GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

20/11/2020

Input

```
symbol = "BA"#"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] "BA"
```

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

High and Low

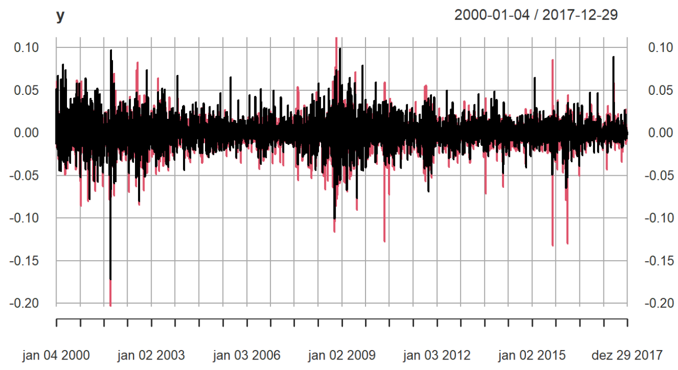
```
H <- Hi(x)
L <- Lo(x)
plot(cbind(H,L))
```

Returns

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

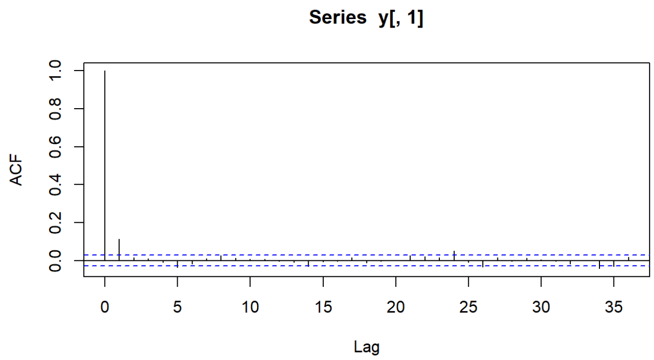
```
##            BA.High    BA.Low
## BA.High 1.0000000 0.7220466
## BA.Low  0.7220466 1.0000000
```

```
plot(y)
```

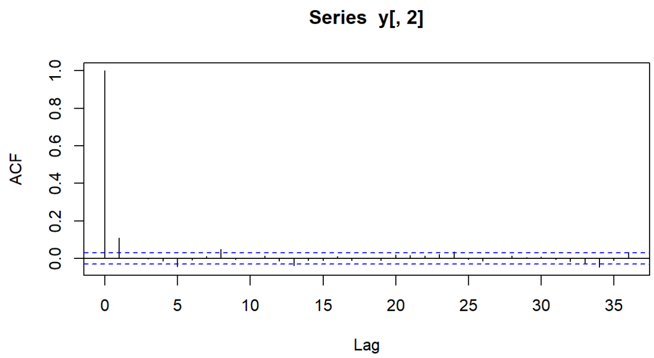


Autocorrelation

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

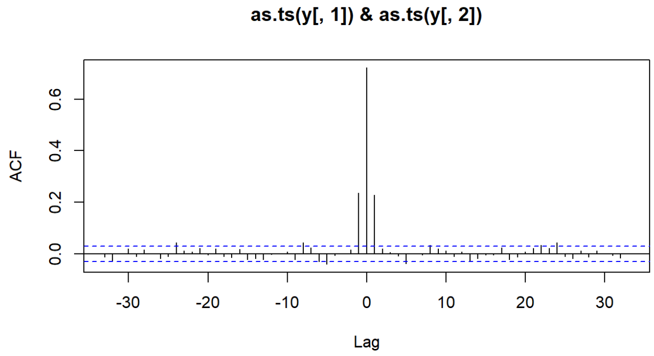



```
acf(y[,2])
```



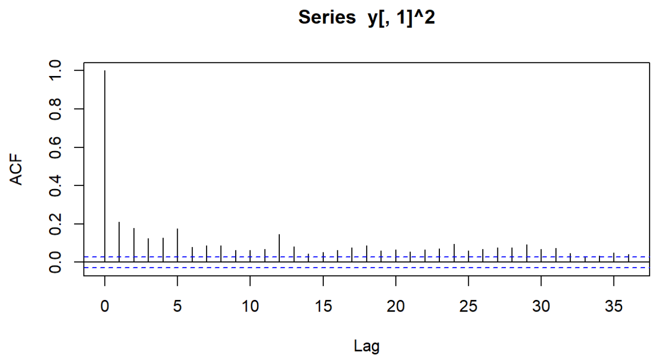
Cross correlation

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

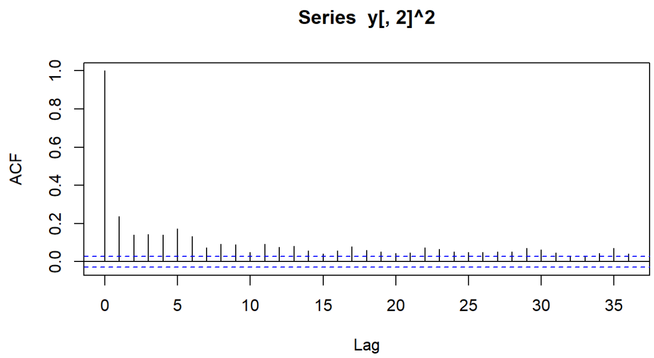


Volatility verification

```
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))
```

```
## 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   0.21   0.08   0.11   0.11   0.16   0.10   0.13   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
##   0.31   0.19   0.17   0.13   0.19   0.17   0.18   0.18   0.23   0.32   0.60
## 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   0.16   0.20   0.19   0.19   0.17   0.14   0.18   0.20   0.31   0.48
## 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   0.19   0.18   0.19   0.28   0.16   0.20   0.34   0.51
## 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   0.16   0.21   0.23   0.17   0.18   0.25   0.15   0.21   0.33   0.38
## 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: 0.47
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

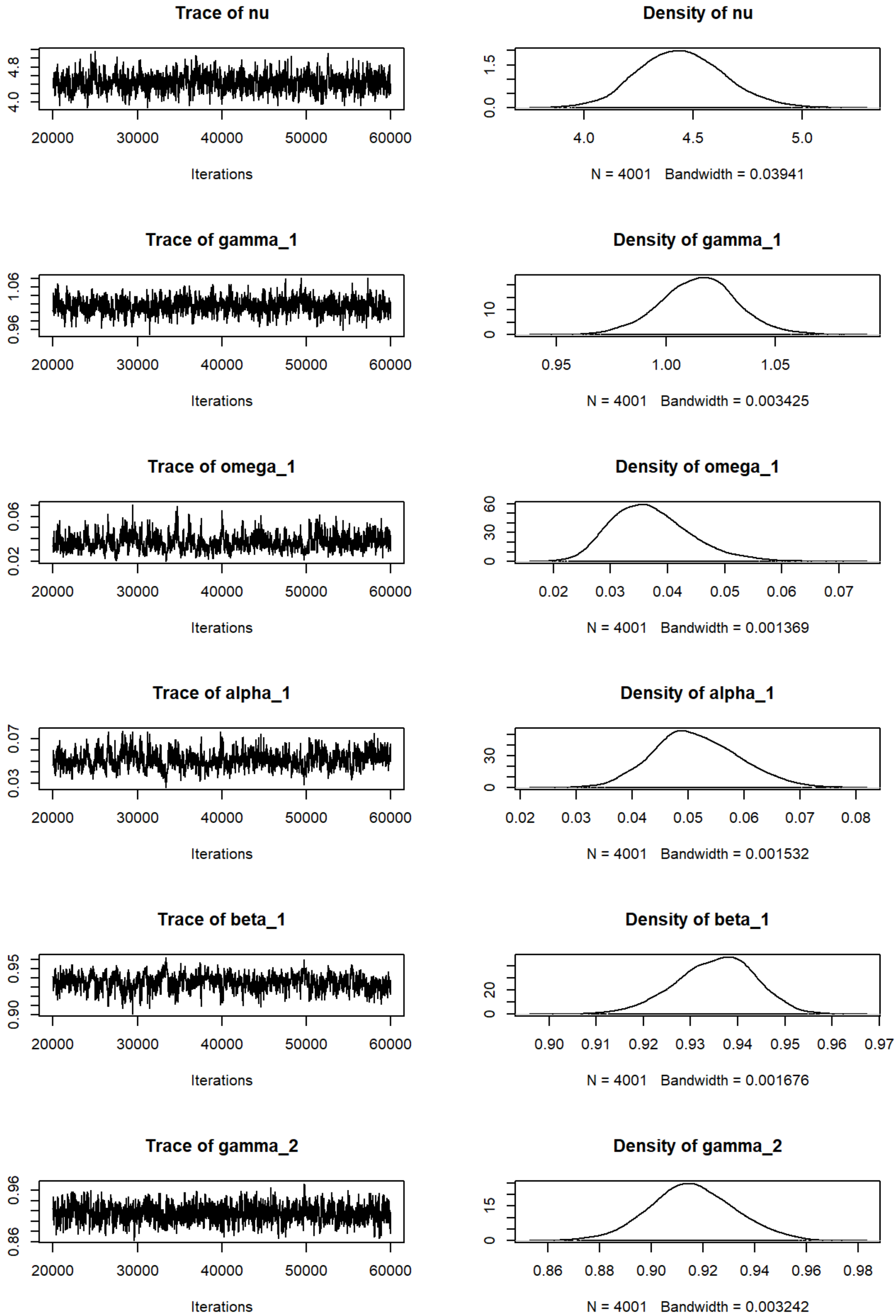
```
out2 <- increaseSim(out, nSim=50000)
```

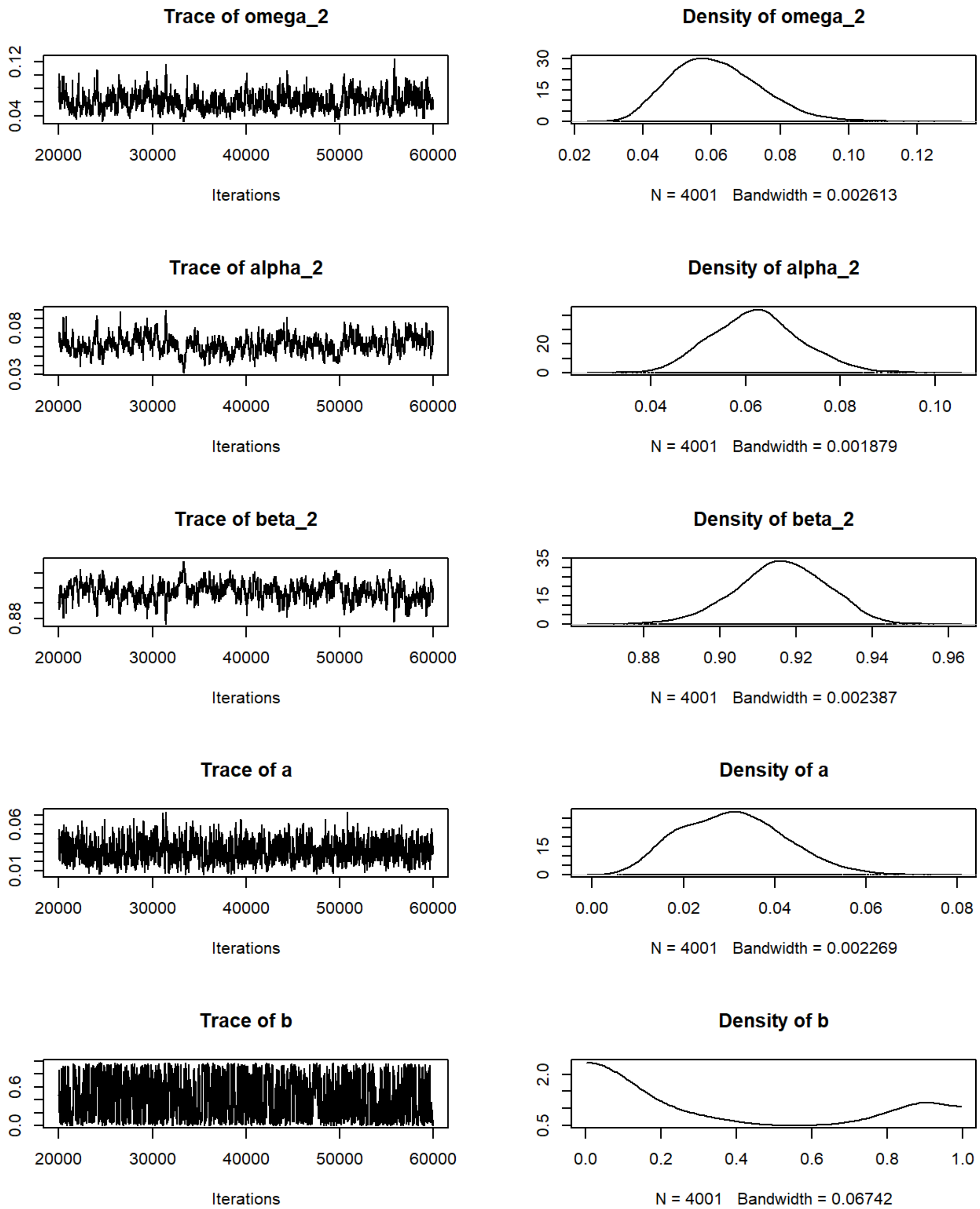
```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.44
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

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

end <- Sys.time()

# elapsed time
end-start
```

```
## Time difference of 6.247188 mins
```

```
# 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      4.44468 0.195314 0.0030878      0.0082069
## gamma_1 1.01506 0.017387 0.0002749      0.0007111
## omega_1 0.03705 0.006891 0.0001089      0.0004151
## alpha_1 0.05147 0.007591 0.0001200      0.0004852
## beta_1  0.93522 0.008307 0.0001313      0.0005418
## gamma_2 0.91580 0.016246 0.0002568      0.0006058
## omega_2 0.06134 0.012952 0.0002048      0.0008593
## alpha_2 0.06229 0.009482 0.0001499      0.0007939
## beta_2  0.91614 0.011842 0.0001872      0.0009573
## a       0.03074 0.011247 0.0001778      0.0004417
## b       0.42067 0.334145 0.0052826      0.0145347
##
## 2. Quantiles for each variable:
##
##            2.5%     25%     50%     75%   97.5%
## nu      4.07870 4.30852 4.43848 4.57485 4.84595
## gamma_1 0.97986 1.00359 1.01531 1.02633 1.05001
## omega_1 0.02570 0.03208 0.03642 0.04117 0.05262
## alpha_1 0.03771 0.04624 0.05096 0.05653 0.06701
## beta_1  0.91778 0.92979 0.93587 0.94104 0.95008
## gamma_2 0.88387 0.90502 0.91529 0.92655 0.94873
## omega_2 0.03995 0.05197 0.06009 0.06965 0.08912
## alpha_2 0.04490 0.05575 0.06208 0.06822 0.08197
## beta_2  0.89179 0.90848 0.91633 0.92433 0.93785
## a       0.01132 0.02216 0.03048 0.03838 0.05349
## b       0.01063 0.10529 0.32601 0.77331 0.95420
```

```
# 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  1.998
## High_UB_S1    0.541
## Low_LB_B1    -0.524
## Low_LB_LBOP  -2.081
## High_omega    0.037
## High_alpha    0.051
## High_beta     0.935
## Low_omega     0.061
## Low_alpha     0.062
## Low_beta      0.916
```