GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

05/02/2021

Input

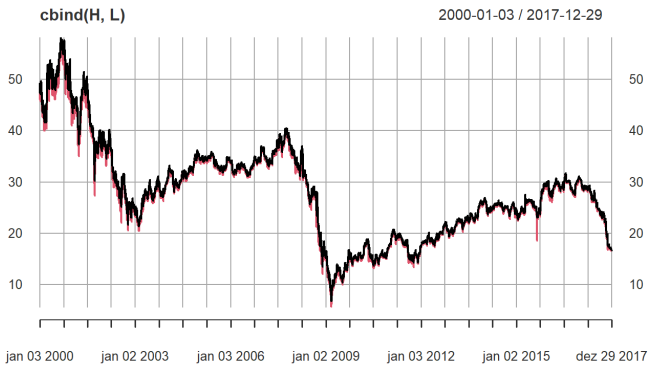
```
symbol = "GE"
from=as.Date('2000-01-01')
to=as.Date('2017-12-31')
C_Trend = 0.95
C_Reaction = 0.50
```

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))
```

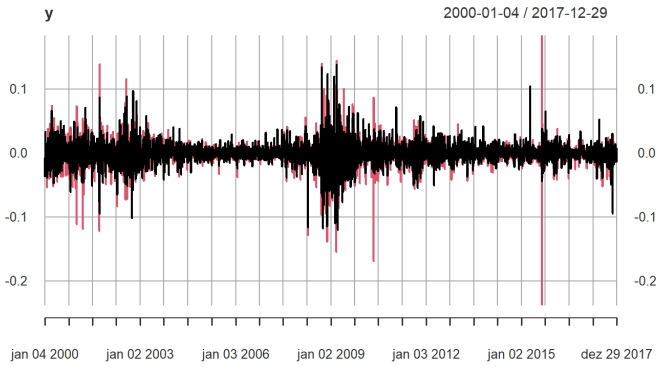


Returns

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

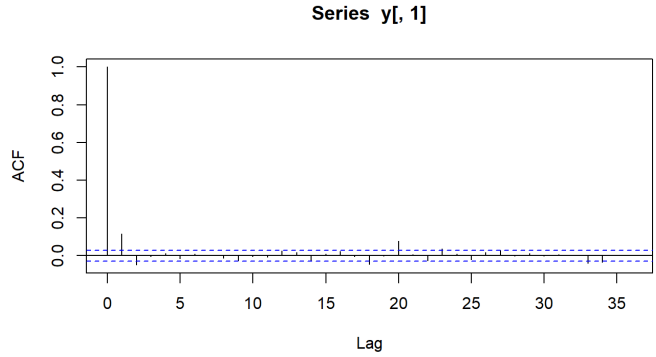
```
##           GE.High    GE.Low
## GE.High 1.0000000 0.6933611
## GE.Low  0.6933611 1.0000000
```

```
plot(y)
```

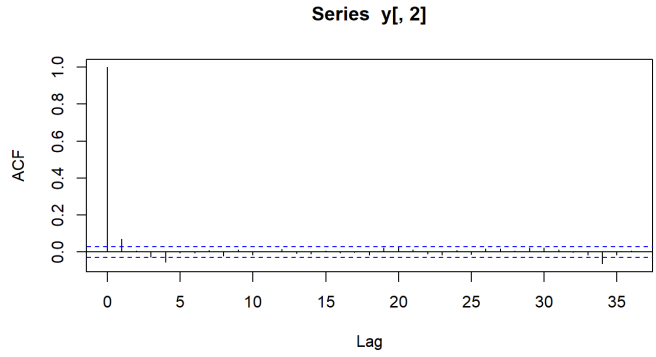


Autocorrelation

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

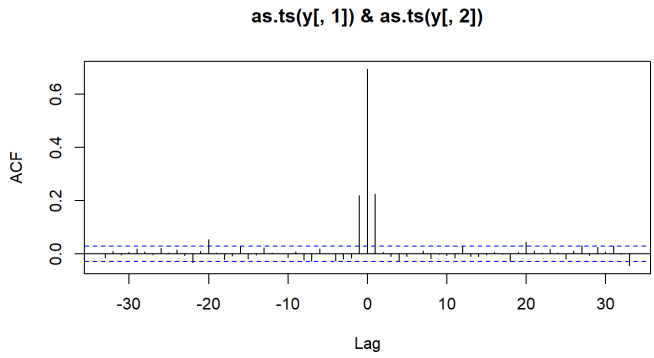



```
acf(y[,2])
```



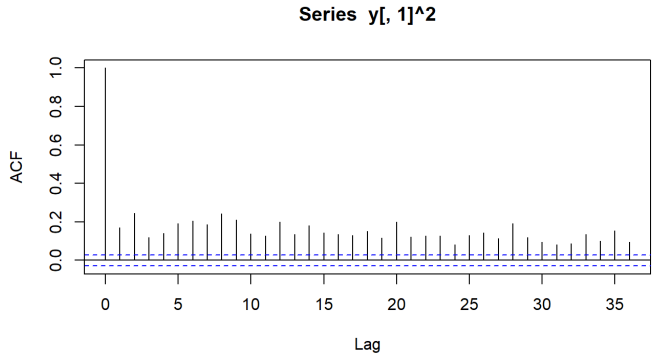
Cross correlation

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

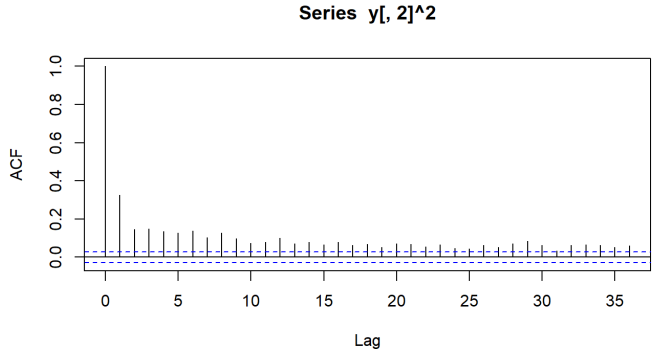


Volatility verification

```
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   0.10   0.22   0.04   0.06   0.10   0.20   0.08   0.11   0.27   0.37
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.30   0.18   0.23   0.07   0.12   0.17   0.21   0.12   0.18   0.28   0.37
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.30   0.16   0.22   0.13   0.18   0.17   0.22   0.23   0.20   0.30   0.36
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.30   0.17   0.21   0.17   0.19   0.18   0.21   0.22   0.20   0.28   0.37
## 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: 0.36
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

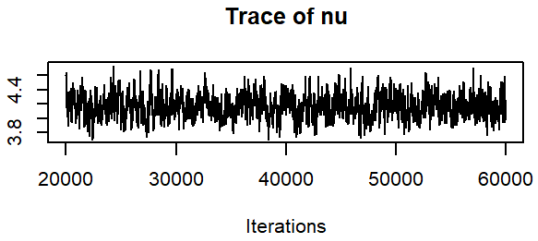
```
out2 <- increaseSim(out, nSim=50000)
```

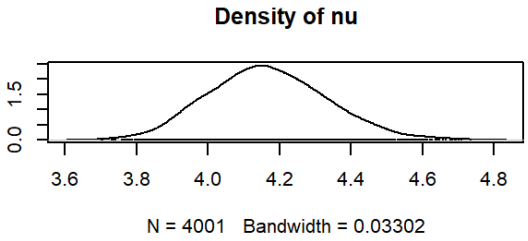
```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.36
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

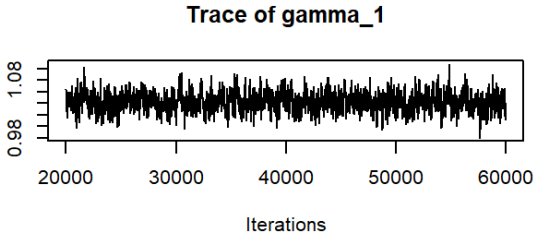
```
out <- window(out2, start=20000, thin=10)
rm(out2)

end <- Sys.time()

# elapsed time
end-start
```

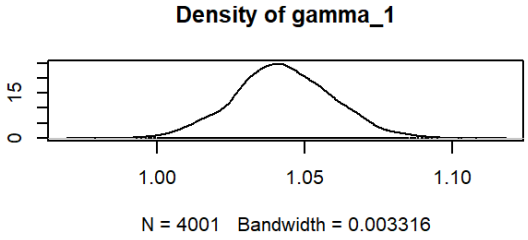
```
## Time difference of 9.949133 mins
```

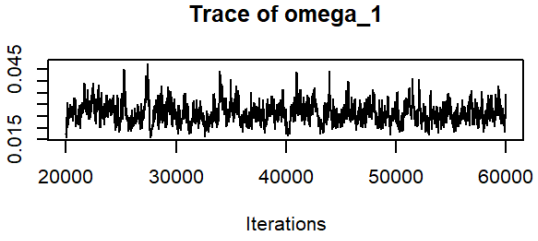
```
## Estimative of parameters
parEst <- summary(out)$statistics[,'Mean']

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

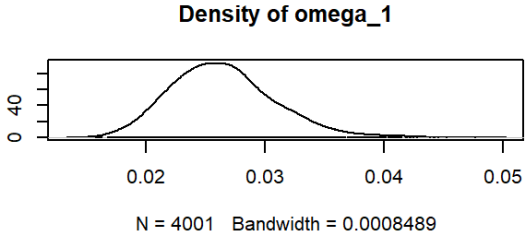


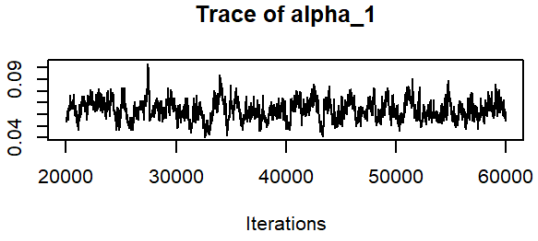


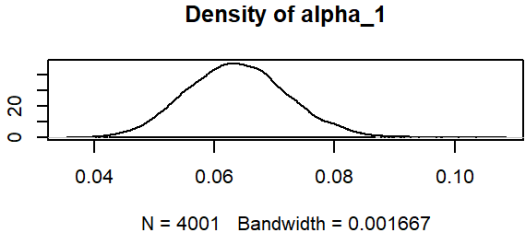


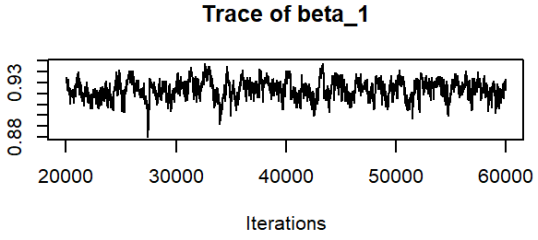


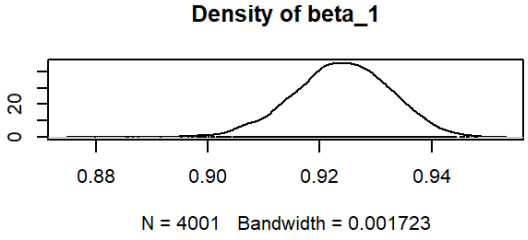


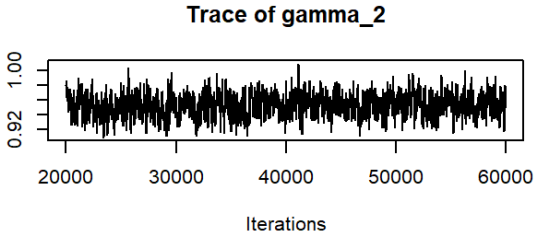


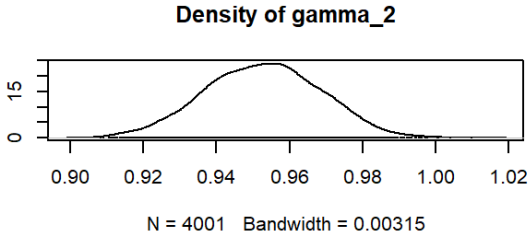


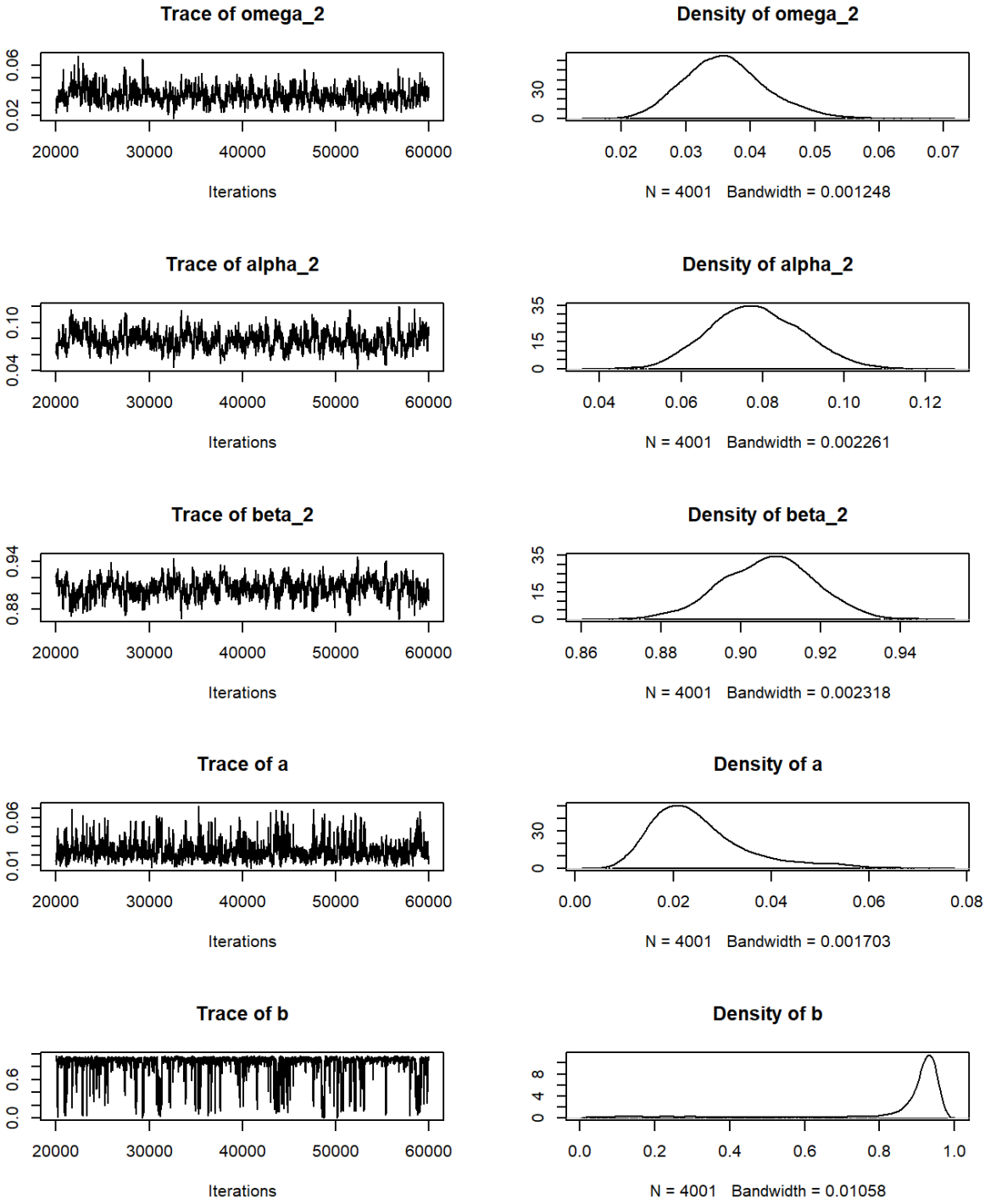








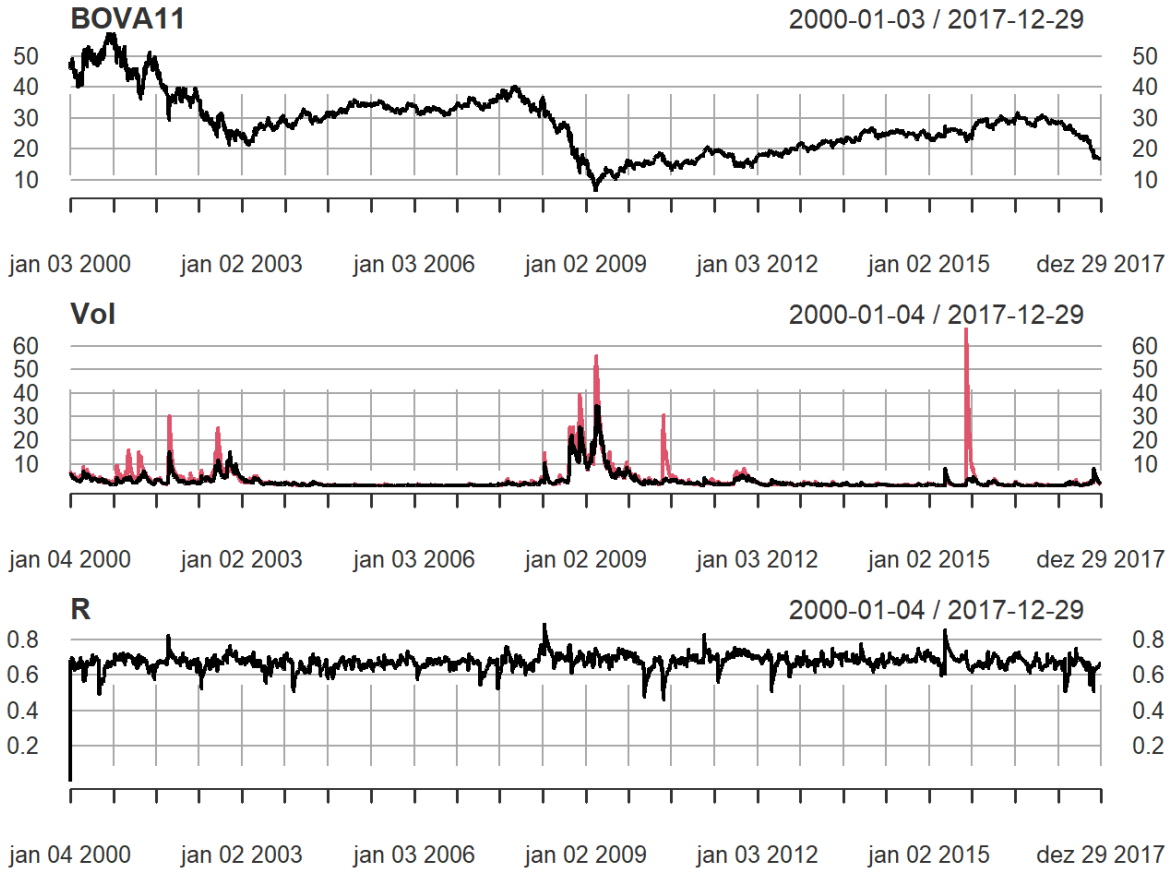




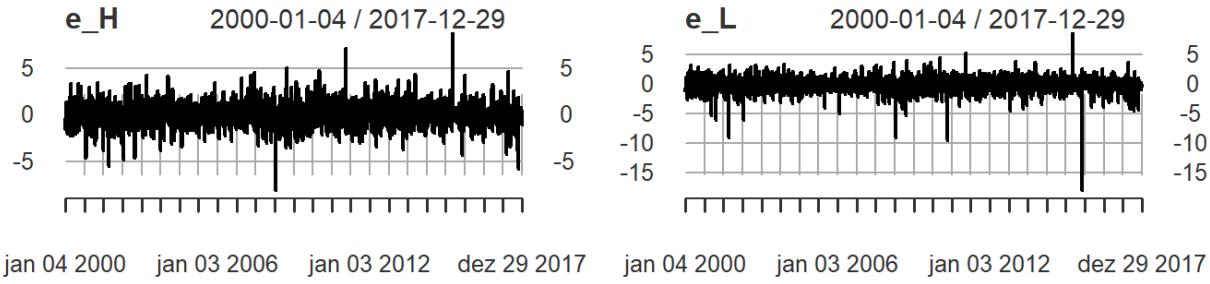

```
## Estimative of parameters
out$MC %>% summary()
```

```
##
## 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      4.16656 0.163725 2.588e-03      0.0078343
## gamma_1 1.04275 0.016508 2.610e-04      0.0006979
## omega_1 0.02646 0.004390 6.941e-05      0.0002943
## alpha_1 0.06385 0.008333 1.317e-04      0.0006989
## beta_1  0.92383 0.008605 1.360e-04      0.0007123
## gamma_2 0.95263 0.015610 2.468e-04      0.0006783
## omega_2 0.03610 0.006368 1.007e-04      0.0003849
## alpha_2 0.07835 0.011205 1.771e-04      0.0007874
## beta_2  0.90687 0.011486 1.816e-04      0.0007910
## a       0.02510 0.009897 1.565e-04      0.0004878
## b       0.82456 0.225925 3.572e-03      0.0135109
##
## 2. Quantiles for each variable:
##
##            2.5%     25%     50%     75%   97.5%
## nu      3.86586 4.05483 4.16022 4.27411 4.50272
## gamma_1 1.01009 1.03178 1.04236 1.05380 1.07564
## omega_1 0.01901 0.02338 0.02615 0.02902 0.03608
## alpha_1 0.04856 0.05810 0.06358 0.06917 0.08064
## beta_1  0.90634 0.91830 0.92412 0.92974 0.93967
## gamma_2 0.92276 0.94162 0.95266 0.96345 0.98223
## omega_2 0.02483 0.03173 0.03578 0.04001 0.04976
## alpha_2 0.05756 0.07057 0.07801 0.08620 0.10072
## beta_2  0.88344 0.89884 0.90715 0.91461 0.92874
## a       0.01134 0.01824 0.02306 0.02955 0.05221
## b       0.11990 0.86754 0.91680 0.93783 0.96347
```

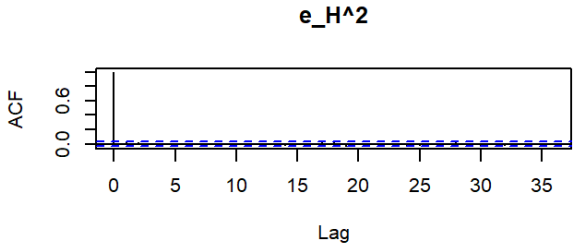
```
## Conditional Correlation
R <-  xts(out$R[,2], order.by=index(y))



## 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")
```

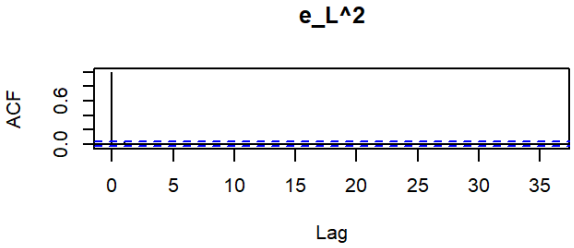


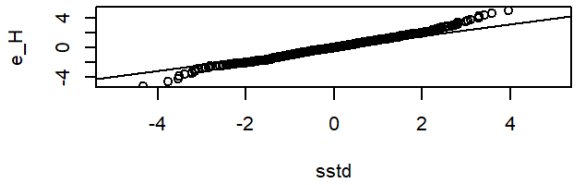

```
## Standard Residuals
r <- mY / sqrt(Vol)

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)
```

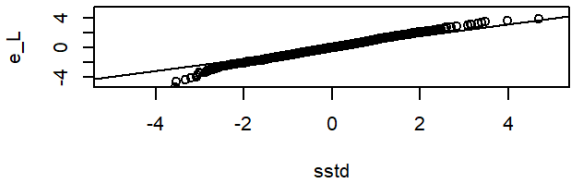

e_H
2000-01-04 / 2017-12-29
5
0
-5
jan 04 2000
jan 03 2006
jan 03 2012
dez 29 2017

e_L
2000-01-04 / 2017-12-29
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  0.177  0.282  0.395  0.522  0.670  0.854  1.108  1.549  2.021  2.726
Low_LB  -0.174 -0.279 -0.393 -0.521 -0.669 -0.854 -1.110 -1.554 -2.029 -2.739
          0.5%
High_UB  3.343
Low_LB  -3.360
```

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

```
              Value
High_UB_HBOP  2.021
High_UB_S1    0.522
Low_LB_B1    -0.521
Low_LB_LBOP  -2.029
High_omega    0.026
High_alpha    0.064
High_beta     0.924
Low_omega     0.036
Low_alpha     0.078
Low_beta      0.907
```