GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

05/02/2021

Input

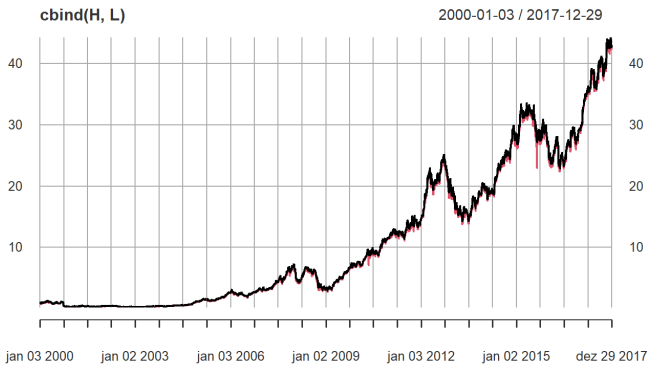
```
symbol = "AAPL"
from=as.Date('2000-01-01')
to=as.Date('2017-12-31')
C_Trend = 0.95
C_Reaction = 0.50
```

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))
```

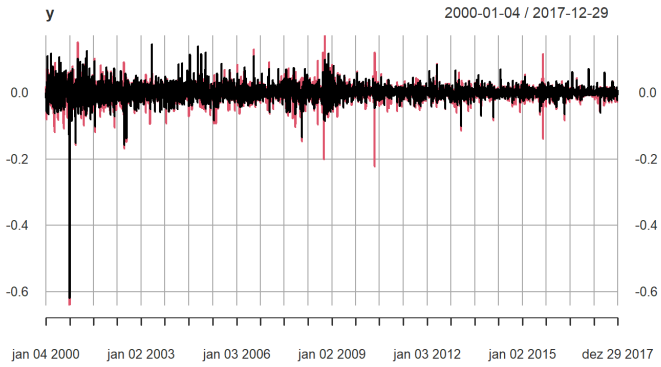


Returns

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

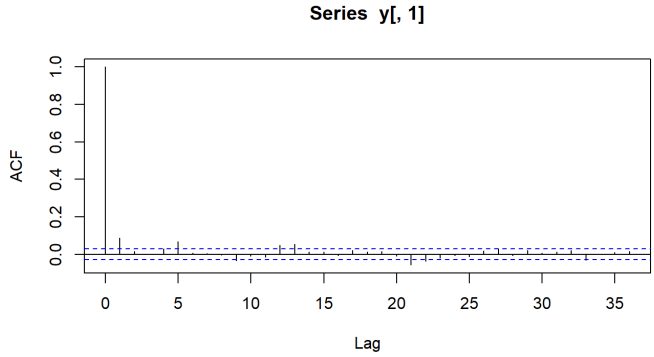
```
##           AAPL.High  AAPL.Low
## AAPL.High 1.0000000 0.7429318
## AAPL.Low  0.7429318 1.0000000
```

```
plot(y)
```

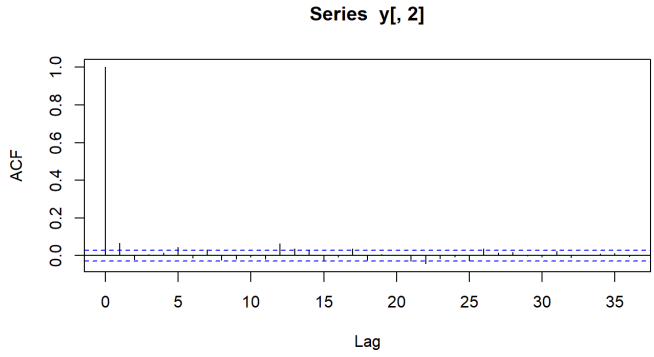


Autocorrelation

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

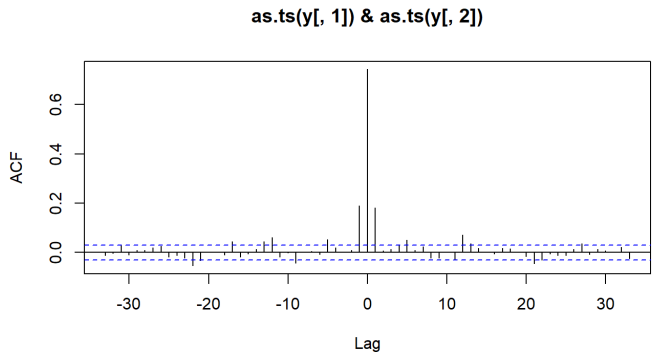



```
acf(y[,2])
```



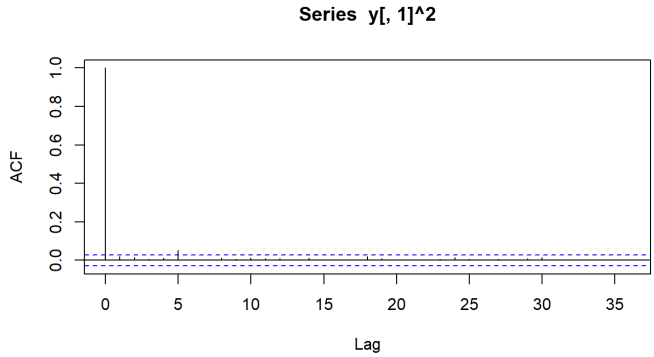
Cross correlation

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

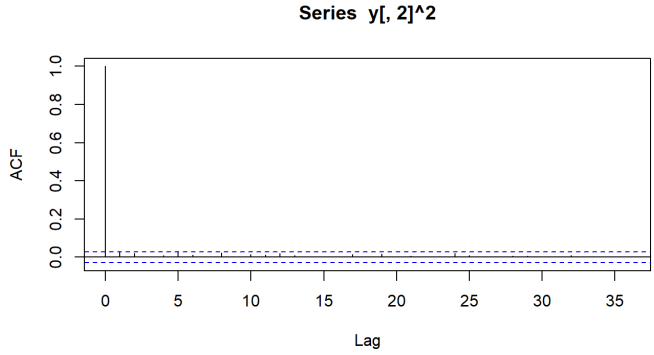


Volatility verification

```
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   0.10   0.22   0.03   0.04   0.09   0.20   0.07   0.09   0.14   0.12
## 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   0.18   0.22   0.04   0.07   0.17   0.21   0.11   0.15   0.25   0.20
## 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   0.17   0.22   0.08   0.12   0.16   0.20   0.17   0.16   0.27   0.19
## 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   0.11   0.18   0.18   0.21   0.17   0.16   0.26   0.20
## 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   0.23   0.22   0.19   0.15   0.22   0.17   0.17   0.26   0.21
## 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   0.17   0.22   0.20   0.18   0.27   0.22   0.18   0.16   0.23   0.19
## 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: 0.46
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

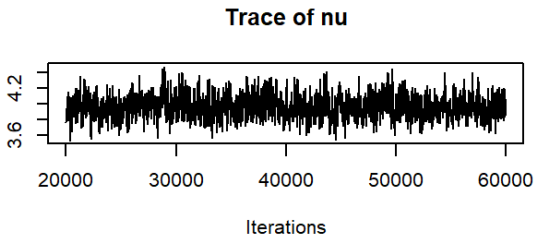
```
out2 <- increaseSim(out, nSim=50000)
```

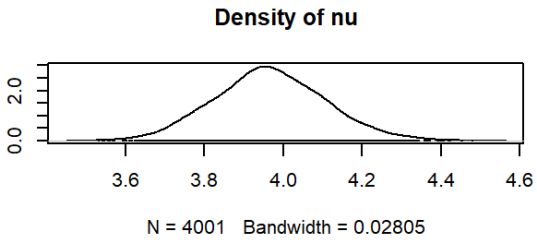
```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.46
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

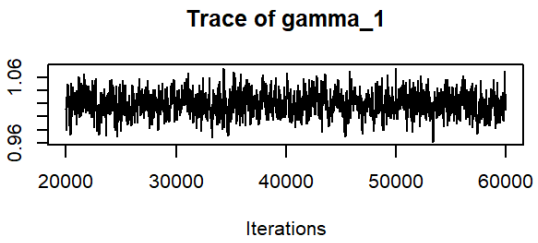
```
out <- window(out2, start=20000, thin=10)
rm(out2)

end <- Sys.time()

# elapsed time
end-start
```

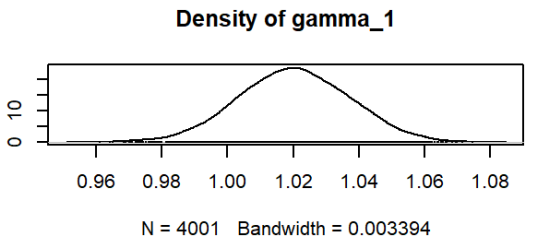
```
## Time difference of 19.40682 mins
```

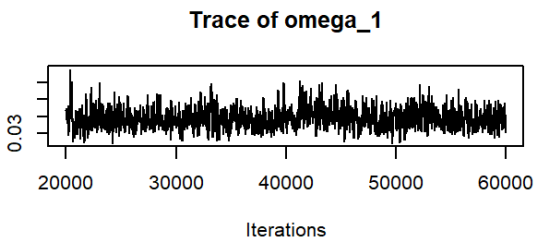
```
## Estimative of parameters
parEst <- summary(out)$statistics[,'Mean']

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

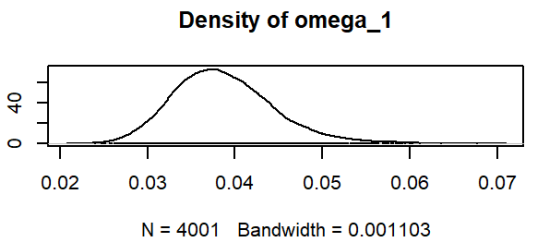


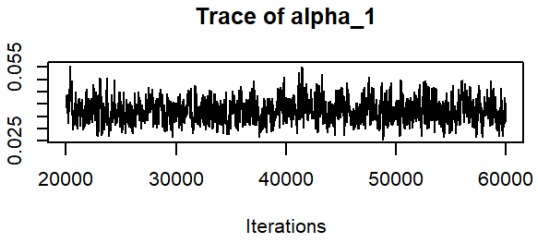


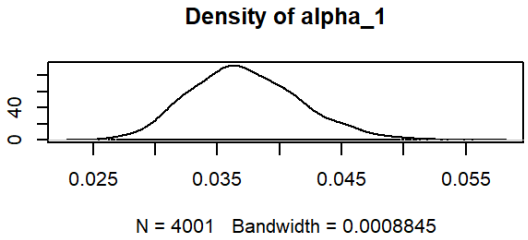


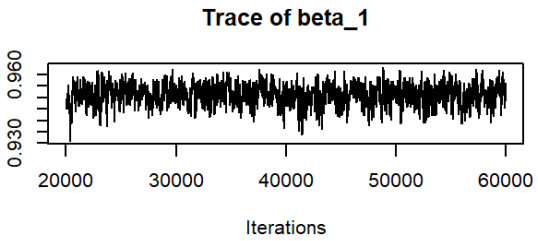


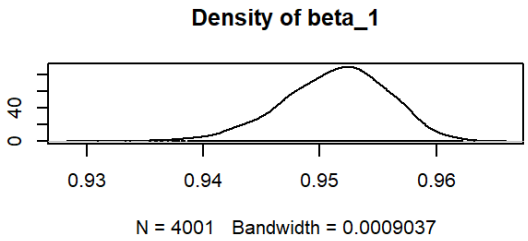


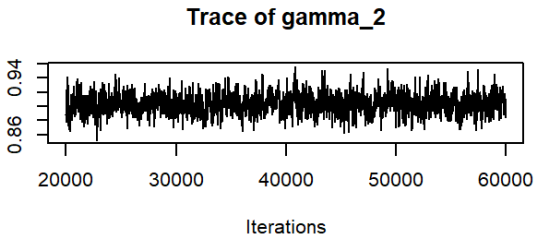


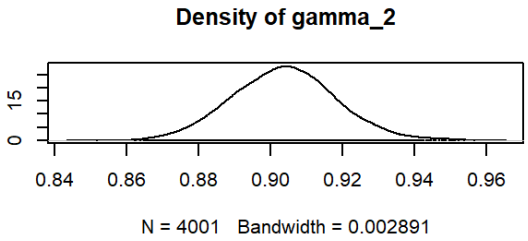

Trace of alpha_1
0.025
0.055
20000
30000
40000
50000
60000
Iterations

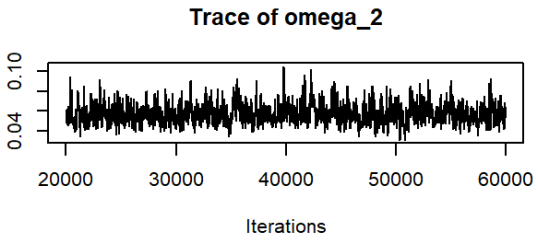

Density of alpha_1
0
40
0.025
0.035
0.045
0.055
N = 4001 Bandwidth = 0.0008845

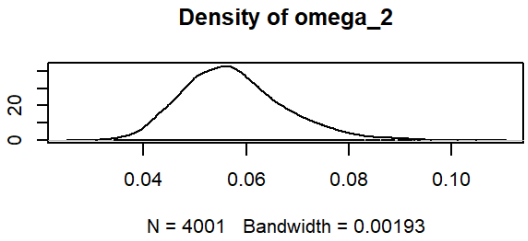

Trace of beta_1
0.930
0.960
20000
30000
40000
50000
60000
Iterations

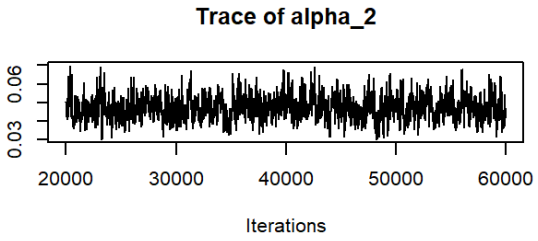

Density of beta_1
0
40
0.93
0.94
0.95
0.96
N = 4001 Bandwidth = 0.0009037

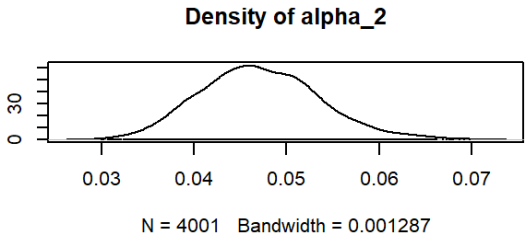

Trace of gamma_2
0.86
0.94
20000
30000
40000
50000
60000
Iterations

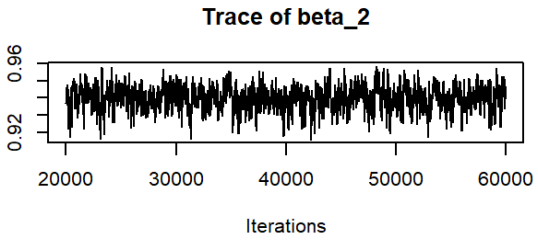

Density of gamma_2
0
15
0.84
0.86
0.88
0.90
0.92
0.94
0.96
N = 4001 Bandwidth = 0.002891

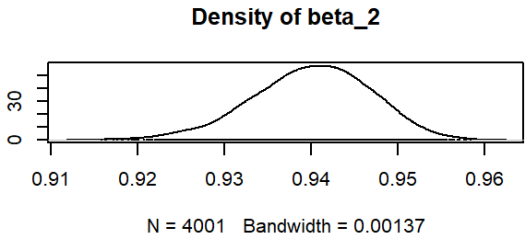

Trace of omega_2
0.04
0.10
20000
30000
40000
50000
60000
Iterations

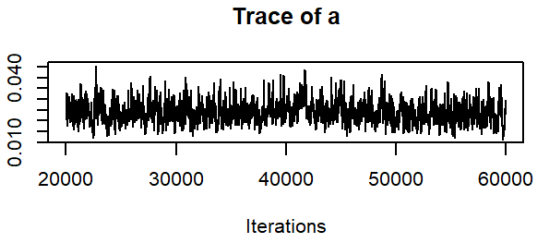

Density of omega_2
0
20
0.04
0.06
0.08
0.10
N = 4001 Bandwidth = 0.00193

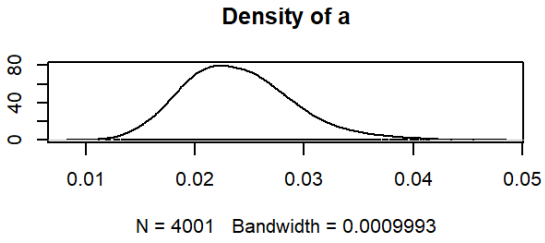

Trace of alpha_2
0.03
0.06
20000
30000
40000
50000
60000
Iterations

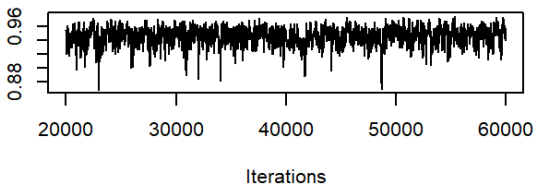

Density of alpha_2
0
30
0.03
0.04
0.05
0.06
0.07
N = 4001 Bandwidth = 0.001287

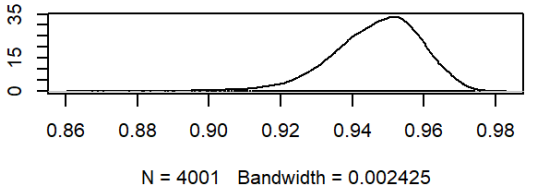

Trace of beta_2
0.92
0.96
20000
30000
40000
50000
60000
Iterations


Density of beta_2
0
30
0.91
0.92
0.93
0.94
0.95
0.96
N = 4001 Bandwidth = 0.00137





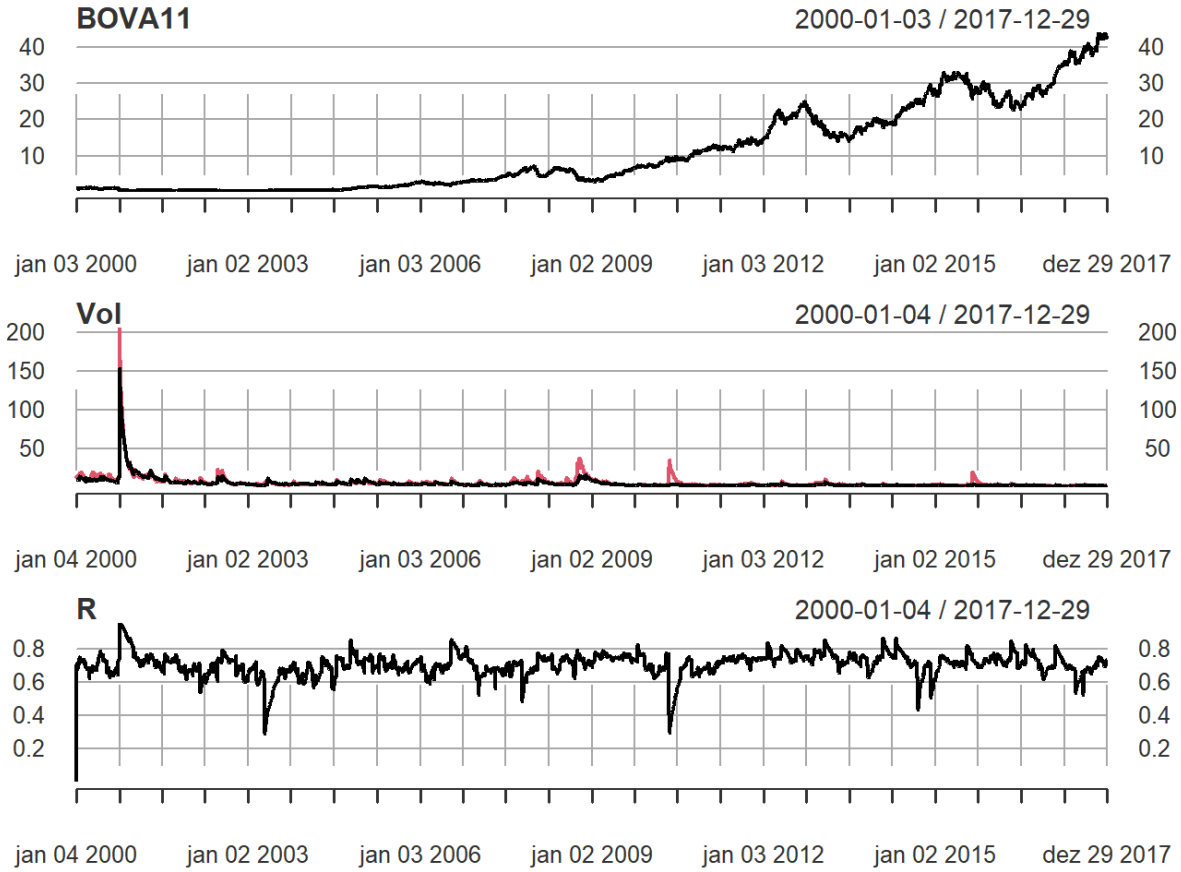




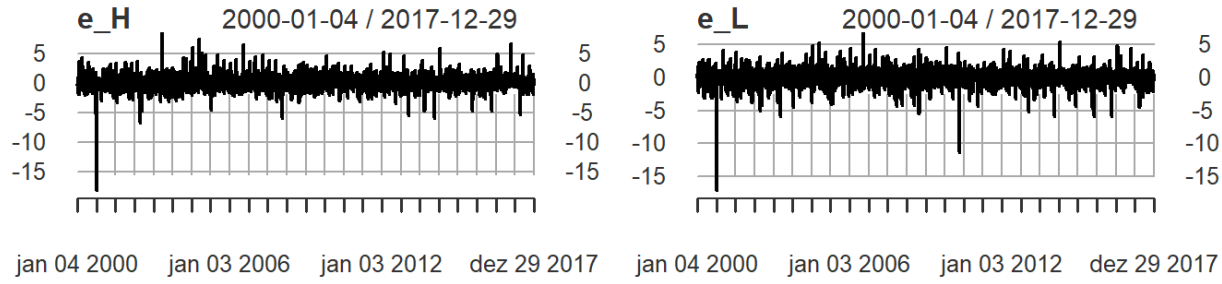

```
## Estimative of parameters
out$MC %>% summary()
```

```
##
## 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      3.96546 0.140956 2.228e-03      0.0052175
## gamma_1 1.02021 0.016821 2.659e-04      0.0007132
## omega_1 0.03866 0.005655 8.940e-05      0.0002300
## alpha_1 0.03722 0.004384 6.930e-05      0.0001980
## beta_1  0.95130 0.004479 7.081e-05      0.0001997
## gamma_2 0.90374 0.014407 2.278e-04      0.0005754
## omega_2 0.05746 0.009940 1.571e-04      0.0004568
## alpha_2 0.04711 0.006376 1.008e-04      0.0003090
## beta_2  0.94004 0.006788 1.073e-04      0.0003358
## a       0.02389 0.004952 7.829e-05      0.0002153
## b       0.94646 0.012736 2.014e-04      0.0005192
##
## 2. Quantiles for each variable:
##
##            2.5%     25%     50%     75%   97.5%
## nu      3.69929 3.87136 3.96227 4.05763 4.24995
## gamma_1 0.98751 1.00884 1.02020 1.03176 1.05296
## omega_1 0.02909 0.03470 0.03815 0.04202 0.05125
## alpha_1 0.02953 0.03409 0.03693 0.04006 0.04643
## beta_1  0.94200 0.94842 0.95162 0.95447 0.95909
## gamma_2 0.87620 0.89383 0.90371 0.91303 0.93272
## omega_2 0.04108 0.05051 0.05647 0.06333 0.07981
## alpha_2 0.03549 0.04267 0.04675 0.05129 0.06068
## beta_2  0.92557 0.93566 0.94040 0.94484 0.95232
## a       0.01548 0.02032 0.02348 0.02696 0.03491
## b       0.91826 0.93919 0.94794 0.95529 0.96707
```

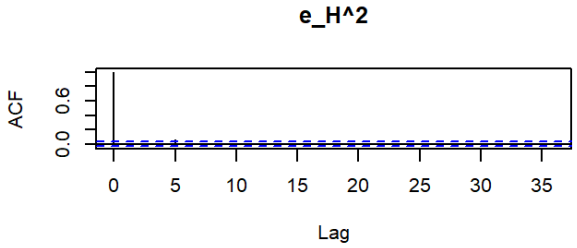
```
## Conditional Correlation
R <-  xts(out$R[,2], order.by=index(y))



## 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")
```

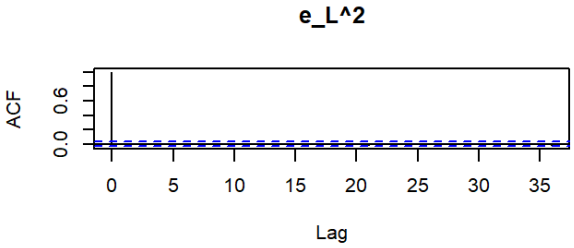


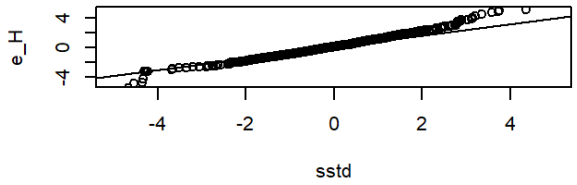

```
## Standard Residuals
r <- mY / sqrt(Vol)

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)
```

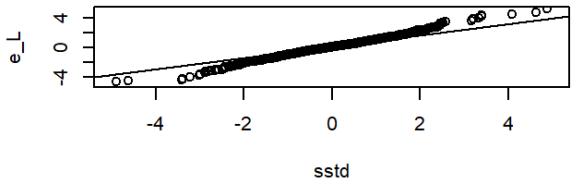

e_H
2000-01-04 / 2017-12-29
5
0
-5
-10
-15
jan 04 2000
jan 03 2006
jan 03 2012
dez 29 2017

e_L
2000-01-04 / 2017-12-29
5
0
-5
-10
-15
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  0.182  0.284  0.394  0.517  0.661  0.839  1.086  1.519  1.986  2.691
Low_LB  -0.148 -0.253 -0.368 -0.497 -0.650 -0.840 -1.106 -1.573 -2.080 -2.849
          0.5%
High_UB  3.316
Low_LB  -3.530
```

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

```
             Value
High_UB_HBOP  1.986
High_UB_S1    0.517
Low_LB_B1    -0.497
Low_LB_LBOP  -2.080
High_omega    0.039
High_alpha    0.037
High_beta     0.951
Low_omega     0.057
Low_alpha     0.047
Low_beta      0.940
```