GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

05/02/2021

Input

```
symbol = "BA"
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

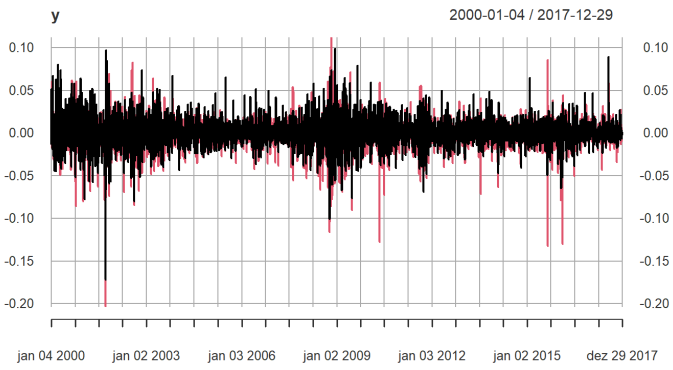
```
H <- Hi(x)
L <- Lo(x)
C <- Cl(x)
plot(cbind(H,L))
```

Returns

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

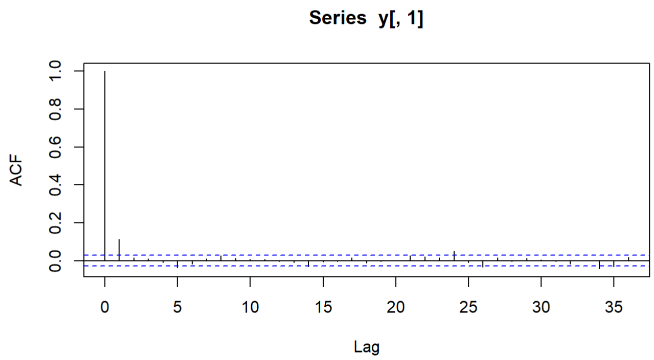
```
##           BA.High    BA.Low
## BA.High 1.0000000 0.7220466
## BA.Low  0.7220466 1.0000000
```

```
plot(y)
```

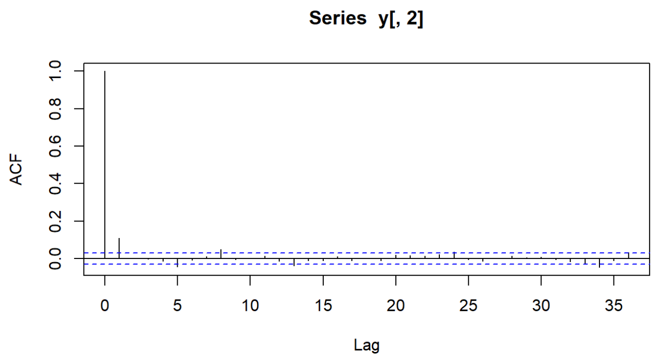


Autocorrelation

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

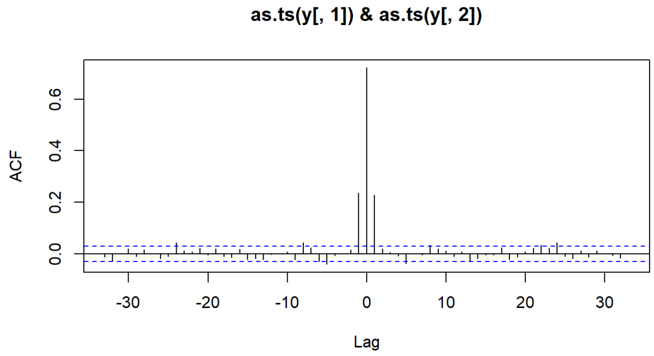



```
acf(y[,2])
```



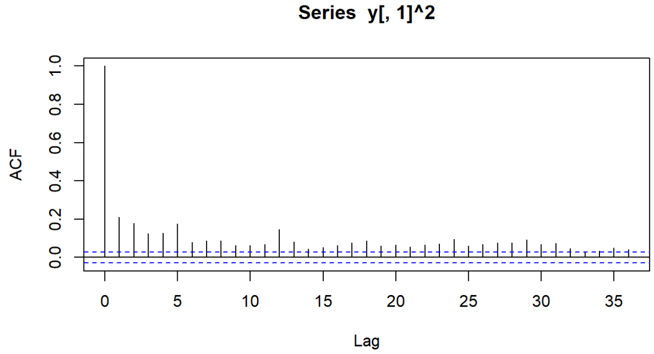
Cross correlation

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

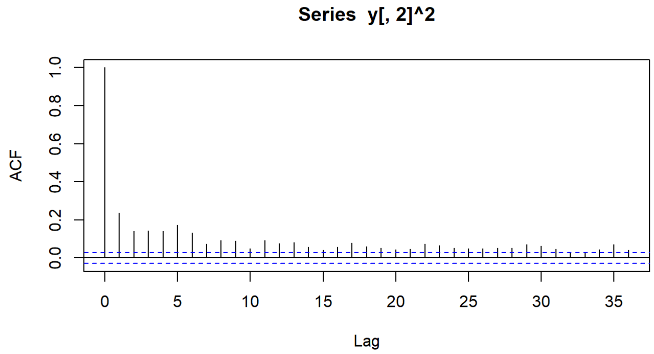


Volatility verification

```
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
##   0.31   0.09   0.20   0.08   0.11   0.10   0.16   0.10   0.14   0.31   0.66
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.32   0.19   0.21   0.13   0.18   0.18   0.16   0.17   0.21   0.34   0.59
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.31   0.17   0.19   0.19   0.18   0.18   0.17   0.17   0.22   0.32   0.48
## 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: 0.45
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

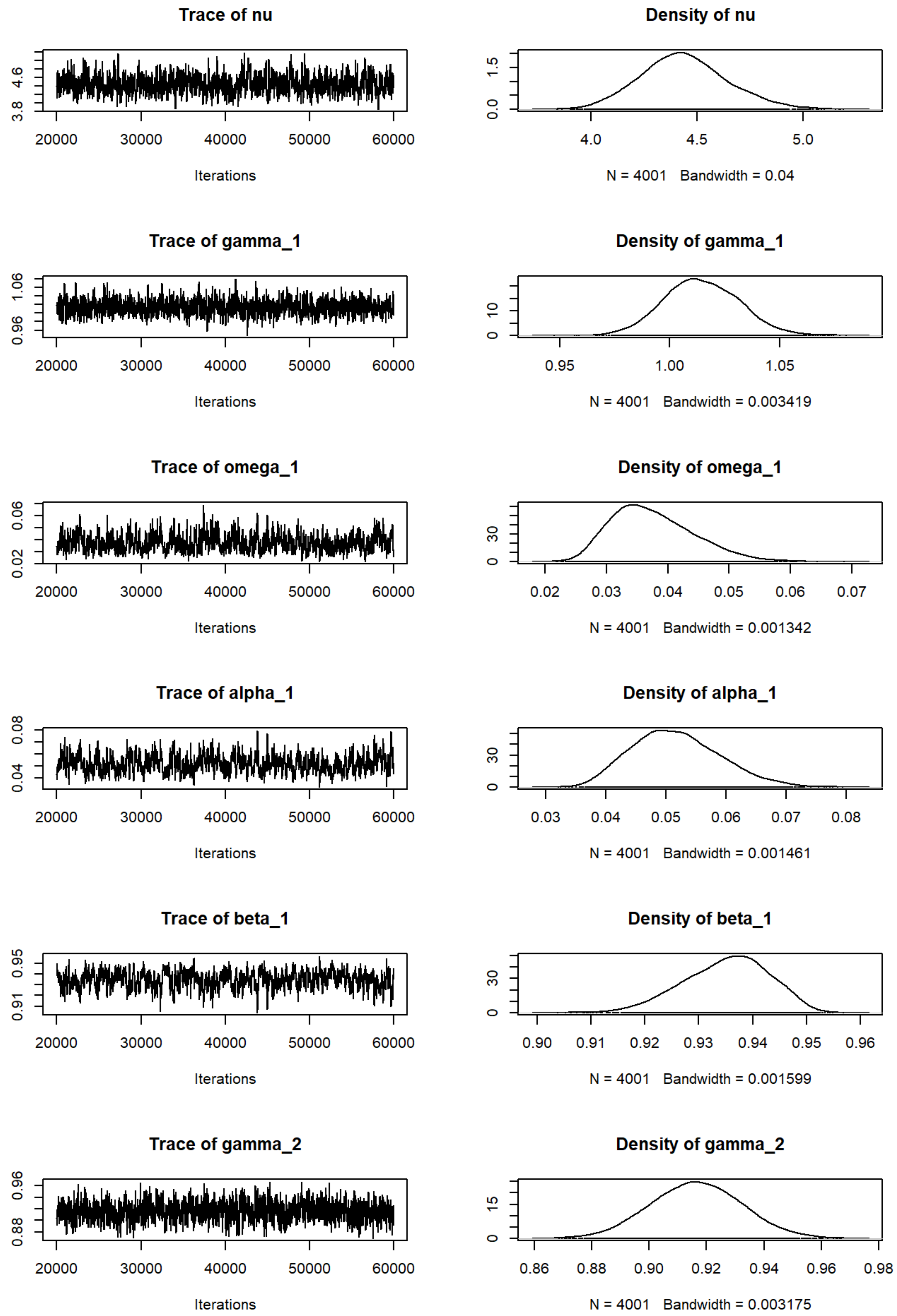
```
out2 <- increaseSim(out, nSim=50000)
```

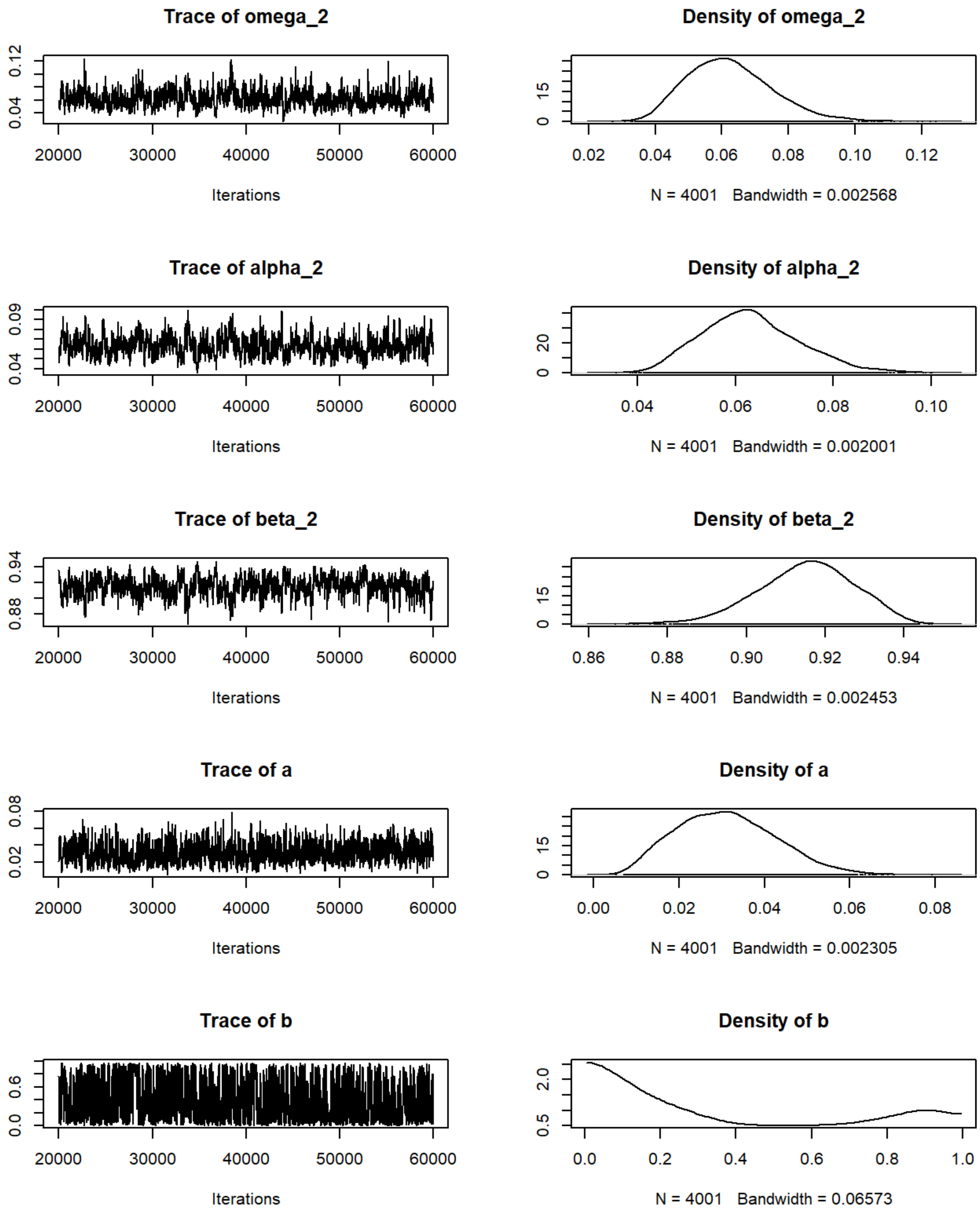
```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.46
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

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

end <- Sys.time()

# elapsed time
end-start
```

```
## Time difference of 13.35493 mins
```

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

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

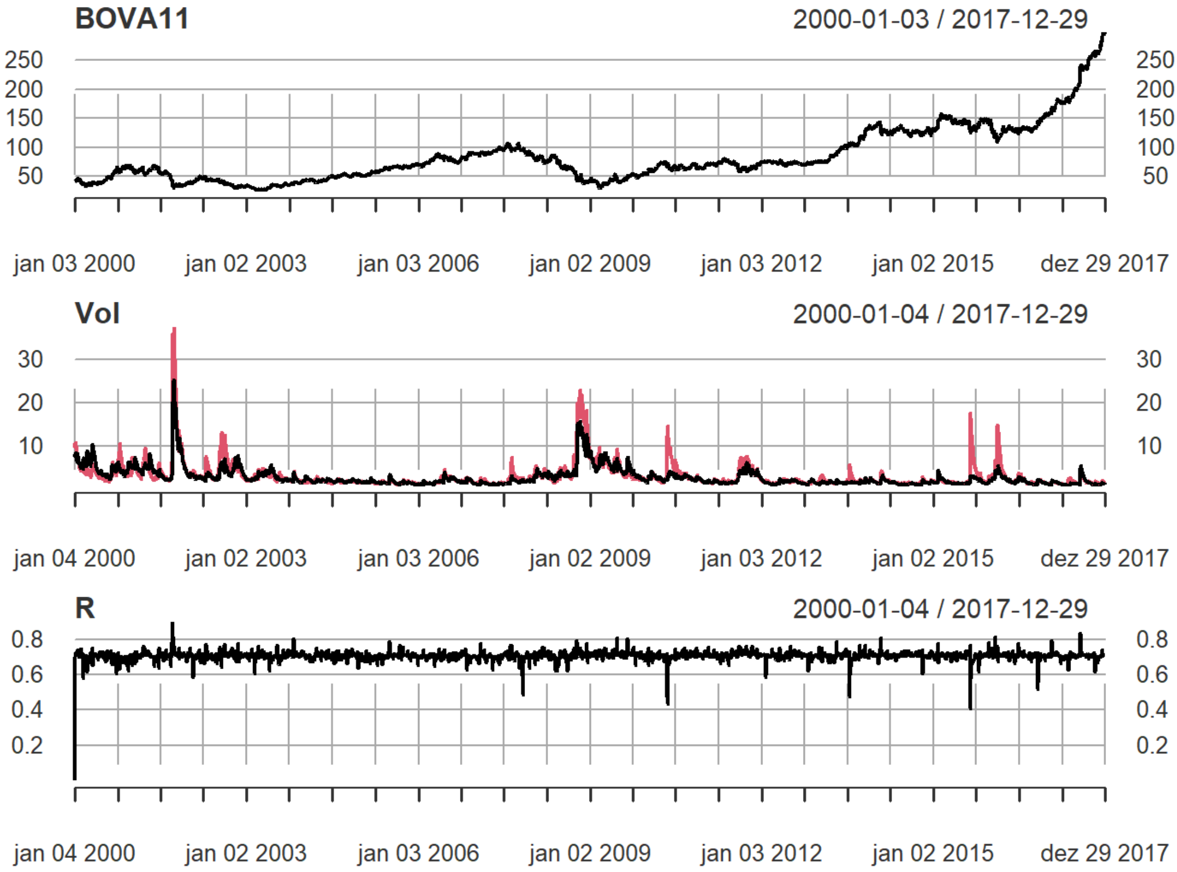

Trace of nu
3.8 4.6
20000 30000 40000 50000 60000
Iterations
Density of nu
0.0 1.5
4.0 4.5 5.0
N = 4001 Bandwidth = 0.04
Trace of gamma_1
0.96 1.06
20000 30000 40000 50000 60000
Iterations
Density of gamma_1
0 10
0.95 1.00 1.05
N = 4001 Bandwidth = 0.003419
Trace of omega_1
0.02 0.06
20000 30000 40000 50000 60000
Iterations
Density of omega_1
0 30
0.02 0.03 0.04 0.05 0.06 0.07
N = 4001 Bandwidth = 0.001342
Trace of alpha_1
0.04 0.08
20000 30000 40000 50000 60000
Iterations
Density of alpha_1
0 30
0.03 0.04 0.05 0.06 0.07 0.08
N = 4001 Bandwidth = 0.001461
Trace of beta_1
0.91 0.95
20000 30000 40000 50000 60000
Iterations
Density of beta_1
0 30
0.90 0.91 0.92 0.93 0.94 0.95 0.96
N = 4001 Bandwidth = 0.001599
Trace of gamma_2
0.88 0.96
20000 30000 40000 50000 60000
Iterations
Density of gamma_2
0 15
0.86 0.88 0.90 0.92 0.94 0.96 0.98
N = 4001 Bandwidth = 0.003175



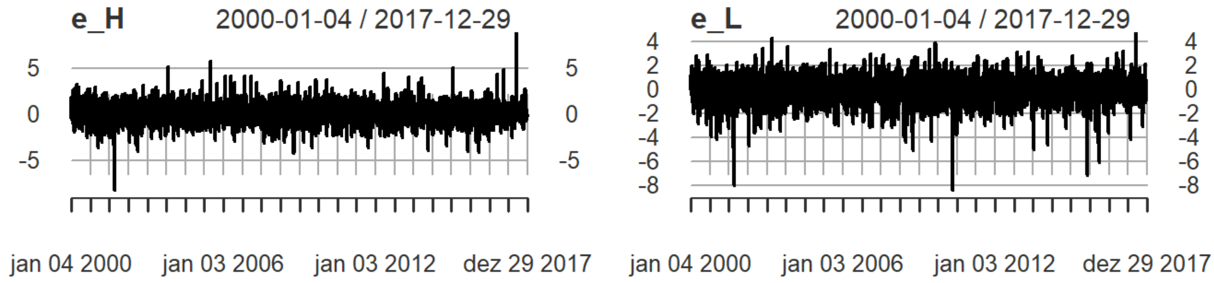

```
## Estimative of parameters
out$MC %>% summary()
```

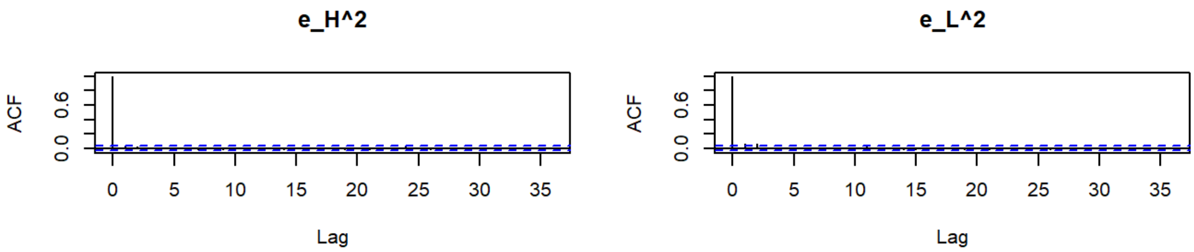
```
##
## Iterations = 20000:60000
## Thinning interval = 10
## Number of chains = 1
## Sample size per chain = 4001
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##            Mean       SD  Naive SE Time-series SE
## nu      4.44039 0.203511 0.0032174      0.0082670
## gamma_1 1.01442 0.016946 0.0002679      0.0006182
## omega_1 0.03753 0.006651 0.0001051      0.0003513
## alpha_1 0.05165 0.007242 0.0001145      0.0004312
## beta_1  0.93505 0.007924 0.0001253      0.0004971
## gamma_2 0.91641 0.015733 0.0002487      0.0005711
## omega_2 0.06234 0.012772 0.0002019      0.0007049
## alpha_2 0.06309 0.009915 0.0001567      0.0005684
## beta_2  0.91517 0.012179 0.0001925      0.0006967
## a       0.03103 0.011425 0.0001806      0.0004130
## b       0.39046 0.325762 0.0051501      0.0128871
##
## 2. Quantiles for each variable:
##
##             2.5%     25%     50%     75%   97.5%
## nu      4.060313 4.30208 4.43031 4.56775 4.87553
## gamma_1 0.982376 1.00260 1.01372 1.02606 1.04860
## omega_1 0.027052 0.03265 0.03669 0.04177 0.05217
## alpha_1 0.039180 0.04643 0.05118 0.05639 0.06728
## beta_1  0.918418 0.92968 0.93565 0.94070 0.94862
## gamma_2 0.885571 0.90563 0.91636 0.92689 0.94740
## omega_2 0.041404 0.05317 0.06129 0.07023 0.09078
## alpha_2 0.046044 0.05605 0.06243 0.06945 0.08416
## beta_2  0.889849 0.90737 0.91578 0.92365 0.93667
## a       0.011441 0.02250 0.03047 0.03883 0.05511
## b       0.008353 0.09631 0.27444 0.71414 0.95159
```

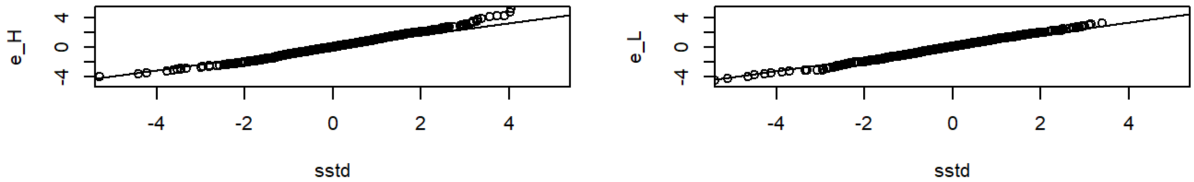
```
## Conditional Correlation
R <-  xts(out$R[,2], order.by=index(y))



## 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
5
0
-5
jan 04 2000
jan 03 2006
jan 03 2012
dez 29 2017
e_L
2000-01-04 / 2017-12-29
4
2
0
-2
-4
-6
-8
jan 04 2000
jan 03 2006
jan 03 2012
dez 29 2017


e_H^2
ACF
0.0
0.6
0 5 10 15 20 25 30 35
Lag
e_L^2
ACF
0.0
0.6
0 5 10 15 20 25 30 35
Lag


e_H
-4
0
4
-4 -2 0 2 4
sstd
e_L
-4
0
4
-4 -2 0 2 4
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  0.193  0.299  0.414  0.541  0.688  0.870  1.119  1.547  1.997  2.659
Low_LB  -0.163 -0.272 -0.391 -0.524 -0.679 -0.872 -1.137 -1.595 -2.080 -2.795
          0.5%
High_UB  3.230
Low_LB  -3.412
```

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

```
              Value
High_UB_HBOP  1.997
High_UB_S1    0.541
Low_LB_B1    -0.524
Low_LB_LBOP  -2.080
High_omega    0.038
High_alpha    0.052
High_beta     0.935
Low_omega     0.062
Low_alpha     0.063
Low_beta      0.915
```