GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

20/11/2020

Input

```
symbol = "BOVA11.SA"#"BOVA11.SA"#
from=as.Date('2000-01-01')#2012
to=as.Date('2017-12-31')#'2018-12-31'
C_Trend = 0.95
C_Reaction = 0.50
```

Data download

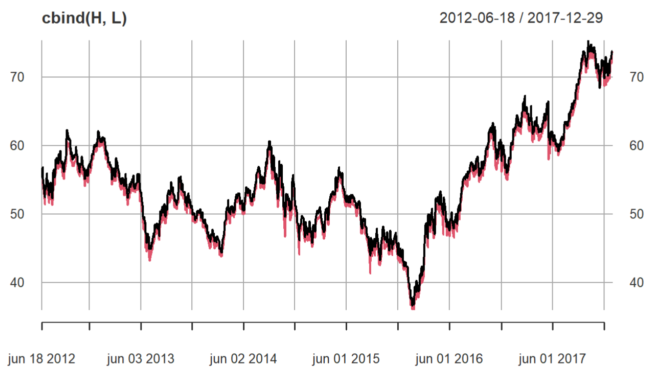
```
getSymbols.yahoo(symbol, from=from, to=to,env=globalenv())
```

```
## [1] "BOVA11.SA"
```

```
x <- get(symbol, envir=globalenv())
rm(list = symbol, envir=globalenv())
```

High and Low

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

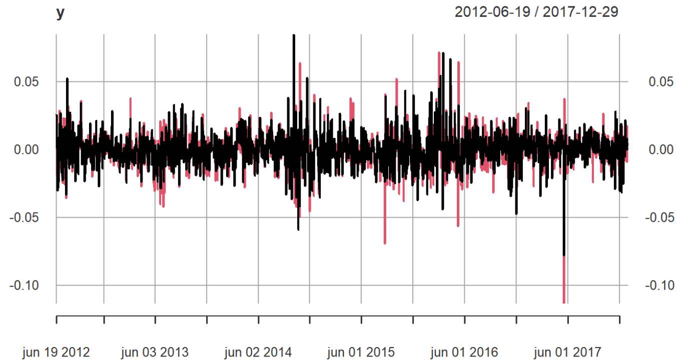


Returns

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

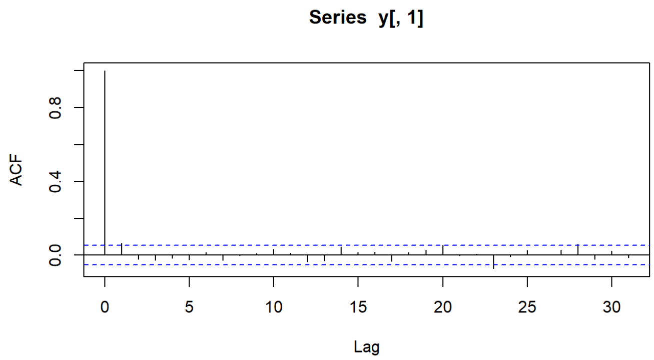
```
##                BOVA11.SA.High BOVA11.SA.Low
## BOVA11.SA.High      1.0000000     0.7361569
## BOVA11.SA.Low       0.7361569     1.0000000
```

```
plot(y)
```

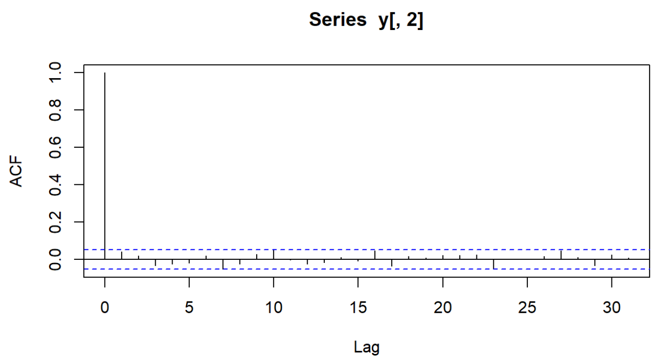


Autocorrelation

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



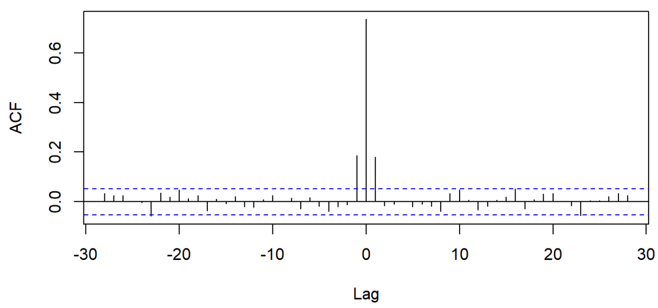

```
acf(y[,2])
```



Cross correlation

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

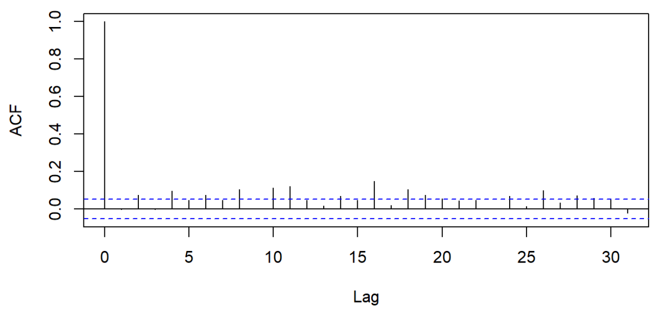
as.ts(y[, 1]) & as.ts(y[, 2])



Volatility verification

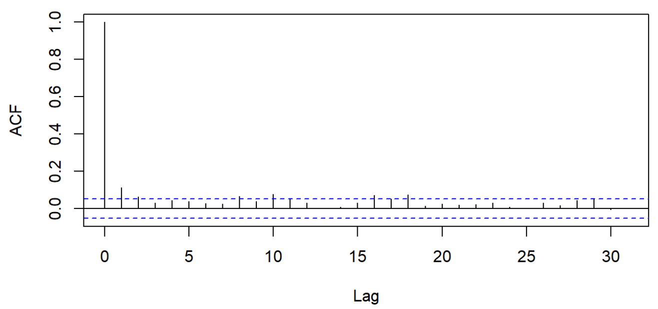
```
acf(y[,1]^2)
```

Series y[, 1]^2




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

Series 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))
```

```
## 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
##   0.50   0.22   0.24   0.21   0.22   0.21   0.23   0.20   0.19   0.26   0.26
## 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: 0.49
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

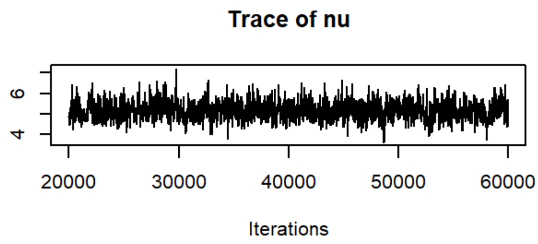
```
out2 <- increaseSim(out, nSim=50000)
```

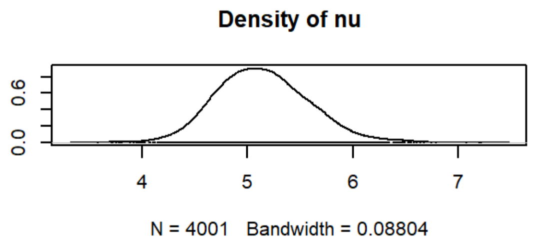
```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.5
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

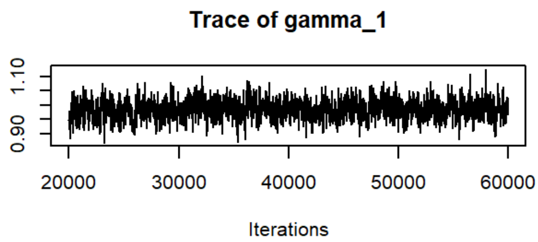
```
out <- window(out2, start=20000, thin=10)
rm(out2)

end <- Sys.time()

# elapsed time
end-start
```

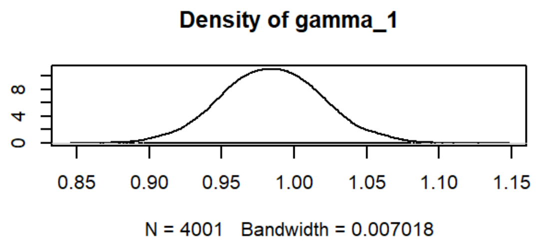
```
## Time difference of 1.323248 mins
```

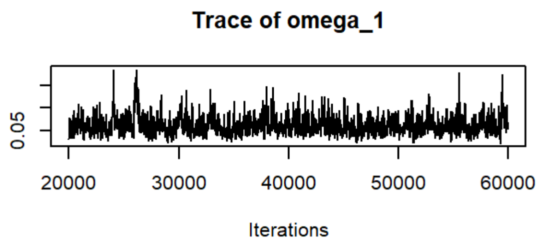
```
# plot Markov Chain
plot(out$MC)
```

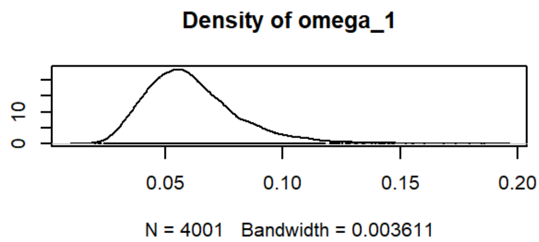

Trace of nu
4
6
20000
30000
40000
50000
60000
Iterations

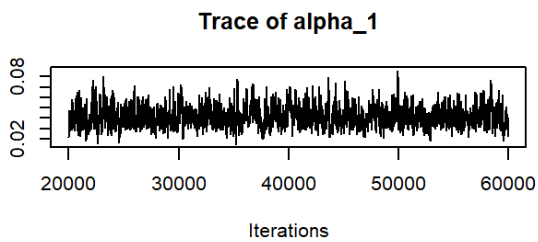

Density of nu
0.0
0.6
4
5
6
7
N = 4001 Bandwidth = 0.08804

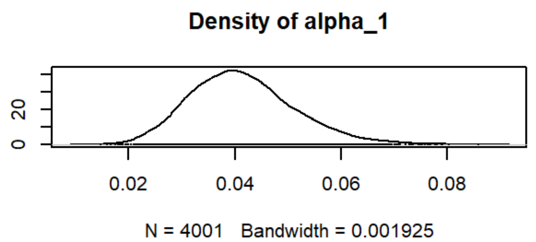

Trace of gamma_1
0.90
1.10
20000
30000
40000
50000
60000
Iterations

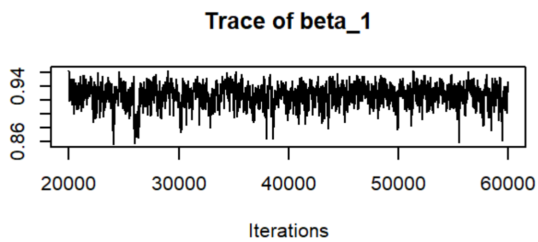

Density of gamma_1
0
4
8
0.85
0.90
0.95
1.00
1.05
1.10
1.15
N = 4001 Bandwidth = 0.007018

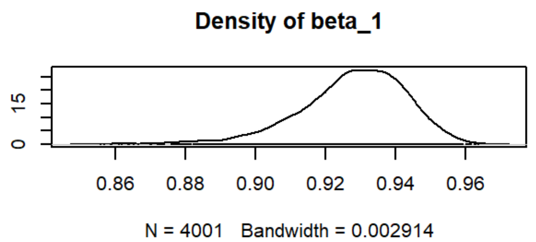

Trace of omega_1
0.05
20000
30000
40000
50000
60000
Iterations

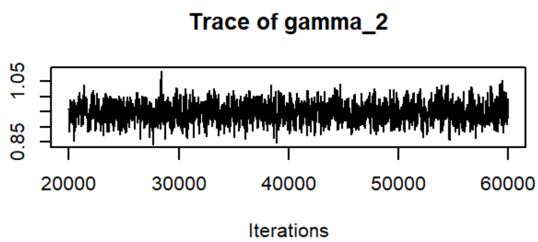

Density of omega_1
0
10
0.05
0.10
0.15
0.20
N = 4001 Bandwidth = 0.003611

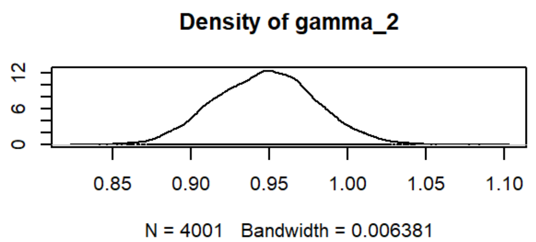

Trace of alpha_1
0.02
0.08
20000
30000
40000
50000
60000
Iterations

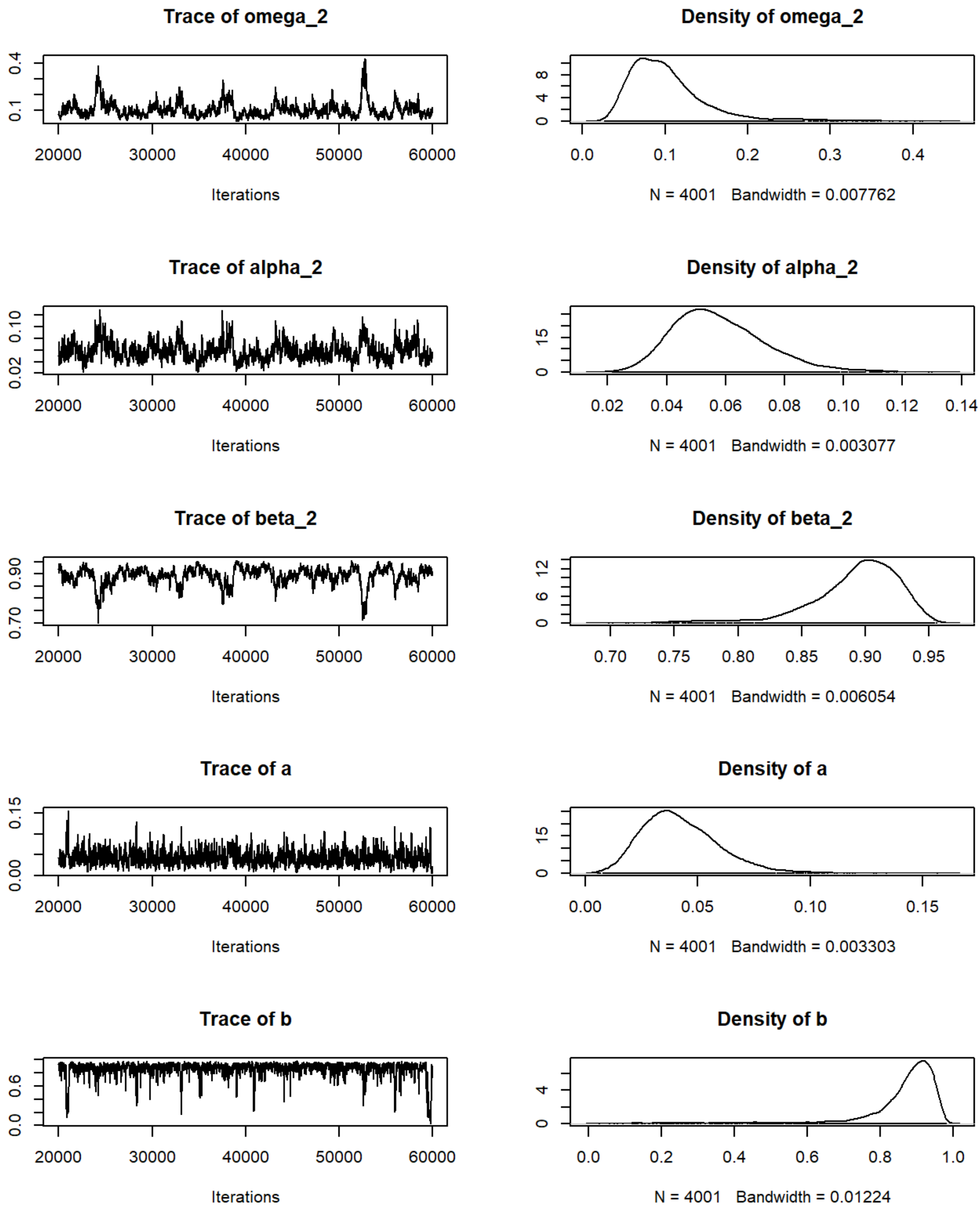

Density of alpha_1
0
20
0.02
0.04
0.06
0.08
N = 4001 Bandwidth = 0.001925


Trace of beta_1
0.86
0.94
20000
30000
40000
50000
60000
Iterations


Density of beta_1
0
15
0.86
0.88
0.90
0.92
0.94
0.96
N = 4001 Bandwidth = 0.002914


Trace of gamma_2
0.85
1.05
20000
30000
40000
50000
60000
Iterations


Density of gamma_2
0
6
12
0.85
0.90
0.95
1.00
1.05
1.10
N = 4001 Bandwidth = 0.006381




```
## 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      5.14565 0.440214 0.0069595      0.0167196
## gamma_1 0.98444 0.034781 0.0005499      0.0013587
## omega_1 0.06135 0.019976 0.0003158      0.0010104
## alpha_1 0.04144 0.009796 0.0001549      0.0003837
## beta_1  0.92776 0.015273 0.0002415      0.0007383
## gamma_2 0.94772 0.031625 0.0005000      0.0010181
## omega_2 0.10305 0.050473 0.0007980      0.0071429
## alpha_2 0.05788 0.015655 0.0002475      0.0014066
## beta_2  0.89267 0.035766 0.0005654      0.0050694
## a       0.04243 0.017082 0.0002701      0.0006992
## b       0.85781 0.123557 0.0019534      0.0099809
##
## 2. Quantiles for each variable:
##
##            2.5%     25%     50%     75%   97.5%
## nu      4.35559 4.84057 5.12182 5.42523 6.07329
## gamma_1 0.91600 0.96074 0.98406 1.00771 1.05483
## omega_1 0.03196 0.04740 0.05836 0.07139 0.10956
## alpha_1 0.02441 0.03449 0.04058 0.04727 0.06308
## beta_1  0.89324 0.91916 0.92921 0.93851 0.95261
## gamma_2 0.88679 0.92536 0.94822 0.96876 1.00985
## omega_2 0.04264 0.06943 0.09297 0.12097 0.24823
## alpha_2 0.03257 0.04666 0.05577 0.06710 0.09441
## beta_2  0.79191 0.87677 0.89892 0.91697 0.94247
## a       0.01592 0.03025 0.04016 0.05219 0.08037
## b       0.44934 0.84228 0.89042 0.92357 0.95918
```

```
# Prepare input for the expert advisor
parEst <- summary(out)$statistics[,'Mean']

## 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'])


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(round(m,3))
```

```
##               Value
## High_UB_HBOP  1.975
## High_UB_S1    0.570
## Low_LB_B1    -0.555
## Low_LB_LBOP  -2.053
## High_omega    0.061
## High_alpha    0.041
## High_beta     0.928
## Low_omega     0.103
## Low_alpha     0.058
## Low_beta      0.893
```