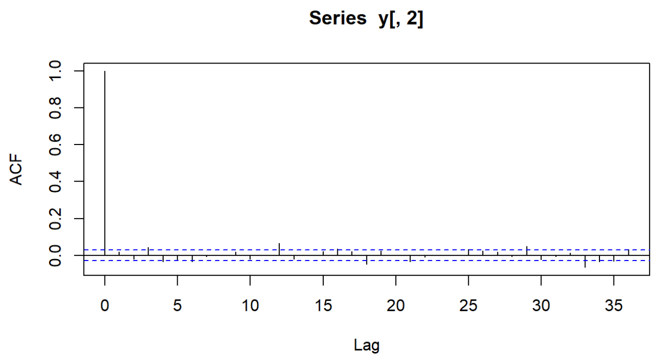


Autocorrelation

```
acf(y[,1])
```

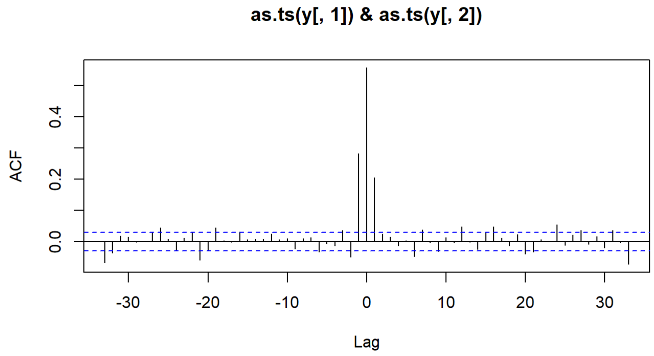



```
acf(y[,2])
```



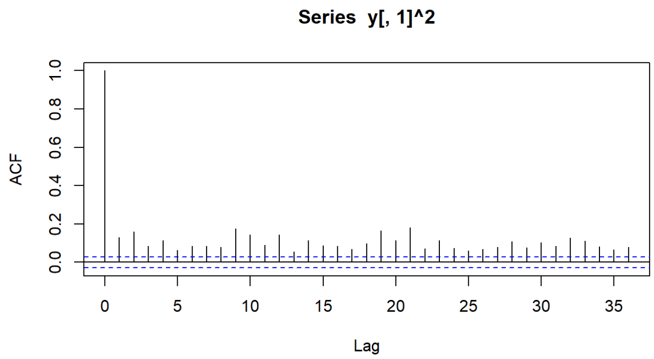
Cross correlation

```
ccf(as.ts(y[,1]),as.ts(y[,2]))
```

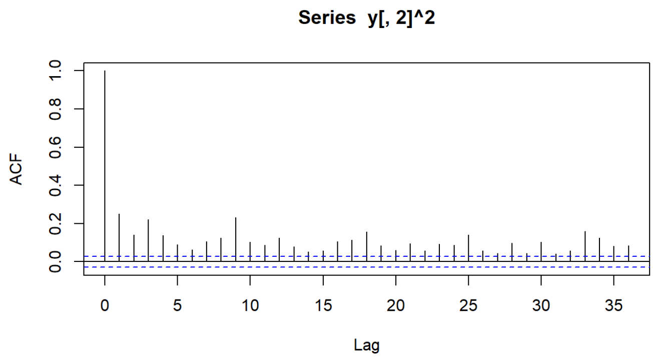


Volatility verification

```
acf(y[,1]^2)
```




```
acf(y[,2]^2)
```



Bivariate DCC-GARCH

We will consider the DCC-GARCH to model the volatility of $y = (r_H, r_L)'$, where r_H and r_L denote the $100\times$log-returns from hight's and low's observations.

```
# returns
mY <- 100*y

# generates the Markov Chain
start <- Sys.time()

out <- bayesDccGarch(mY, control=list(print=FALSE, nPilotSim=3000))
```

```
## Maximizing the log-posterior density function.
## Done.
## One approximation for covariance matrix of parameters cannot be directly computed through the hessian matrix.
## Calibrating the standard deviations for simulation:
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.29   0.09   0.21   0.06   0.10   0.09   0.19   0.09   0.10   0.34   0.63
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.27   0.16   0.22   0.12   0.16   0.17   0.20   0.12   0.17   0.36   0.52
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.28   0.15   0.21   0.17   0.15   0.16   0.19   0.22   0.19   0.37   0.42
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.27   0.16   0.20   0.15   0.25   0.17   0.21   0.19   0.18   0.38   0.44
## Computing the covariance matrix of pilot sample.
```

```
## Warning in if (class(control$cholCov) != "try-error") {: a condição tem
## comprimento > 1 e somente o primeiro elemento será usado
```

```
## Done.
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.44
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

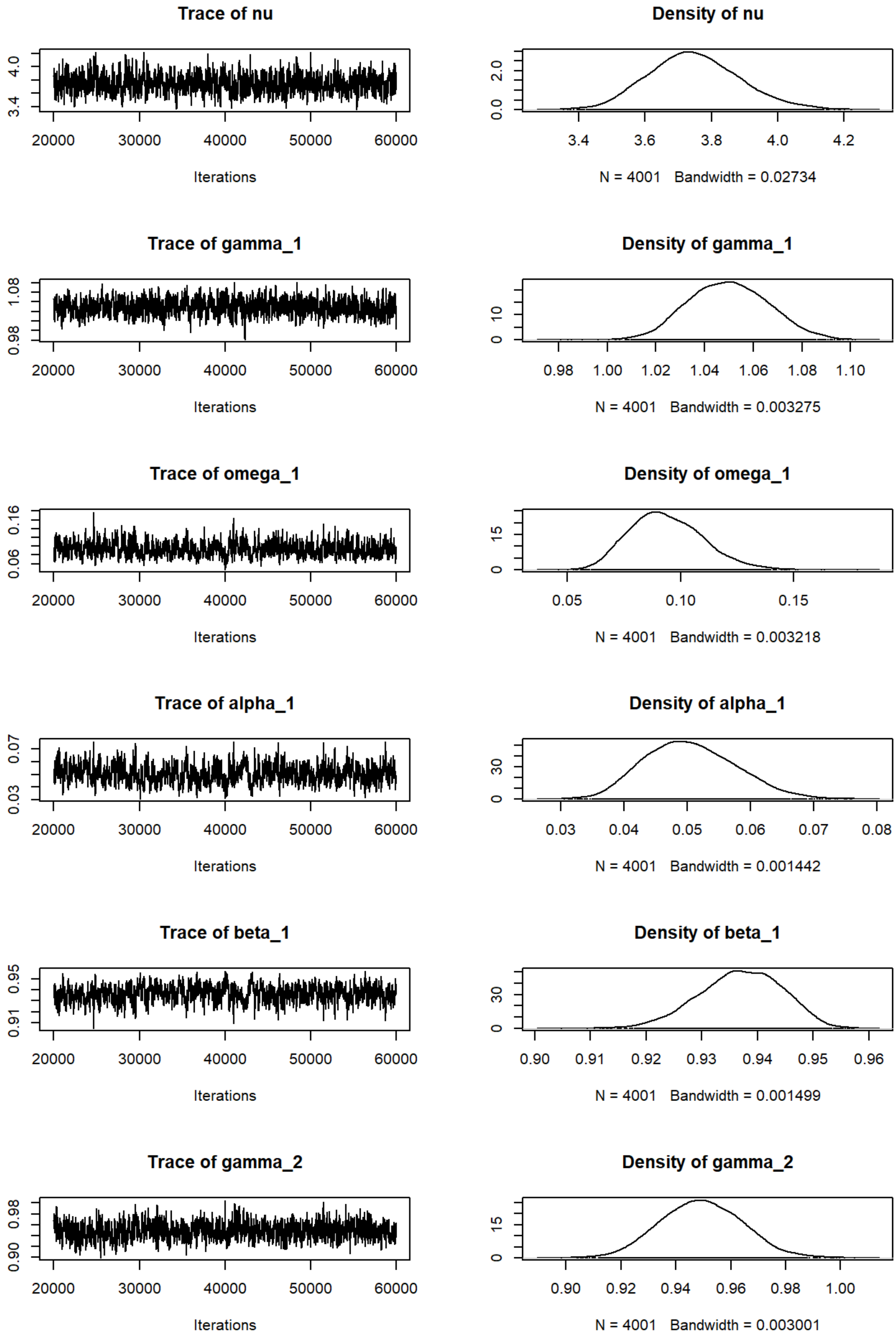
```
out2 <- increaseSim(out, nSim=50000)
```

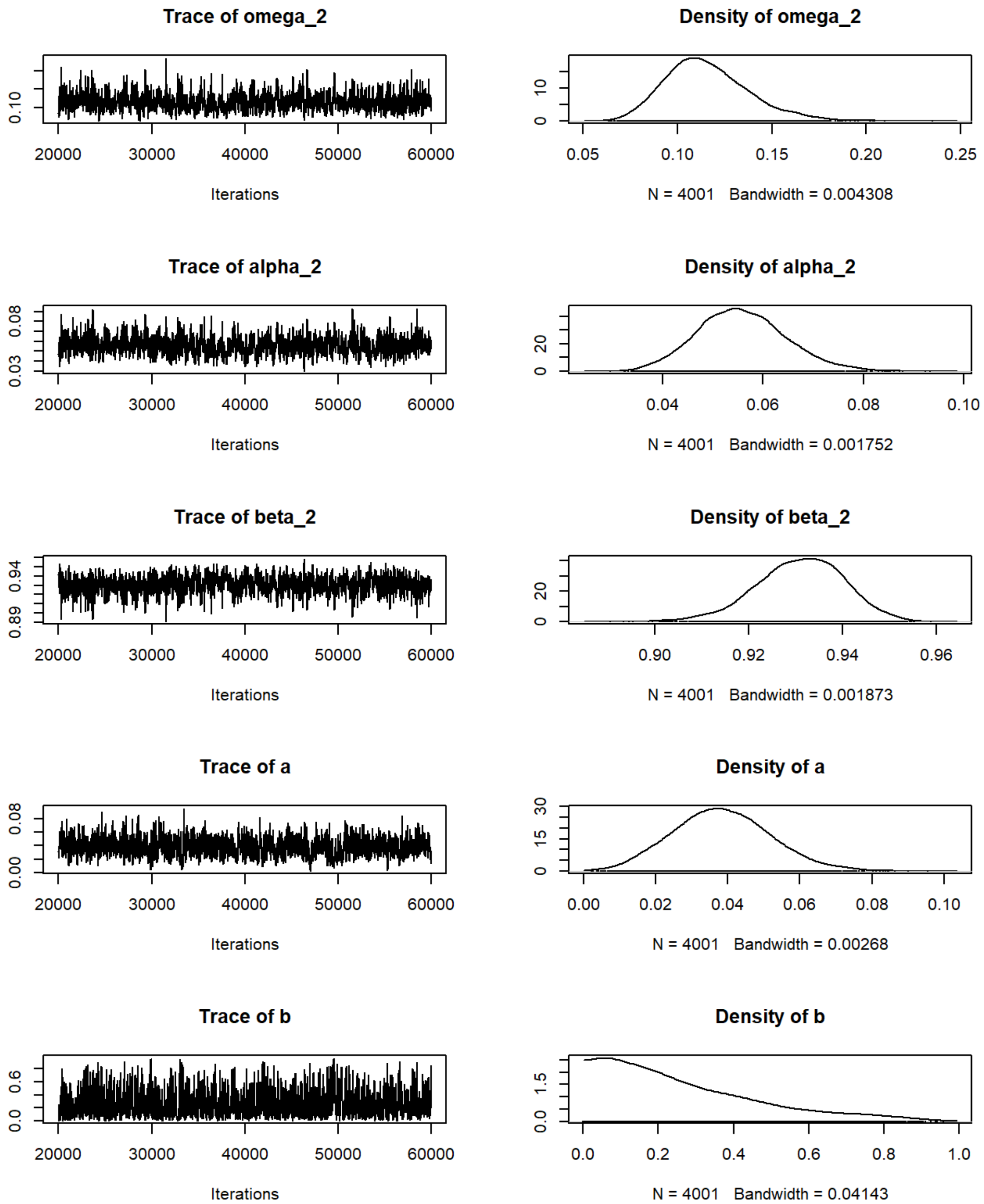
```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.43
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

```
out <- window(out2, start=20000, thin=10)
rm(out2)

end <- Sys.time()

# elapsed time
end-start
```

```
## Time difference of 16.99646 mins
```

```
## Estimative of parameters
parEst <- summary(out)$statistics[,'Mean']

# plot Markov Chain
plot(out$MC)
```

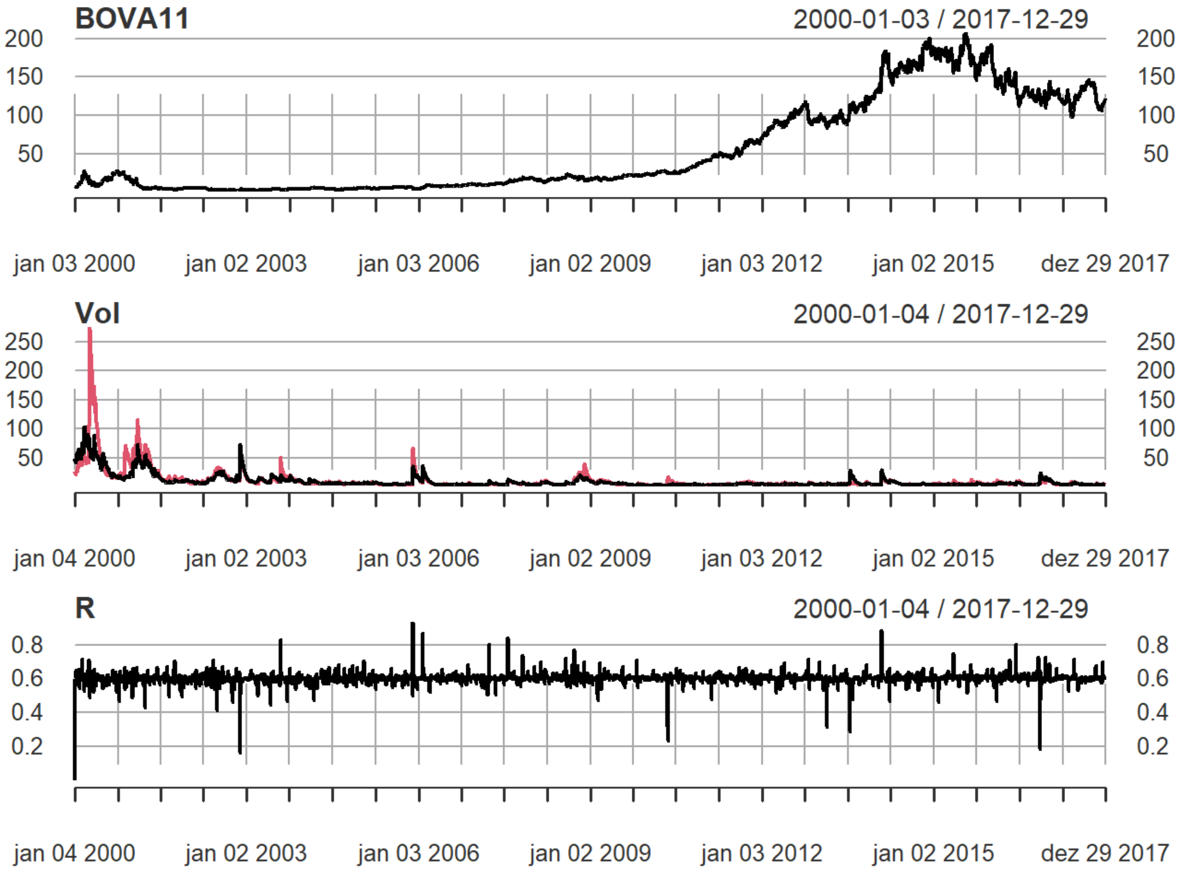




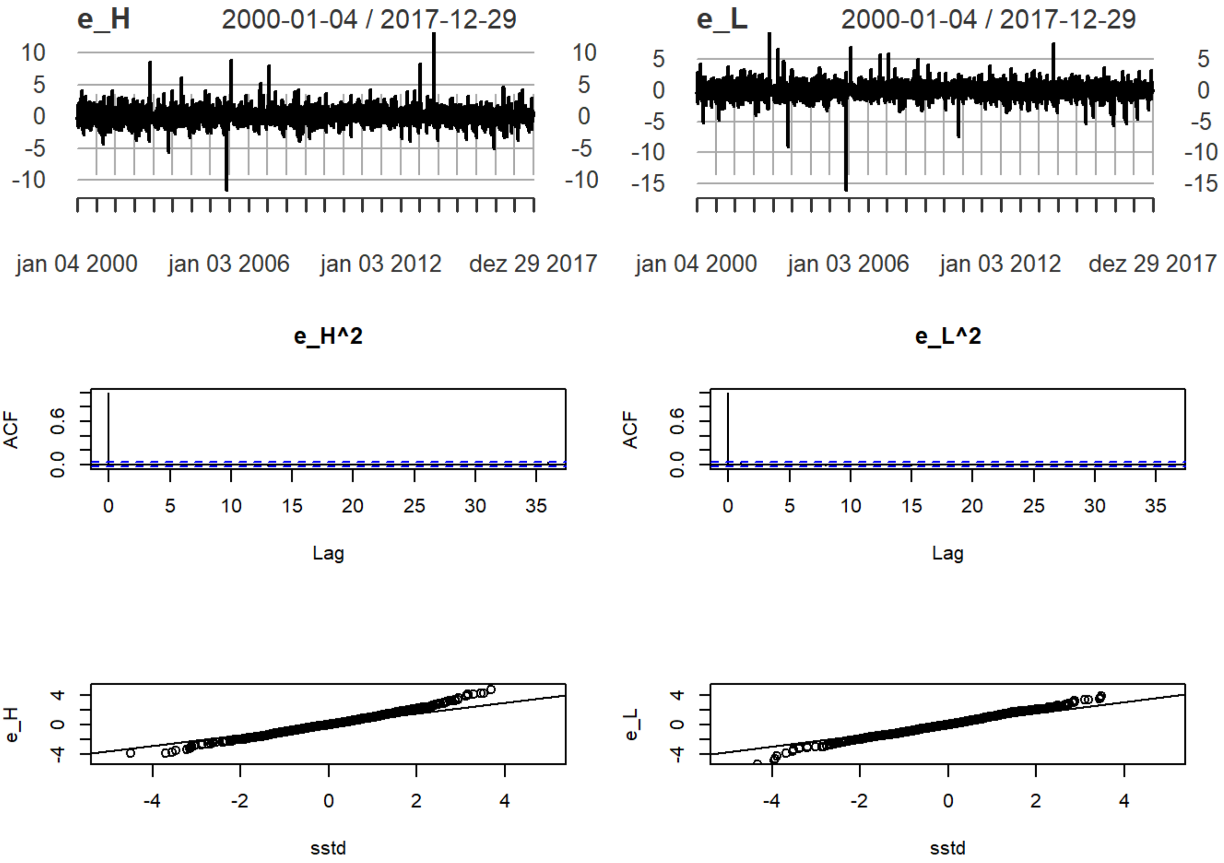

```
## Estimative of parameters
out$MC %>% summary()
```

```
##
## Iterations = 20000:60000
## Thinning interval = 10
## Number of chains = 1
## Sample size per chain = 4001
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##             Mean       SD  Naive SE Time-series SE
## nu      3.74802 0.135489 0.0021420      0.0056241
## gamma_1 1.04973 0.016230 0.0002566      0.0006307
## omega_1 0.09364 0.015948 0.0002521      0.0007207
## alpha_1 0.05052 0.007149 0.0001130      0.0003651
## beta_1  0.93688 0.007431 0.0001175      0.0003763
## gamma_2 0.94930 0.014871 0.0002351      0.0005685
## omega_2 0.11556 0.021933 0.0003467      0.0008812
## alpha_2 0.05581 0.008785 0.0001389      0.0003408
## beta_2  0.93093 0.009281 0.0001467      0.0003312
## a       0.03835 0.013504 0.0002135      0.0005064
## b       0.26023 0.205320 0.0032460      0.0077479
##
## 2. Quantiles for each variable:
##
##             2.5%     25%     50%     75%   97.5%
## nu      3.50204 3.65439 3.74165 3.83611 4.03085
## gamma_1 1.01950 1.03826 1.04962 1.06107 1.08170
## omega_1 0.06656 0.08216 0.09228 0.10405 0.12772
## alpha_1 0.03800 0.04540 0.05006 0.05528 0.06550
## beta_1  0.92166 0.93192 0.93713 0.94224 0.95008
## gamma_2 0.92116 0.93902 0.94908 0.95948 0.97880
## omega_2 0.07866 0.10018 0.11300 0.12878 0.16434
## alpha_2 0.03961 0.04967 0.05531 0.06130 0.07440
## beta_2  0.91104 0.92499 0.93142 0.93752 0.94793
## a       0.01345 0.02924 0.03795 0.04704 0.06655
## b       0.01352 0.09591 0.20843 0.37823 0.77545
```

```
## Conditional Correlation
R <-  xts(out$R[,2], order.by=index(y))



## Volatility
Vol <- xts(out$H[,c("H_1,1","H_2,2")], order.by=index(y))

par(mfrow=c(3,1))
plot(C, main="BOVA11")
plot(Vol)
plot(R, main="R")
```




```
## Standard Residuals
r <- mY / sqrt(Vol)

par(mfrow=c(3,2))

plot(r[,1], main="e_H")
plot(r[,2], main="e_L")
acf(r[,1]^2, main="e_H^2")
acf(r[,2]^2, main="e_L^2")
r1 <- as.numeric(r[,1])
x <- rsstd(2000, mean = 0, sd = 1, nu = parEst['nu'], xi =parEst['gamma_1'])
qqplot(x=x, y=r1, xlim=c(-5, 5), ylim=c(-5, 5), ylab="e_H",xlab="sstd")
qqline(r1)
r2 <- as.numeric(r[,2])
x <- rsstd(2000, mean = 0, sd = 1, nu = parEst['nu'], xi =parEst['gamma_2'])
qqplot(x=x, y=r2 , xlim=c(-5, 5), ylim=c(-5, 5), ylab="e_L",xlab="sstd" )
qqline(r2)
```


e_H 2000-01-04 / 2017-12-29
10
5
0
-5
-10
jan 04 2000
jan 03 2006
jan 03 2012
dez 29 2017
e_L 2000-01-04 / 2017-12-29
5
0
-5
-10
-15
e_H^2
ACF
0.0
0.6
Lag
e_L^2
e_H
e_L
sstd

```
# Prepare input for the expert advisor

## High
#HBOP
High_UB_HBOP = qsstd(p=1-(1-C_Trend)/2, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['gam
ma_1'])
#S1
High_UB_S1 = qsstd(p=1-(1-C_Reaction)/2, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['ga
mma_1'])


## Low
#B1
Low_LB_B1 = qsstd(p=(1-C_Reaction)/2, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['gamma
_2'])
#LBOP
Low_LB_LBOP = qsstd(p=(1-C_Trend)/2, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['gamma_
2'])

pH <- c(0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.975, 0.99, 0.995)
qH <- round(qsstd(p=pH, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['gamma_1']),3)
names(qH) <- paste0(100*pH,"%")
pL <- 1 - pH
qL <- round(qsstd(p=pL, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['gamma_2']),3)
names(qL) <- paste0(100*pL,"%")

qC <- rbind(qH, qL)
rownames(qC) <- c("High_UB", "Low_LB")
colnames(qC) <- paste0(100*pL,"%")

m = matrix(NA,nrow=10,ncol=1)
rownames(m) = c("High_UB_HBOP","High_UB_S1","Low_LB_B1","Low_LB_LBOP",
               "High_omega", "High_alpha","High_beta",
                     "Low_omega",  "Low_alpha", "Low_beta" )
colnames(m) = 'Value'

m["High_UB_HBOP",1] = High_UB_HBOP
m["High_UB_S1",1] = High_UB_S1
m["Low_LB_B1",1] =  Low_LB_B1
m["Low_LB_LBOP",1] = Low_LB_LBOP

m["High_omega",1] = parEst["omega_1"]
m["High_alpha",1] = parEst["alpha_1"]
m["High_beta",1] = parEst["beta_1"]

m["Low_omega",1] = parEst["omega_2"]
m["Low_alpha",1] = parEst["alpha_2"]
m["Low_beta",1] = parEst["beta_2"]

# Input for expert advisor
print(qC)
```

```
          40%    35%    30%    25%    20%    15%    10%     5%   2.5%     1%
High_UB  0.165  0.265  0.374  0.497  0.641  0.821  1.073  1.519  2.007  2.757
Low_LB  -0.163 -0.264 -0.373 -0.496 -0.640 -0.821 -1.074 -1.521 -2.011 -2.764
          0.5%
High_UB  3.431
Low_LB  -3.441
```

```
print(round(m,3))
```

```
              Value
High_UB_HBOP  2.007
High_UB_S1    0.497
Low_LB_B1    -0.496
Low_LB_LBOP  -2.011
High_omega    0.094
High_alpha    0.051
High_beta     0.937
Low_omega     0.116
Low_alpha     0.056
Low_beta      0.931
```