GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

20/11/2020

Input

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

Data download

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

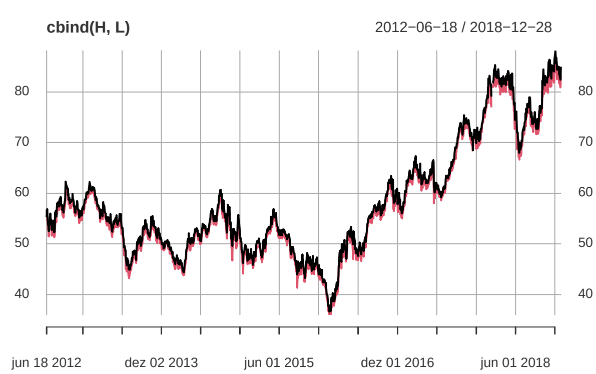
```
## Warning: BOVA11.SA contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## [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))
```

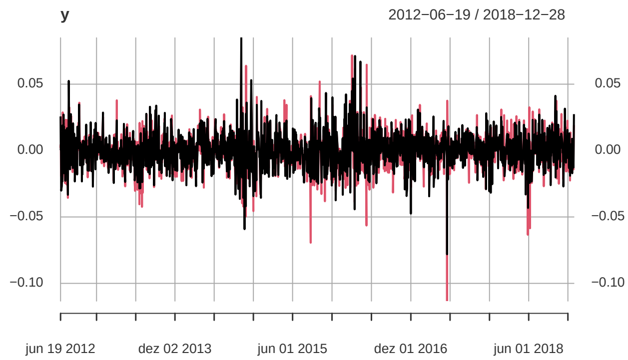


Returns

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

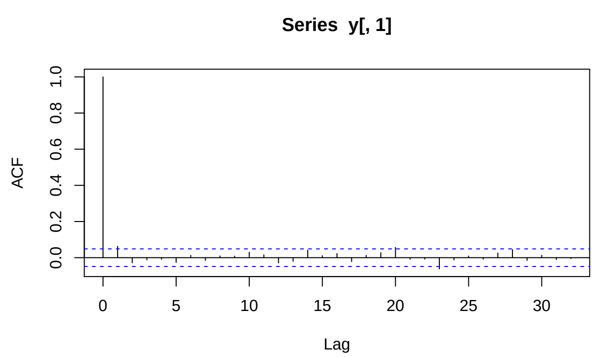
```
##                BOVA11.SA.High BOVA11.SA.Low
## BOVA11.SA.High      1.0000000     0.7256971
## BOVA11.SA.Low       0.7256971     1.0000000
```

```r
plot(y)
```

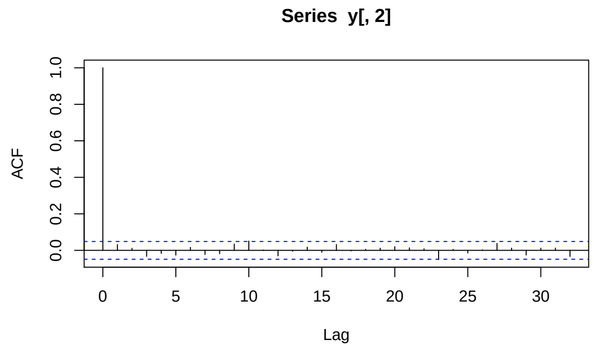


Autocorrelation

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

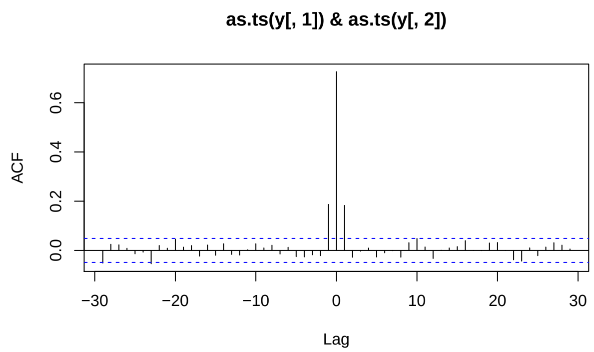



```r
acf(y[,2])
```



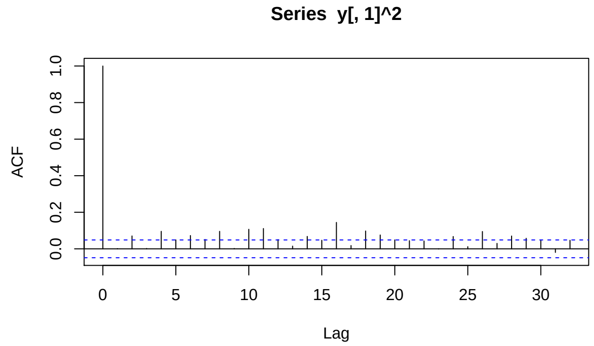
Cross correlation

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

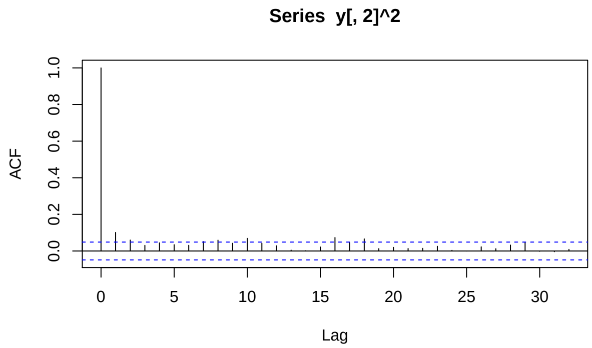


Volatility verification

```
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×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
## 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
##   0.45   0.20   0.19   0.22   0.22   0.19   0.18   0.22   0.19   0.29   0.31
## 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: 0.51
## lambda: 0.48
## Accept Rate: 0.42
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

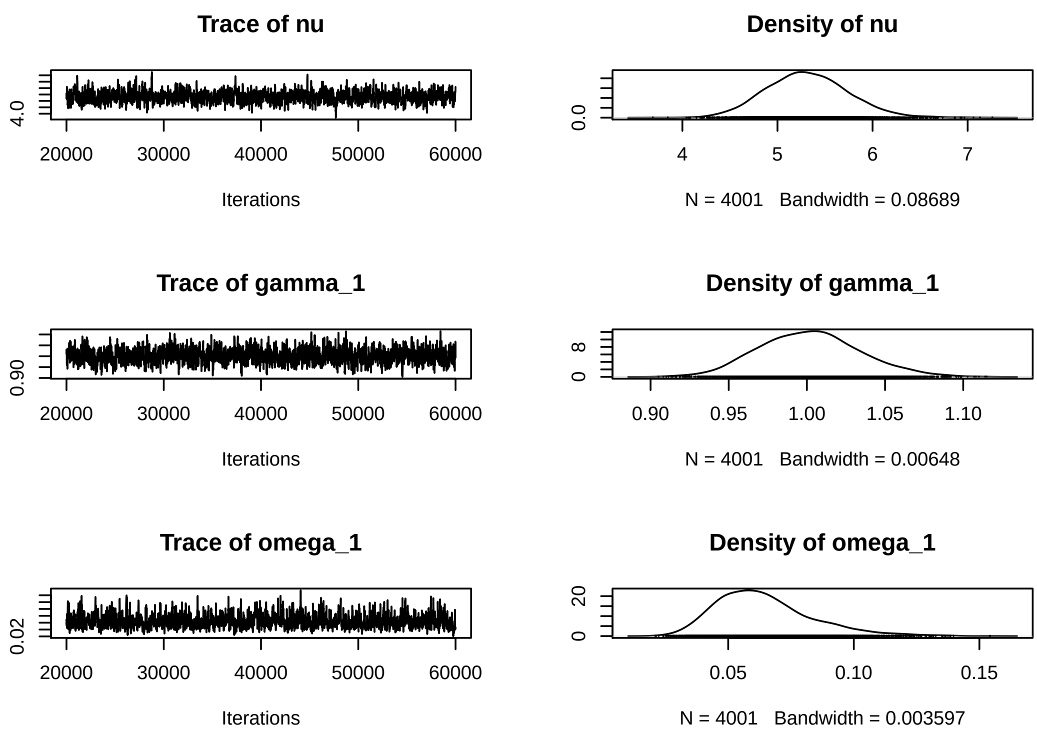
```
out2 <- increaseSim(out, nSim=50000)
```

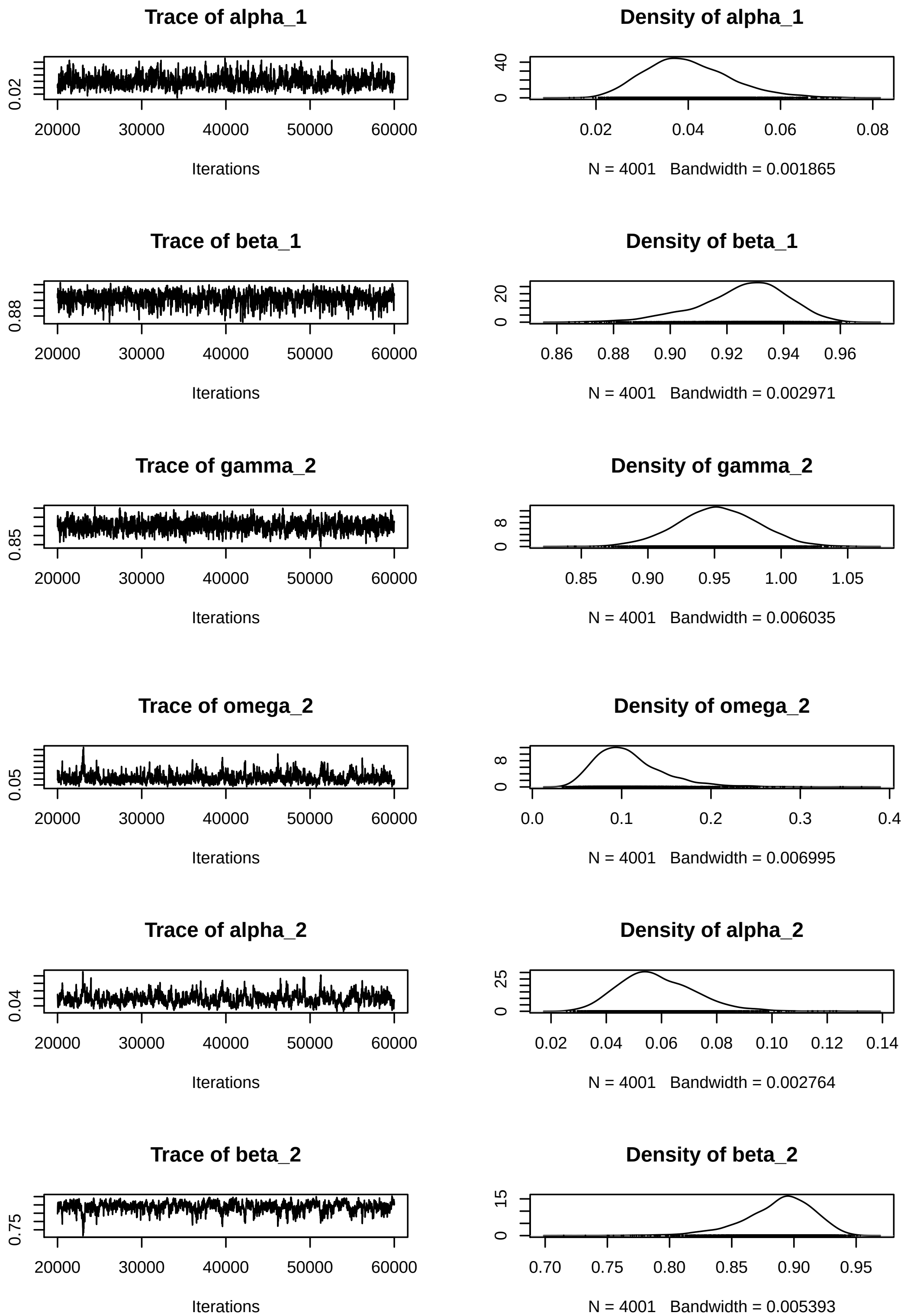
```
## Calibrating the Lambda coefficient:
## lambda: 0.48
## Accept Rate: 0.43
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

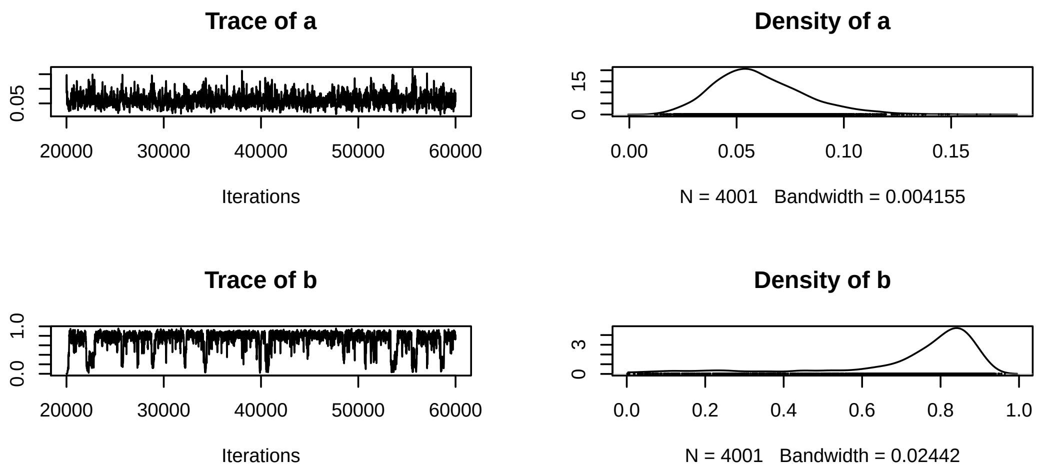
```r
out <- window(out2, start=20000, thin=10)
rm(out2)

end <- Sys.time()

# elapsed time
end-start
```

```
## Time difference of 1.225343 mins
```

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




Trace of alpha_1
0.02
20000 30000 40000 50000 60000
Iterations
Density of alpha_1
0 40
0.02 0.04 0.06 0.08
N = 4001 Bandwidth = 0.001865
Trace of beta_1
0.88
20000 30000 40000 50000 60000
Iterations
Density of beta_1
0 20
0.86 0.88 0.90 0.92 0.94 0.96
N = 4001 Bandwidth = 0.002971
Trace of gamma_2
0.85
20000 30000 40000 50000 60000
Iterations
Density of gamma_2
0 8
0.85 0.90 0.95 1.00 1.05
N = 4001 Bandwidth = 0.006035
Trace of omega_2
0.05
20000 30000 40000 50000 60000
Iterations
Density of omega_2
0 8
0.0 0.1 0.2 0.3 0.4
N = 4001 Bandwidth = 0.006995
Trace of alpha_2
0.04
20000 30000 40000 50000 60000
Iterations
Density of alpha_2
0 25
0.02 0.04 0.06 0.08 0.10 0.12 0.14
N = 4001 Bandwidth = 0.002764
Trace of beta_2
0.75
20000 30000 40000 50000 60000
Iterations
Density of beta_2
0 15
0.70 0.75 0.80 0.85 0.90 0.95
N = 4001 Bandwidth = 0.005393




```r
## 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      5.31448 0.43224 0.0068335      0.0151886
## gamma_1 1.00193 0.03216 0.0005084      0.0010361
## omega_1 0.06417 0.01909 0.0003018      0.0007214
## alpha_1 0.03994 0.00925 0.0001462      0.0003580
## beta_1  0.92581 0.01569 0.0002480      0.0006159
## gamma_2 0.95273 0.03027 0.0004785      0.0010810
## omega_2 0.10857 0.03917 0.0006192      0.0021234
## alpha_2 0.05849 0.01386 0.0002191      0.0009409
## beta_2  0.88750 0.02891 0.0004571      0.0018473
## a       0.06086 0.02145 0.0003391      0.0007736
## b       0.71769 0.21151 0.0033439      0.0186528
##
## 2. Quantiles for each variable:
##
##            2.5%     25%     50%     75%   97.5%
## nu      4.51976 5.01750 5.29704 5.59455 6.20628
```

```
## gamma_1 0.94208 0.97950 1.00132 1.02254 1.06815
## omega_1 0.03449 0.05042 0.06145 0.07431 0.10996
## alpha_1 0.02441 0.03339 0.03904 0.04577 0.06051
## beta_1  0.89024 0.91689 0.92771 0.93663 0.95219
## gamma_2 0.89250 0.93289 0.95263 0.97297 1.01151
## omega_2 0.05246 0.08122 0.10206 0.12767 0.20416
## alpha_2 0.03541 0.04873 0.05688 0.06708 0.08911
## beta_2  0.81963 0.87178 0.89160 0.90759 0.93268
## a       0.02574 0.04584 0.05795 0.07344 0.11026
## b       0.10817 0.68913 0.79811 0.85128 0.91185
```

```r
# 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['gamma_1'])
#S1
High_UB_S1 = qsstd(p=1-(1-C_Reaction)/2, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['gamma_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  1.997
## High_UB_S1    0.571
## Low_LB_B1    -0.561
## Low_LB_LBOP  -2.048
```

```
## High_omega    0.064
## High_alpha    0.040
## High_beta     0.926
## Low_omega     0.109
## Low_alpha     0.058
## Low_beta      0.888
```