GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

05/02/2021

Input

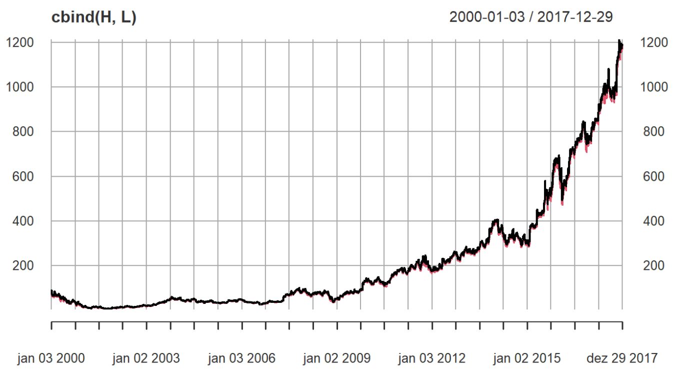
```
symbol = "AMZN"
from=as.Date('2000-01-01')
to=as.Date('2017-12-31')
C_Trend = 0.95
C_Reaction = 0.50
```

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))
```

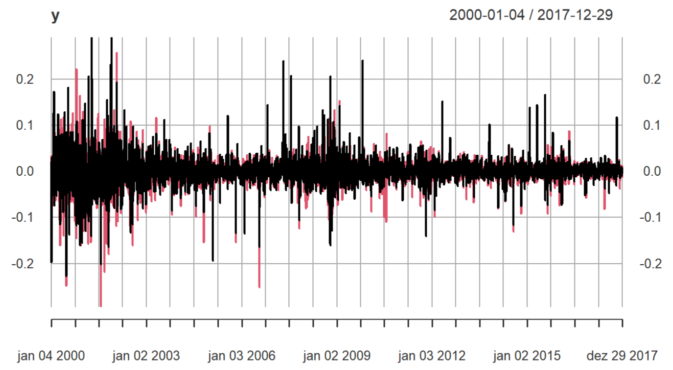


Returns

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

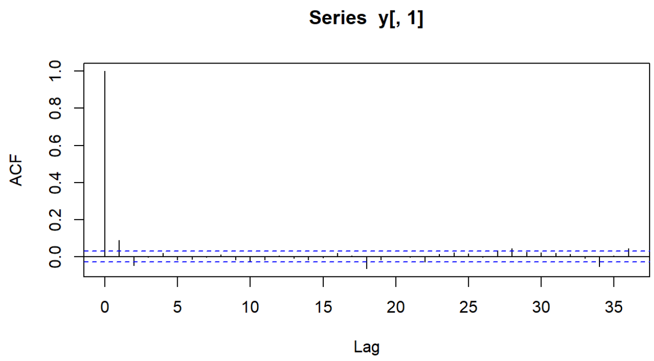
```
##           AMZN.High AMZN.Low
## AMZN.High  1.000000 0.732942
## AMZN.Low   0.732942 1.000000
```

```
plot(y)
```

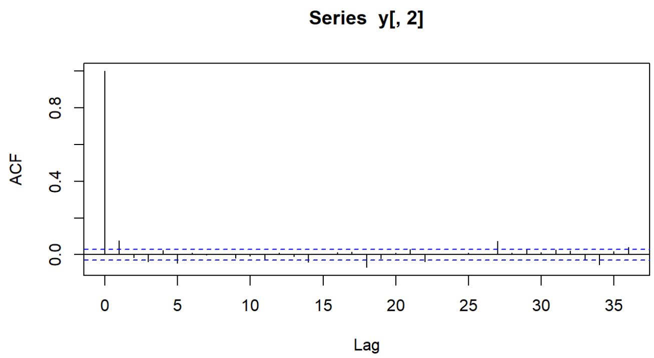


Autocorrelation

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

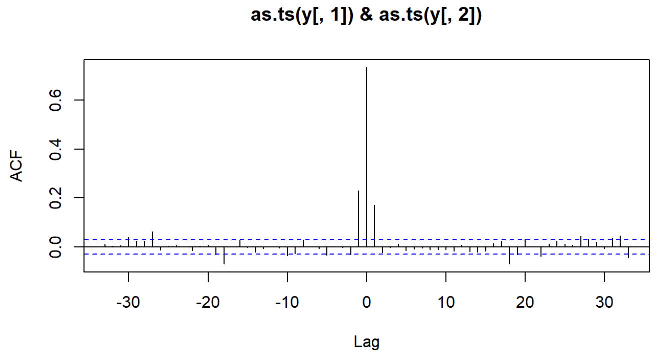



```
acf(y[,2])
```



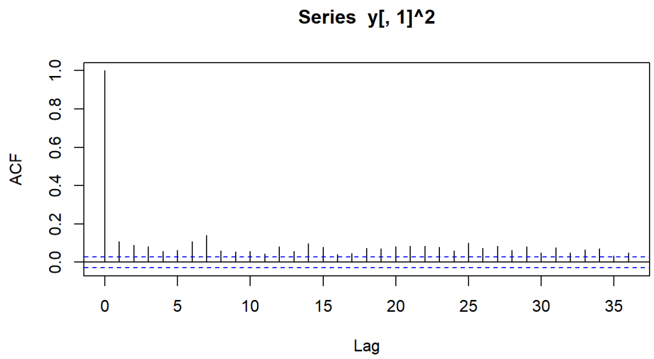
Cross correlation

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

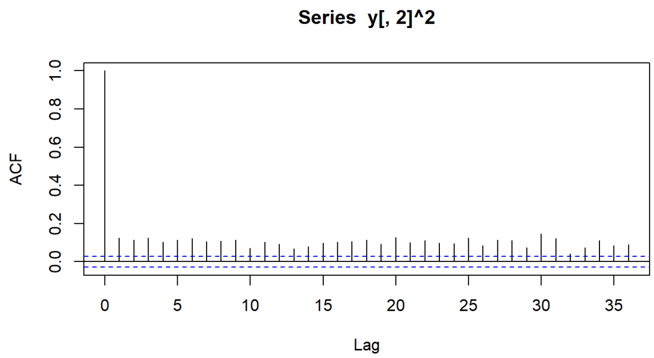


Volatility verification

```
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
##   0.26   0.09   0.21   0.05   0.08   0.09   0.18   0.05   0.05   0.19   0.17
## 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   0.17   0.22   0.08   0.10   0.16   0.19   0.05   0.07   0.20   0.17
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.26   0.16   0.20   0.11   0.16   0.16   0.20   0.09   0.14   0.21   0.17
## 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   0.16   0.21   0.17   0.16   0.18   0.20   0.13   0.20   0.21   0.18
## 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   0.21   0.17   0.14   0.20   0.23   0.23   0.20   0.18
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.25   0.17   0.21   0.20   0.17   0.26   0.22   0.22   0.21   0.18   0.16
## 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: 0.26
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

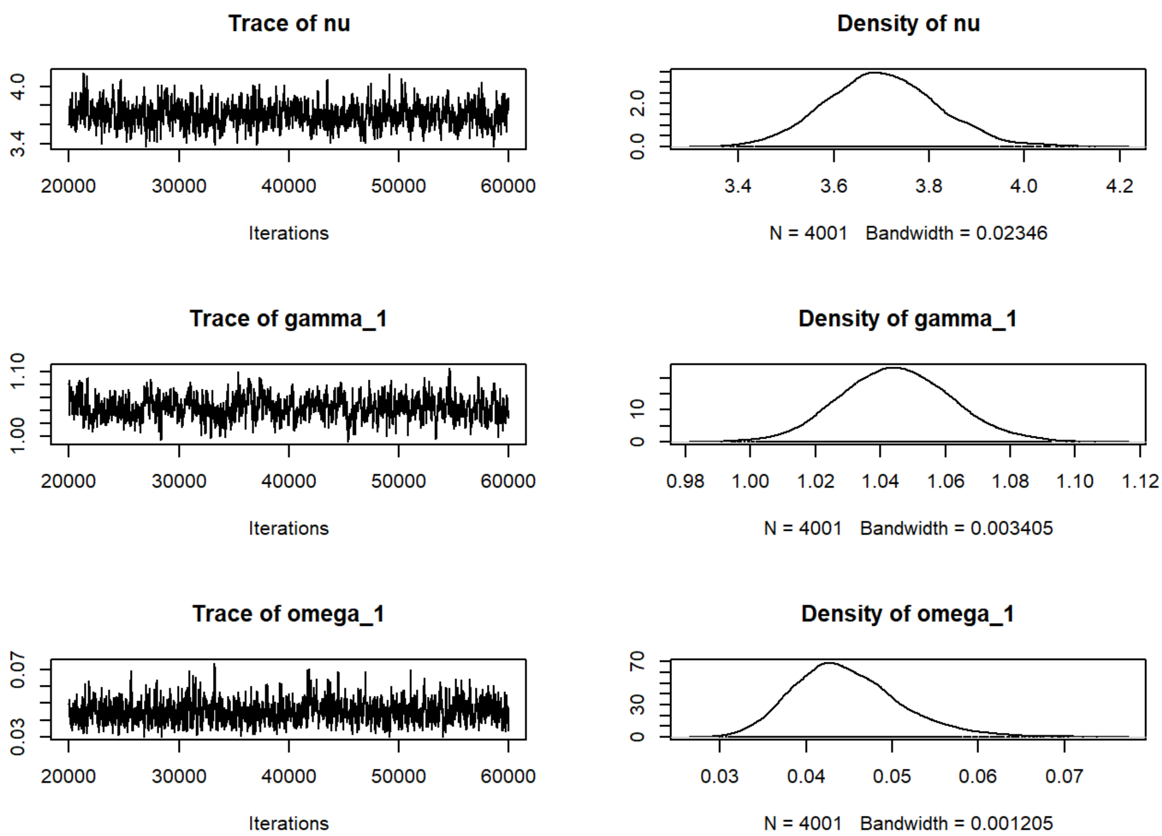
```
out2 <- increaseSim(out, nSim=50000)
```

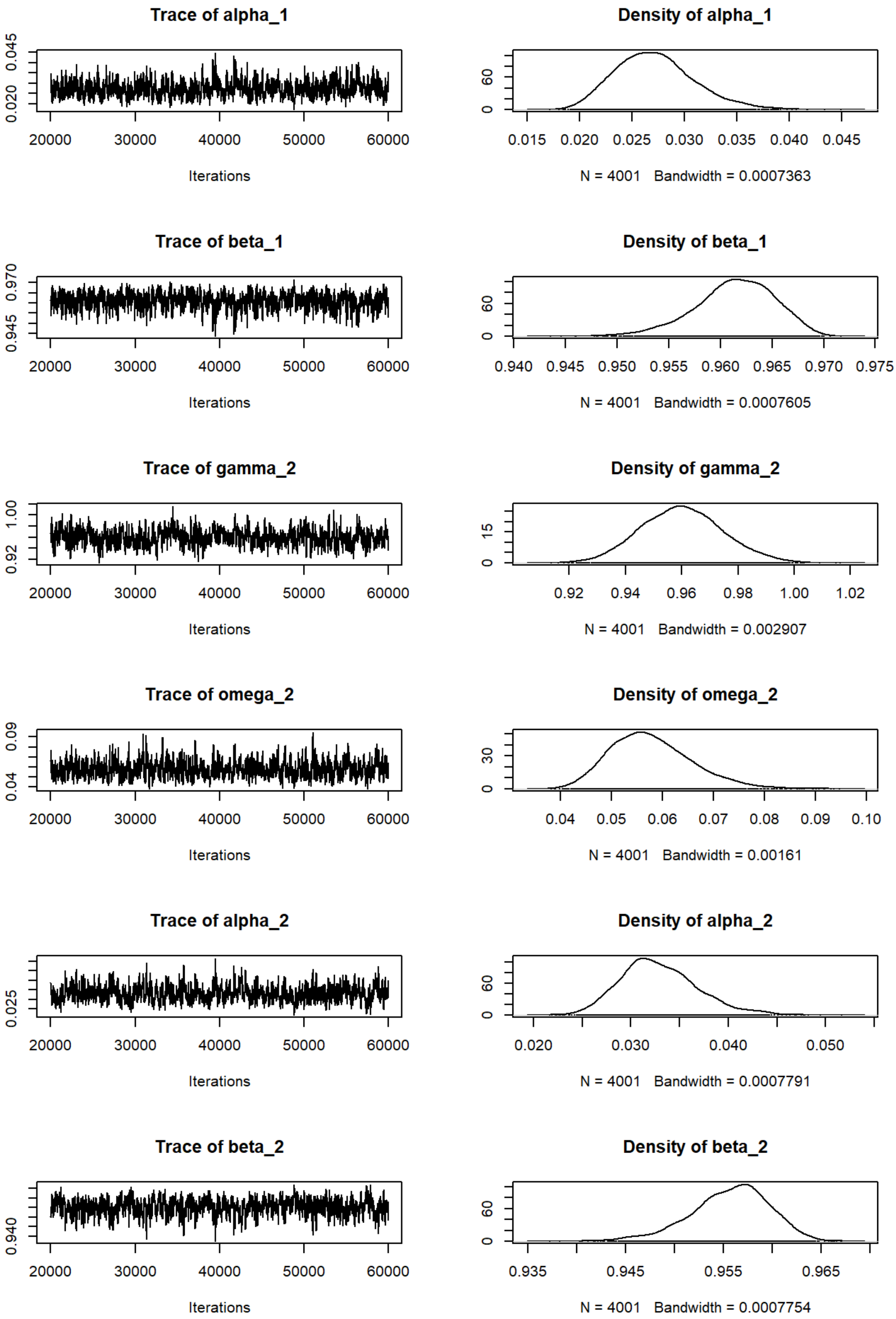
```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.26
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

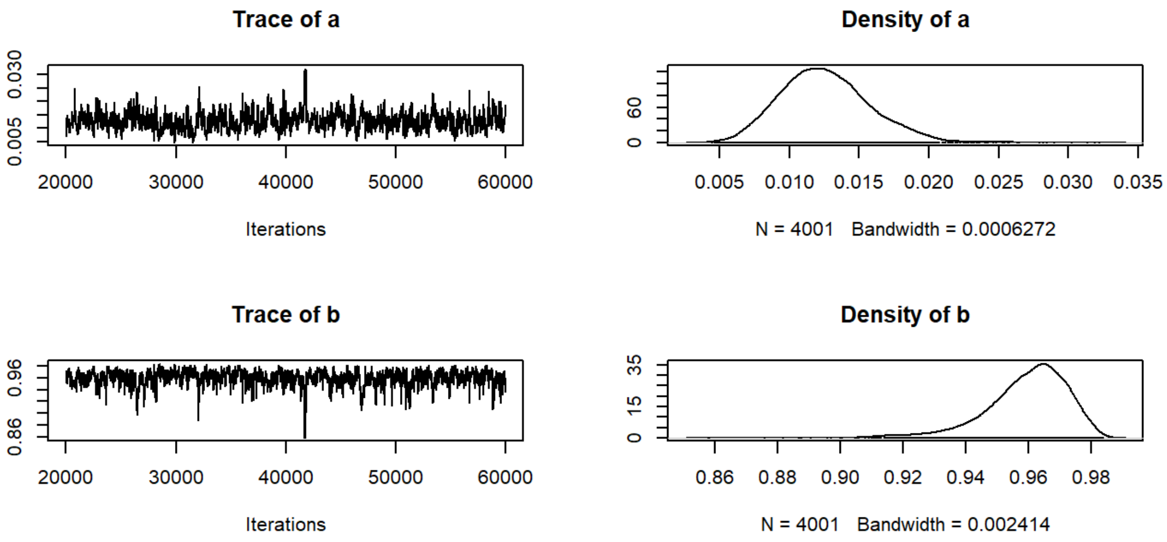
```
out <- window(out2, start=20000, thin=10)
rm(out2)

end <- Sys.time()

# elapsed time
end-start
```

```
## Time difference of 21.81299 mins
```

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

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



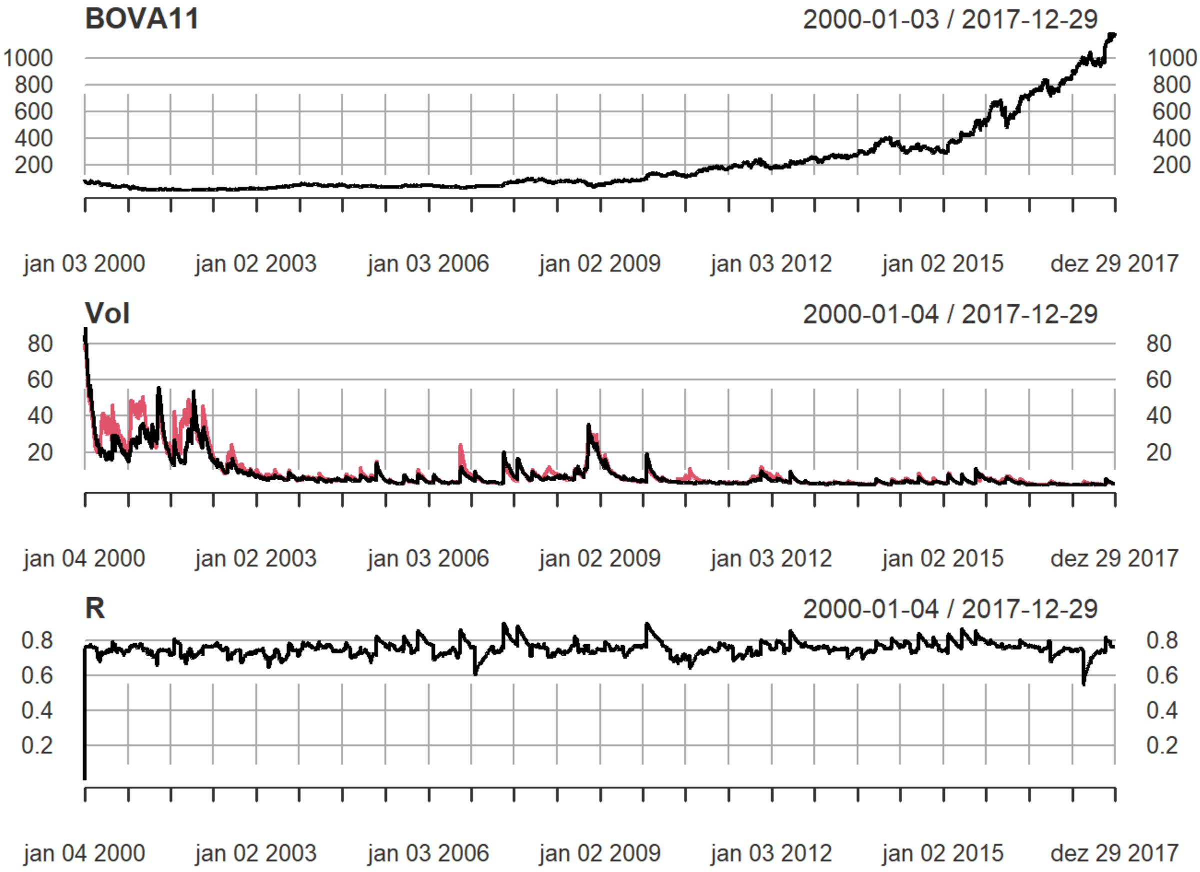

Trace of alpha_1
Iterations
Density of alpha_1
N = 4001 Bandwidth = 0.0007363
Trace of beta_1
Iterations
Density of beta_1
N = 4001 Bandwidth = 0.0007605
Trace of gamma_2
Iterations
Density of gamma_2
N = 4001 Bandwidth = 0.002907
Trace of omega_2
Iterations
Density of omega_2
N = 4001 Bandwidth = 0.00161
Trace of alpha_2
Iterations
Density of alpha_2
N = 4001 Bandwidth = 0.0007791
Trace of beta_2
Iterations
Density of beta_2
N = 4001 Bandwidth = 0.0007754



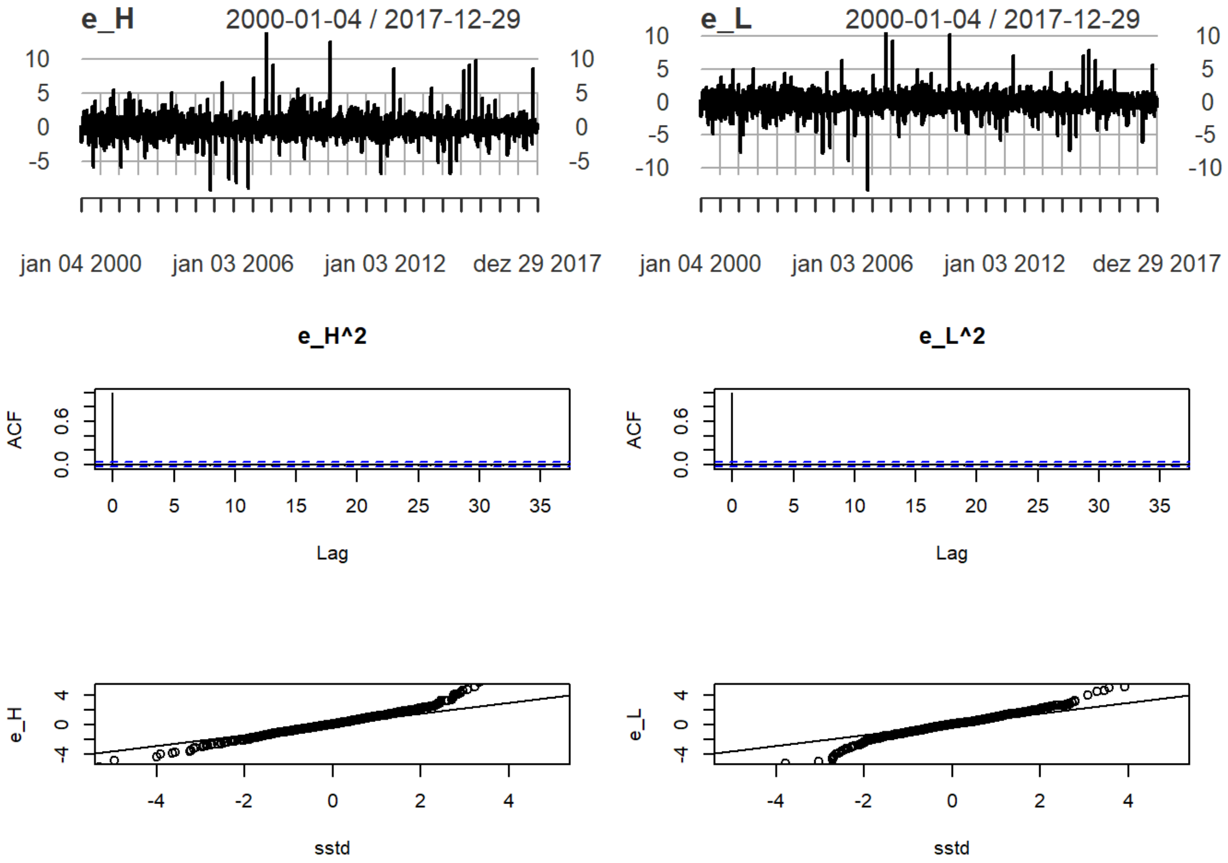

```
## Estimative of parameters
out$MC %>% summary()
```

```
##
## 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      3.70159 0.116252 1.838e-03      0.0055738
## gamma_1 1.04483 0.016877 2.668e-04      0.0009341
## omega_1 0.04475 0.006231 9.851e-05      0.0002925
## alpha_1 0.02717 0.003732 5.901e-05      0.0001918
## beta_1  0.96139 0.003841 6.073e-05      0.0001858
## gamma_2 0.96002 0.014408 2.278e-04      0.0007164
## omega_2 0.05782 0.008086 1.278e-04      0.0004043
## alpha_2 0.03289 0.003945 6.236e-05      0.0002168
## beta_2  0.95566 0.004030 6.372e-05      0.0002118
## a       0.01261 0.003284 5.191e-05      0.0001990
## b       0.95930 0.013532 2.139e-04      0.0007718
##
## 2. Quantiles for each variable:
##
##             2.5%     25%     50%     75%   97.5%
## nu      3.482750 3.62182 3.69779 3.77763 3.93316
## gamma_1 1.012682 1.03326 1.04460 1.05610 1.07901
## omega_1 0.034289 0.04039 0.04405 0.04839 0.05890
## alpha_1 0.020897 0.02450 0.02693 0.02939 0.03543
## beta_1  0.952966 0.95908 0.96161 0.96413 0.96787
## gamma_2 0.932804 0.94976 0.95980 0.96953 0.98960
## omega_2 0.044498 0.05200 0.05703 0.06269 0.07565
## alpha_2 0.026166 0.03016 0.03253 0.03534 0.04184
## beta_2  0.946535 0.95327 0.95606 0.95842 0.96255
## a       0.007031 0.01037 0.01236 0.01454 0.01953
## b       0.925185 0.95263 0.96167 0.96866 0.97895
```

```
## 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
jan 04 2000
jan 03 2006
jan 03 2012
dez 29 2017
e_L
2000-01-04 / 2017-12-29
10
5
0
-5
-10
e_H^2
ACF
0.0
0.6
0
5
10
15
20
25
30
35
Lag
e_L^2
e_H
-4
0
4
-2
2
sstd
e_L

```
# 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  0.495  0.638  0.817  1.067  1.511  1.998  2.749
Low_LB  -0.167 -0.266 -0.375 -0.496 -0.638 -0.817 -1.066 -1.508 -1.994 -2.743
         0.5%
High_UB  3.426
Low_LB  -3.417
```

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

```
              Value
High_UB_HBOP  1.998
High_UB_S1    0.495
Low_LB_B1    -0.496
Low_LB_LBOP  -1.994
High_omega    0.045
High_alpha    0.027
High_beta     0.961
Low_omega     0.058
Low_alpha     0.033
Low_beta      0.956
```