GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

05/02/2021

Input

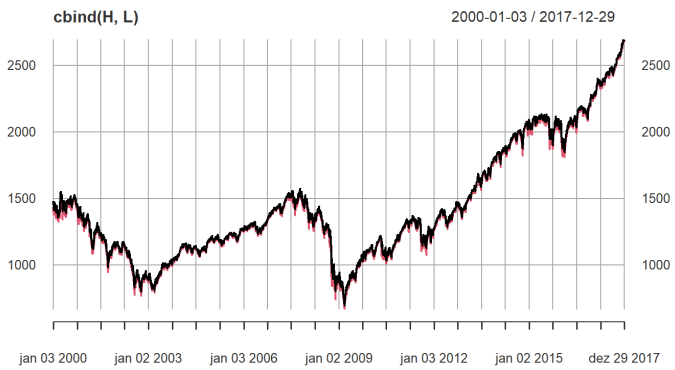
```
symbol = "^GSPC"
from=as.Date('2000-01-01')
to=as.Date('2017-12-31')
C_Trend = 0.95
C_Reaction = 0.50
```

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))
```

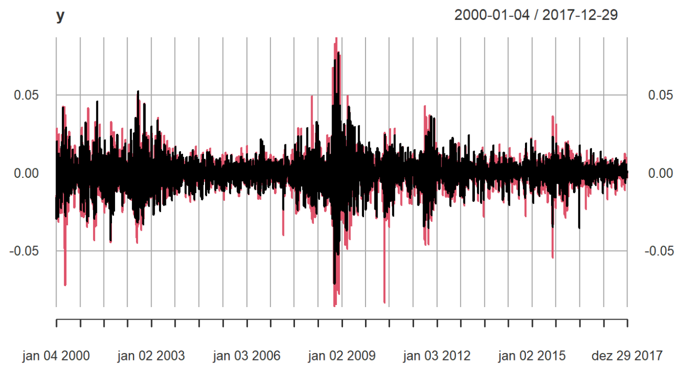


Returns

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

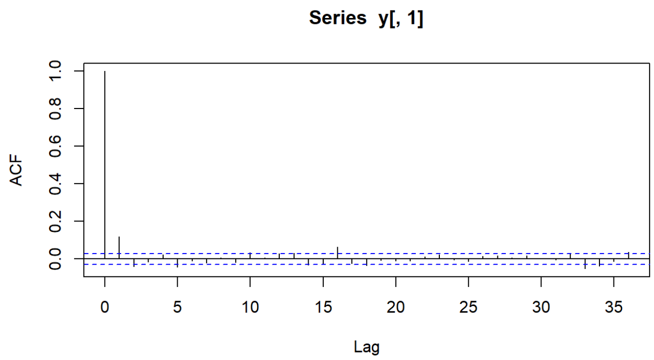
```
##           GSPC.High  GSPC.Low
## GSPC.High 1.0000000 0.6716581
## GSPC.Low  0.6716581 1.0000000
```

```
plot(y)
```

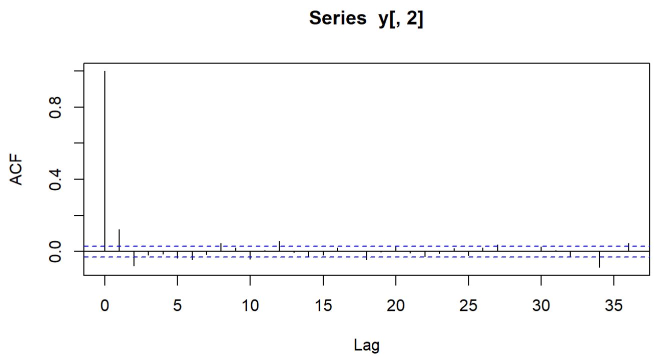


Autocorrelation

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

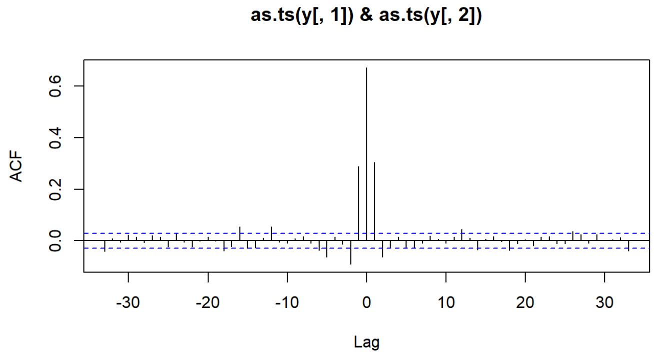



```
acf(y[,2])
```



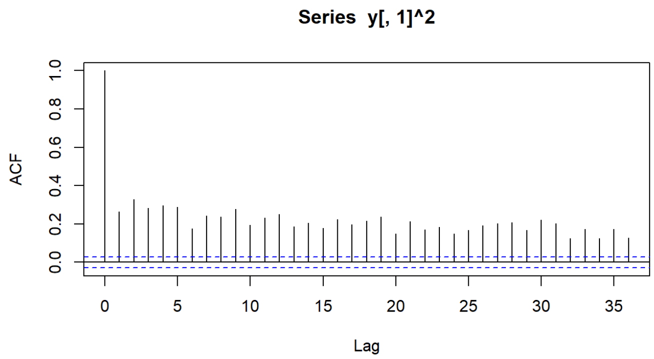
Cross correlation

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

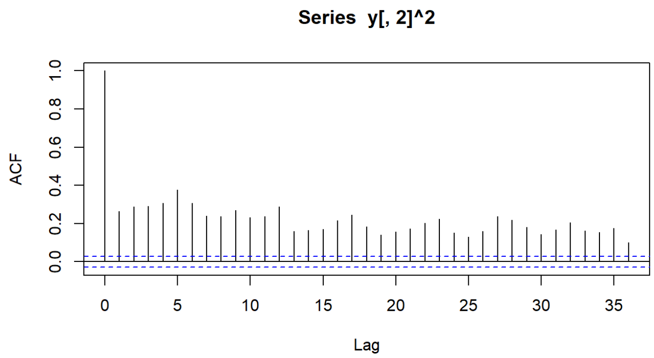


Volatility verification

```
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.
```

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

```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.15
## lambda: 0.32
## Accept Rate: 0.2
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

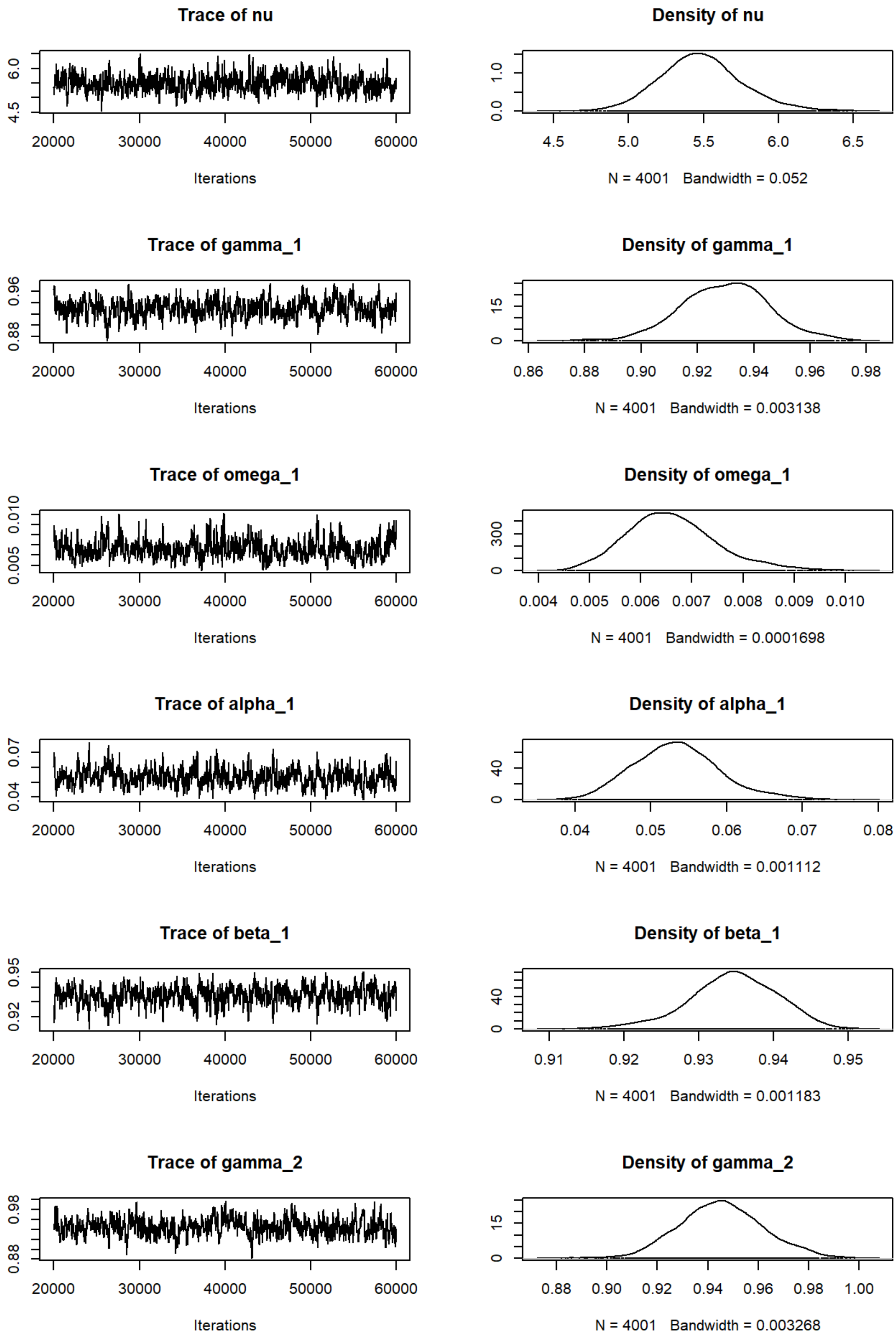
```
out2 <- increaseSim(out, nSim=50000)
```

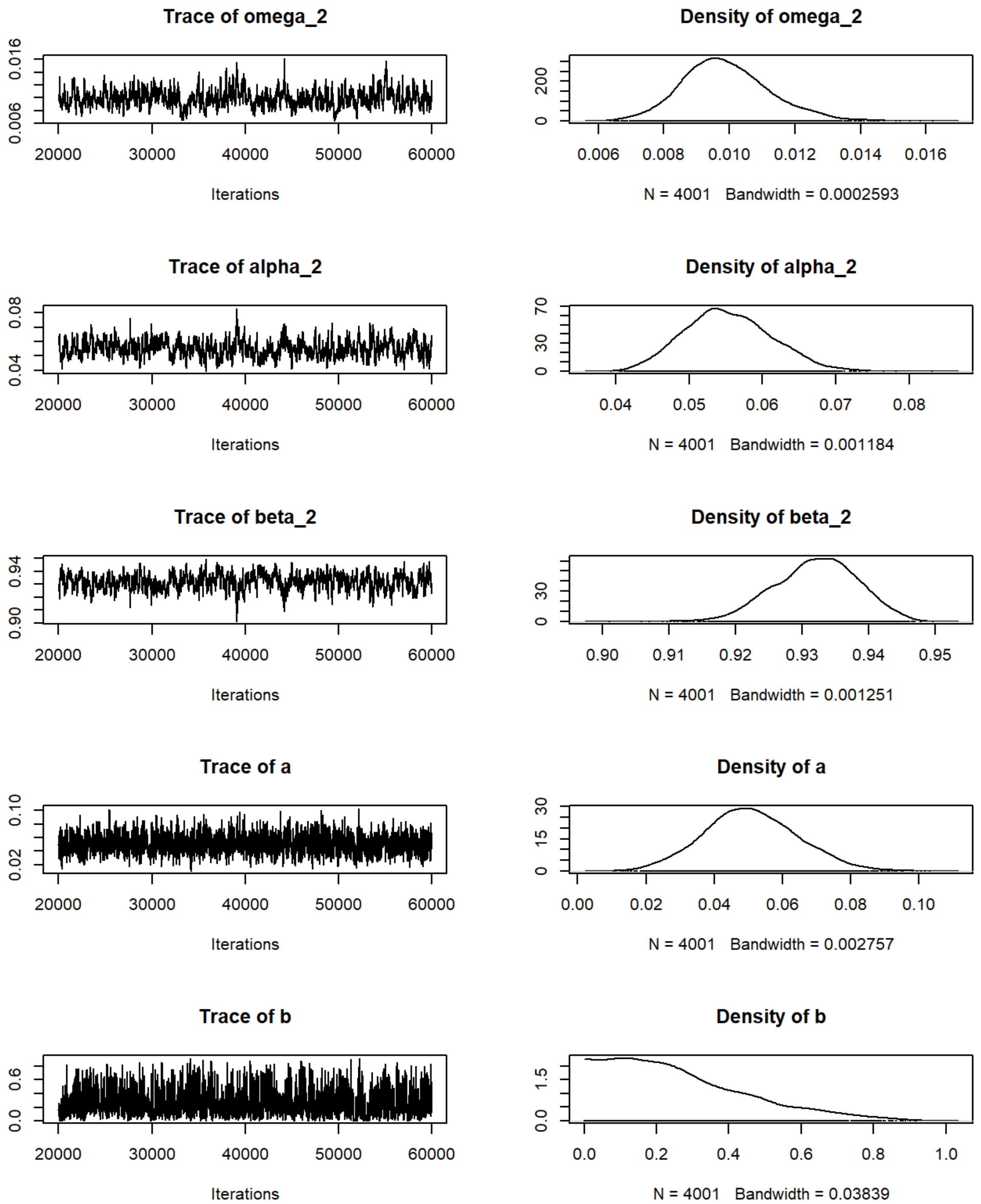
```
## Calibrating the Lambda coefficient:
## lambda: 0.32
## Accept Rate: 0.21
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

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

end <- Sys.time()

# elapsed time
end-start
```

```
## Time difference of 2.643491 mins
```

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

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

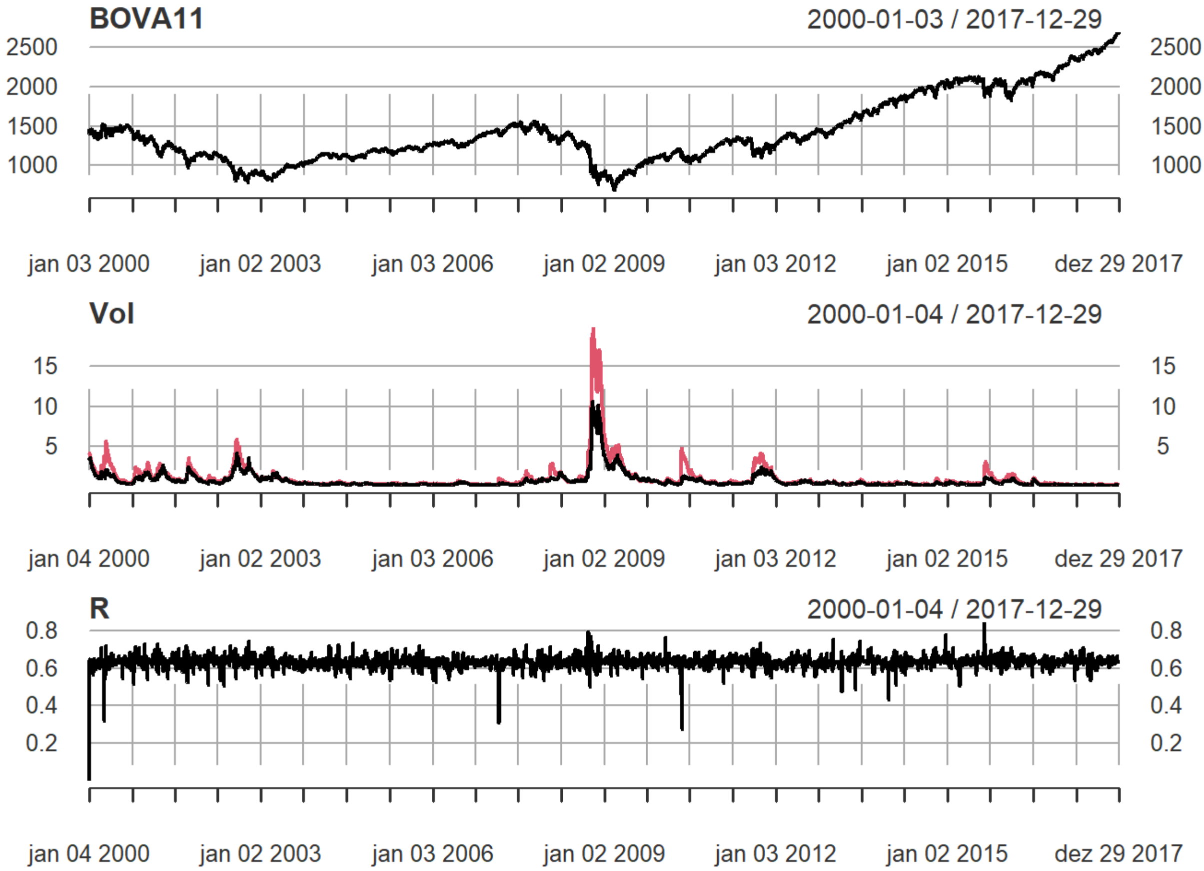

Trace of nu
4.5
6.0
20000 30000 40000 50000 60000
Iterations
Density of nu
0.0
1.0
4.5 5.0 5.5 6.0 6.5
N = 4001 Bandwidth = 0.052
Trace of gamma_1
0.88
0.96
20000 30000 40000 50000 60000
Iterations
Density of gamma_1
0
15
0.86 0.88 0.90 0.92 0.94 0.96 0.98
N = 4001 Bandwidth = 0.003138
Trace of omega_1
0.005
0.010
20000 30000 40000 50000 60000
Iterations
Density of omega_1
0
300
0.004 0.005 0.006 0.007 0.008 0.009 0.010
N = 4001 Bandwidth = 0.0001698
Trace of alpha_1
0.04
0.07
20000 30000 40000 50000 60000
Iterations
Density of alpha_1
0
40
0.04 0.05 0.06 0.07 0.08
N = 4001 Bandwidth = 0.001112
Trace of beta_1
0.92
0.95
20000 30000 40000 50000 60000
Iterations
Density of beta_1
0
40
0.91 0.92 0.93 0.94 0.95
N = 4001 Bandwidth = 0.001183
Trace of gamma_2
0.88
0.98
20000 30000 40000 50000 60000
Iterations
Density of gamma_2
0
15
0.88 0.90 0.92 0.94 0.96 0.98 1.00
N = 4001 Bandwidth = 0.003268



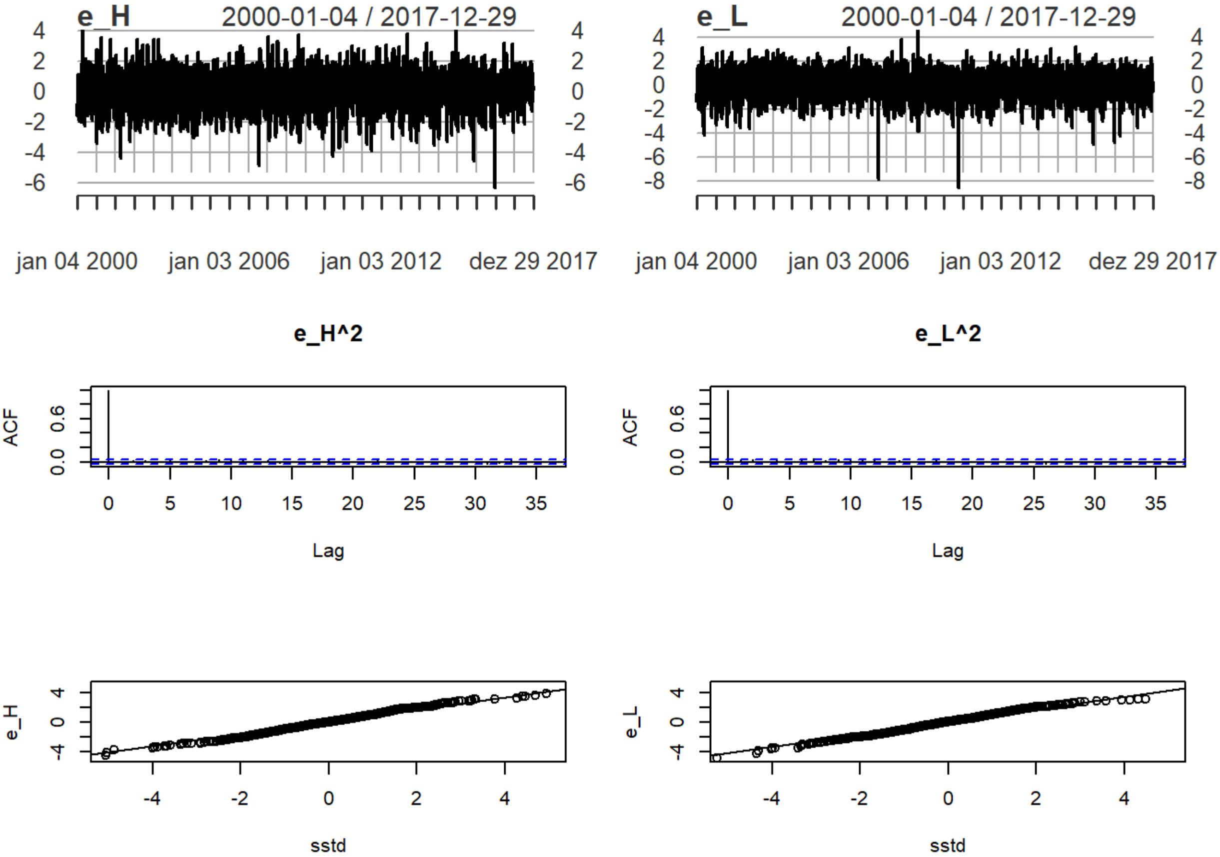

```
## Estimative of parameters
out$MC %>% summary()
```

```
##
## 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      5.484110 0.2749170 4.346e-03      1.545e-02
## gamma_1 0.929884 0.0155683 2.461e-04      1.047e-03
## omega_1 0.006611 0.0008803 1.392e-05      5.839e-05
## alpha_1 0.053401 0.0056697 8.963e-05      3.888e-04
## beta_1  0.934552 0.0059045 9.335e-05      3.974e-04
## gamma_2 0.945270 0.0165553 2.617e-04      1.115e-03
## omega_2 0.009893 0.0013258 2.096e-05      8.478e-05
## alpha_2 0.055290 0.0058670 9.275e-05      3.676e-04
## beta_2  0.932069 0.0061993 9.801e-05      3.897e-04
## a       0.050838 0.0136997 2.166e-04      4.202e-04
## b       0.261790 0.1902750 3.008e-03      6.299e-03
##
## 2. Quantiles for each variable:
##
##             2.5%      25%      50%      75%    97.5%
## nu      4.974331 5.302969 5.470750 5.648316 6.075065
## gamma_1 0.899489 0.919370 0.930289 0.940210 0.961446
## omega_1 0.005038 0.006006 0.006548 0.007134 0.008583
## alpha_1 0.043134 0.049500 0.053213 0.056884 0.066007
## beta_1  0.921490 0.930844 0.934717 0.938697 0.945008
## gamma_2 0.914621 0.934280 0.945072 0.955986 0.979058
## omega_2 0.007501 0.008965 0.009765 0.010687 0.012743
## alpha_2 0.044591 0.051191 0.054914 0.059159 0.067069
## beta_2  0.919653 0.928016 0.932435 0.936418 0.943486
## a       0.024742 0.041624 0.050204 0.059936 0.078603
## b       0.008834 0.112184 0.223772 0.374810 0.713938
```

```
## 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
e_L 2000-01-04 / 2017-12-29
jan 04 2000
jan 03 2006
jan 03 2012
dez 29 2017
e_H^2
e_L^2
ACF
Lag
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  0.241  0.350  0.465  0.591  0.734  0.908  1.140  1.524  1.912  2.454
Low_LB  -0.190 -0.304 -0.427 -0.564 -0.721 -0.915 -1.176 -1.613 -2.058 -2.683
          0.5%
High_UB  2.901
Low_LB  -3.200
```

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

```
              Value
High_UB_HBOP  1.912
High_UB_S1    0.591
Low_LB_B1    -0.564
Low_LB_LBOP  -2.058
High_omega    0.007
High_alpha    0.053
High_beta     0.935
Low_omega     0.010
Low_alpha     0.055
Low_beta      0.932
```