GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

13/02/2021

Input

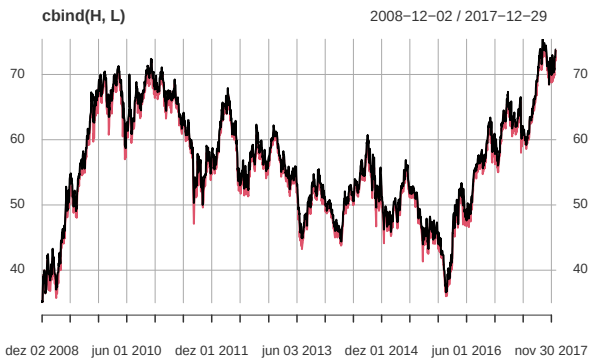
```
symbol = "BOVA11.SA"
from=as.Date('2000-01-01')
to=as.Date('2017-12-31')
C_Trend = 0.95
C_Reaction = 0.50
```

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))
```



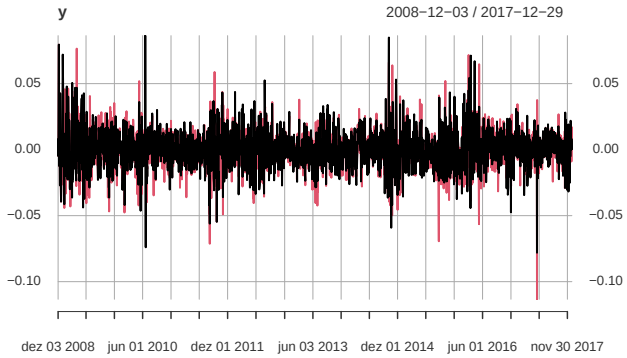
Returns

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

```
##                 BOVA11.SA.High BOVA11.SA.Low
## BOVA11.SA.High       1.0000000     0.7085079
```

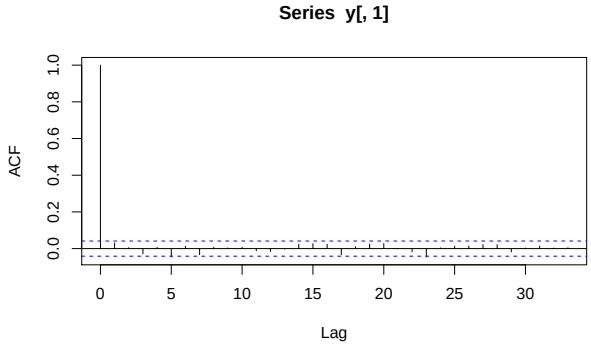
```
## BOVA11.SA.Low       0.7085079      1.0000000
```

```
plot(y)
```

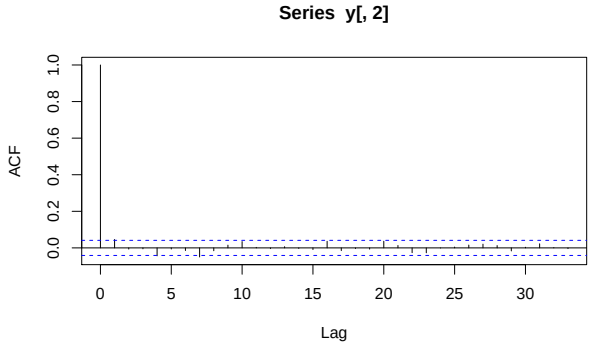


Autocorrelation

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

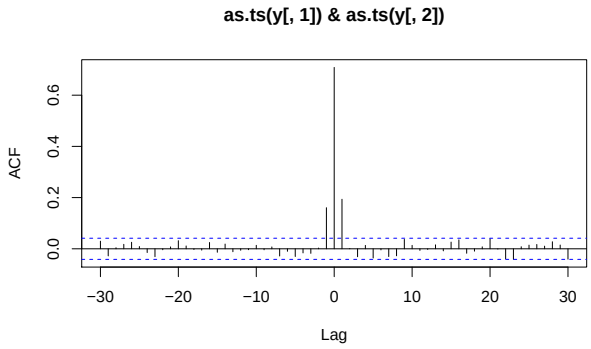



```
acf(y[,2])
```



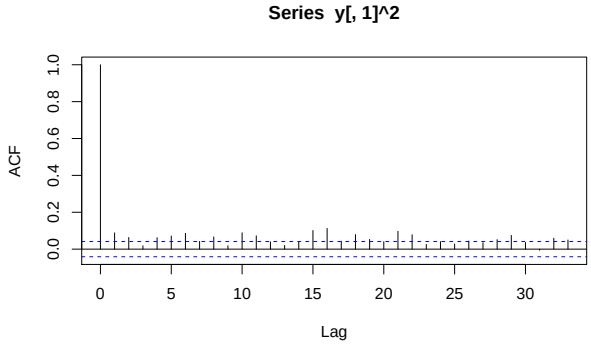
Cross correlation

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

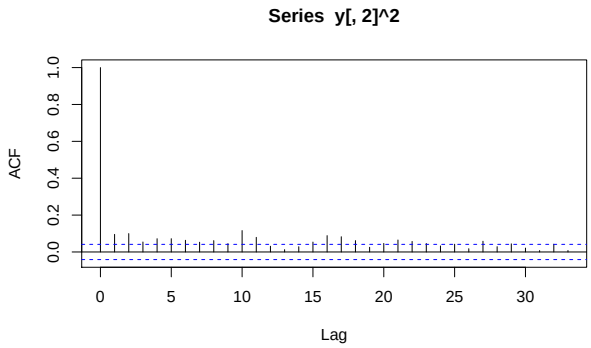


Volatility verification

```
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
```

```r
start <- Sys.time()

out <- bayesDccGarch(mY, control=list(print=FALSE, nPilotSim=3000))
```

```
## Maximizing the log-posterior density function.
## Done.
```

```
## Warning in if (class(control$cholCov) != "try-error") {: a condição tem
## comprimento > 1 e somente o primeiro elemento será usado
```

```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.35
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

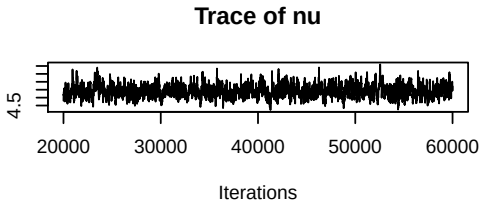
```r
out2 <- increaseSim(out, nSim=50000)
```

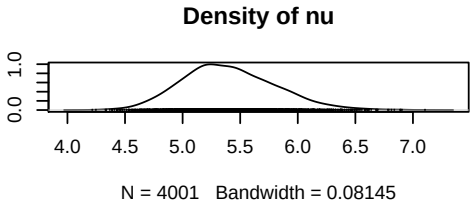
```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.37
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

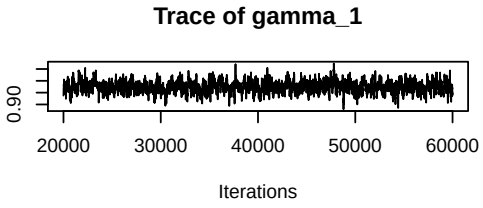
```r
out <- window(out2, start=20000, thin=10)
rm(out2)

end <- Sys.time()

# elapsed time
end-start
```

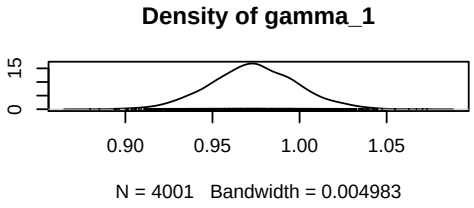
```
## Time difference of 1.453167 mins
```

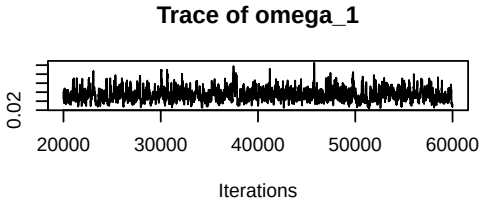
```r
## Estimative of parameters
parEst <- summary(out)$statistics[,'Mean']


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

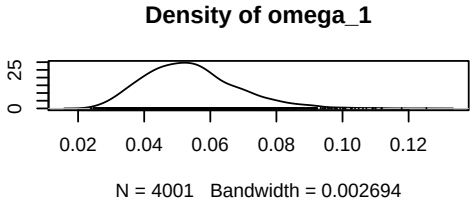

Trace of nu
4.5
20000
30000
40000
50000
60000
Iterations

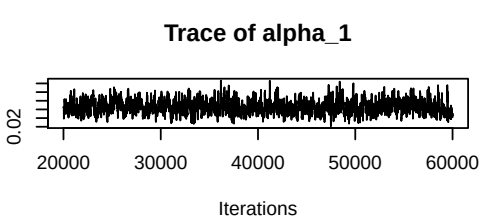

Density of nu
0.0
1.0
4.0
4.5
5.0
5.5
6.0
6.5
7.0
N = 4001 Bandwidth = 0.08145

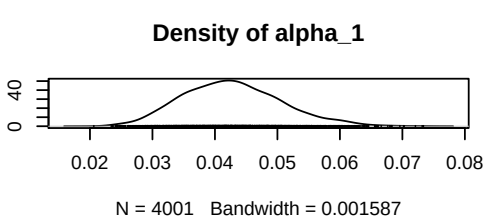

Trace of gamma_1
0.90
20000
30000
40000
50000
60000
Iterations

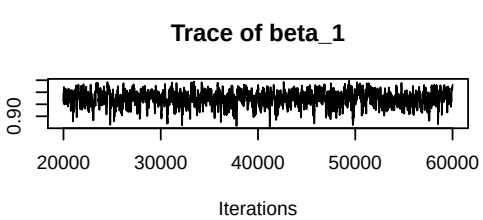

Density of gamma_1
0
15
0.90
0.95
1.00
1.05
N = 4001 Bandwidth = 0.004983

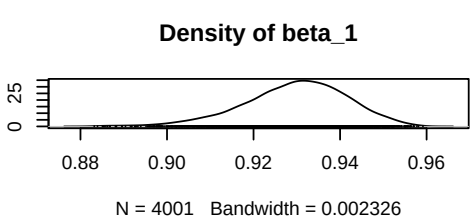

Trace of omega_1
0.02
20000
30000
40000
50000
60000
Iterations

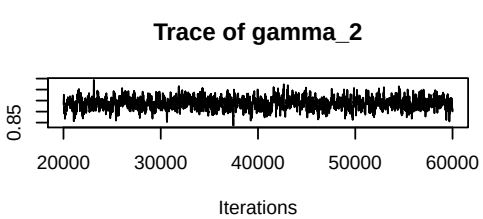

Density of omega_1
0
25
0.02
0.04
0.06
0.08
0.10
0.12
N = 4001 Bandwidth = 0.002694

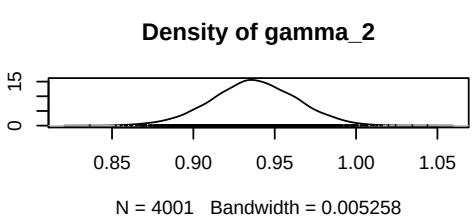

Trace of alpha_1
0.02
20000
30000
40000
50000
60000
Iterations

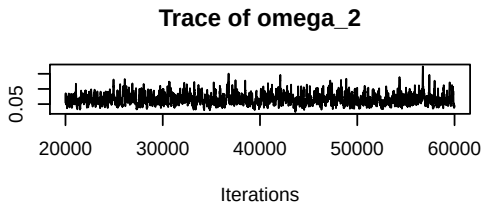

Density of alpha_1
0
40
0.02
0.03
0.04
0.05
0.06
0.07
0.08
N = 4001 Bandwidth = 0.001587

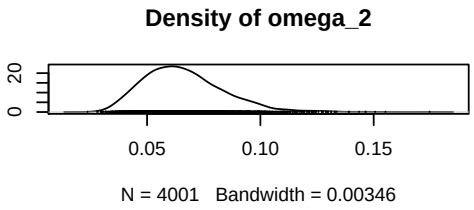

Trace of beta_1
0.90
20000
30000
40000
50000
60000
Iterations

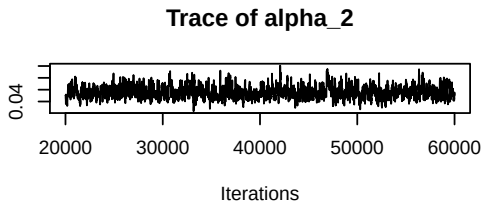

Density of beta_1
0
25
0.88
0.90
0.92
0.94
0.96
N = 4001 Bandwidth = 0.002326

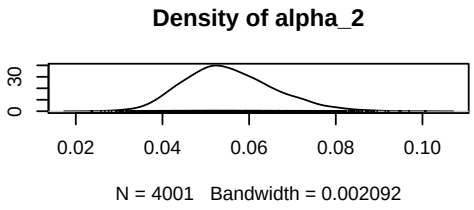

Trace of gamma_2
0.85
20000
30000
40000
50000
60000
Iterations

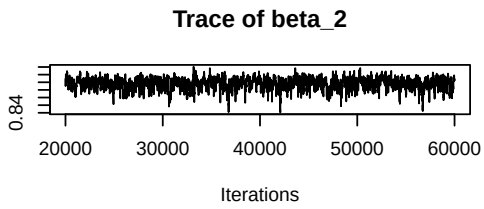

Density of gamma_2
0
15
0.85
0.90
0.95
1.00
1.05
N = 4001 Bandwidth = 0.005258

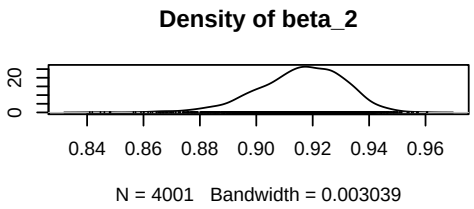


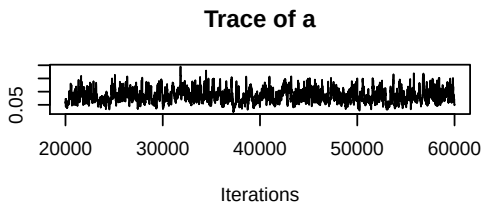


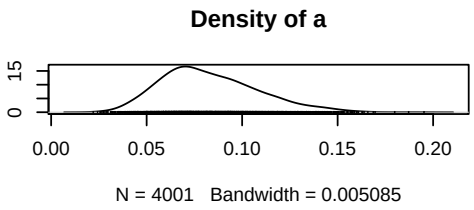


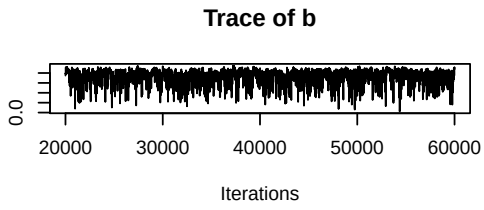


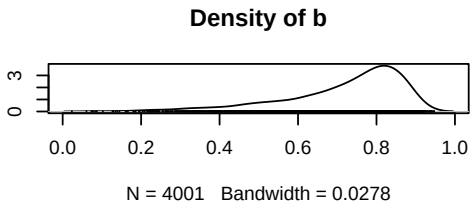













```
## Markov Chain convergence
out$MC %>% geweke.diag()
```

```
##
```

```
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##       nu  gamma_1  omega_1  alpha_1   beta_1  gamma_2  omega_2  alpha_2
##  0.66276  0.95051 -1.06946 -0.55557  0.95487 -0.58825 -0.98527  0.01639
##   beta_2        a        b
##  0.43885 -0.91982  1.13178
```

```
## Model adequability / residual hypothesis verification
Vol <- xts(out$H[,c("H_1,1","H_2,2")], order.by=index(y)) # estimated volatility
r <- mY / sqrt(Vol) # Standard Residuals

par(mfrow=c(3,2))

plot(r[,1], main="e_H")
plot(r[,2], main="e_L")

# Volatility test
acf(r[,1]^2, main="e_H^2")
acf(r[,2]^2, main="e_L^2")
Box.test(r[,1]^2)
```

```
##
##  Box-Pierce test
##
## data:  r[, 1]^2
## X-squared = 0.15303, df = 1, p-value = 0.6957
```

```
Box.test(r[,2]^2)
```

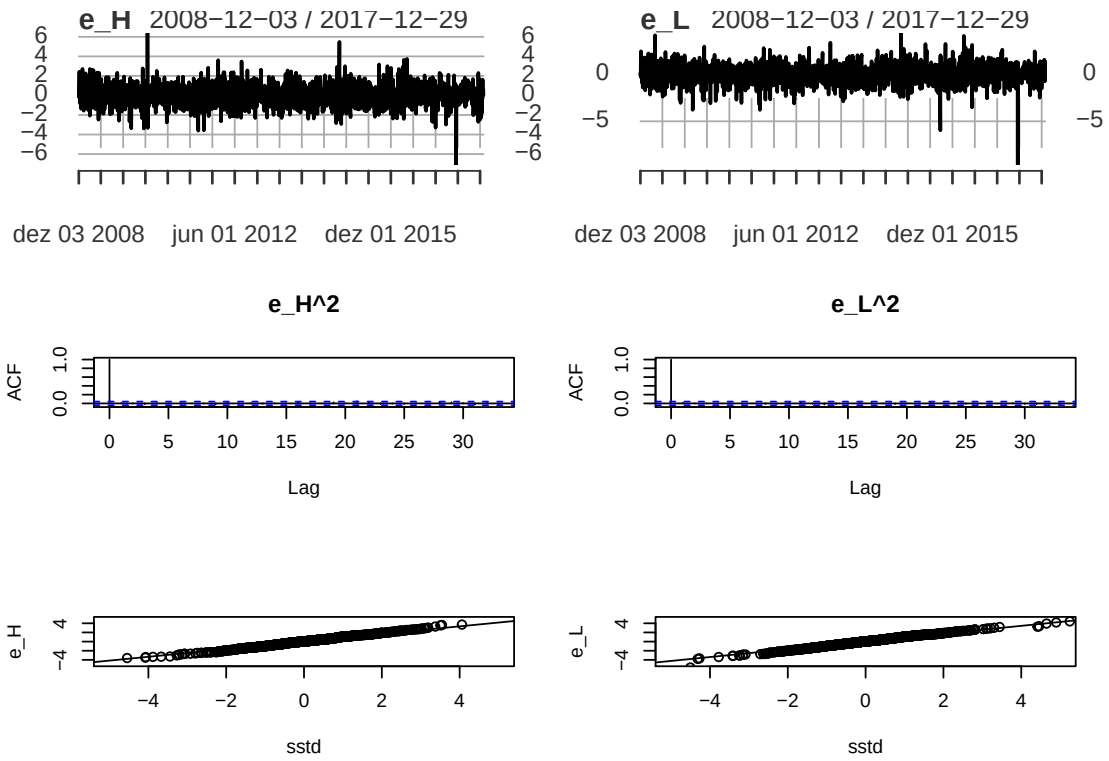
```
##
##  Box-Pierce test
##
## data:  r[, 2]^2
## X-squared = 0.38652, df = 1, p-value = 0.5341
```

```
# adequability
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)

ks.test(r1 , x)
```

```
## Warning in ks.test(r1, x): p-value will be approximate in the presence of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  r1 and x
## D = 0.035313, p-value = 0.1424
## alternative hypothesis: two-sided
```

```
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)
```




```
ks.test(r2 , x)
```

```
## Warning in ks.test(r2, x): p-value will be approximate in the presence of ties
```

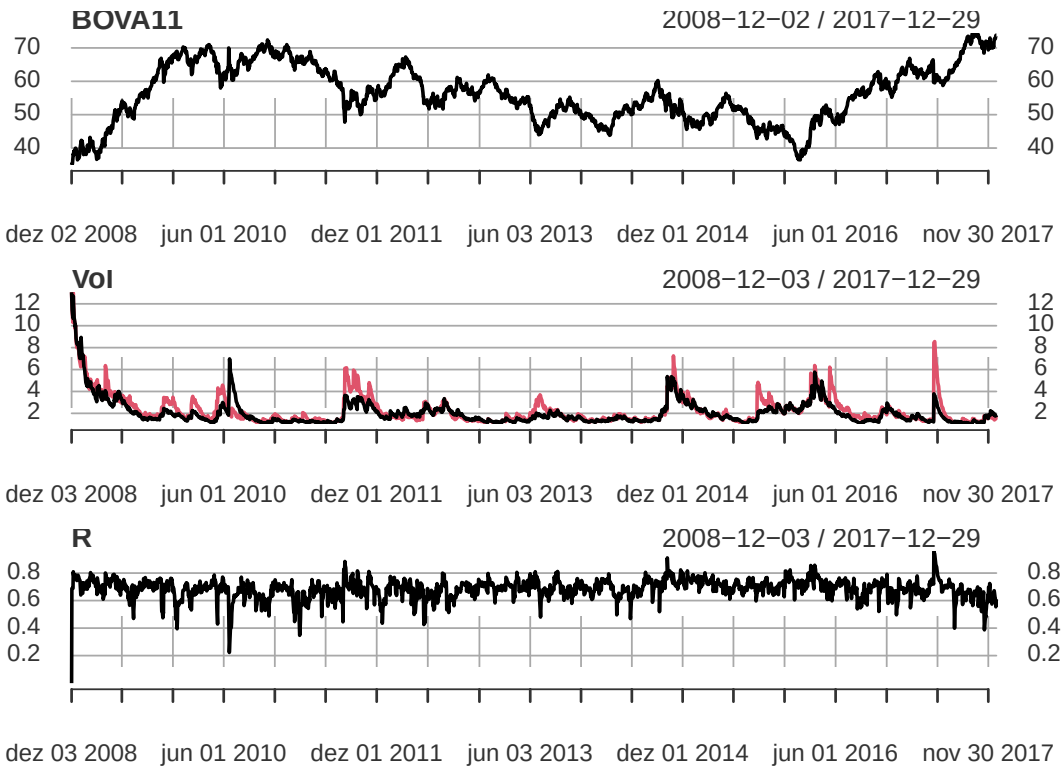
```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  r2 and x
## D = 0.02323, p-value = 0.617
## alternative hypothesis: two-sided
```

```
## 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      5.40439 0.403682 0.0063820      0.0173865
## gamma_1 0.97378 0.024991 0.0003951      0.0010712
## omega_1 0.05410 0.014011 0.0002215      0.0006568
## alpha_1 0.04275 0.007864 0.0001243      0.0003122
## beta_1  0.92969 0.011954 0.0001890      0.0005318
## gamma_2 0.93726 0.026225 0.0004146      0.0011520
```

```
## omega_2 0.06615 0.017711 0.0002800      0.0007377
## alpha_2 0.05544 0.010424 0.0001648      0.0004405
## beta_2  0.91581 0.015446 0.0002442      0.0007026
## a       0.08303 0.025202 0.0003984      0.0012115
## b       0.71678 0.154192 0.0024377      0.0058450
##
## 2. Quantiles for each variable:
##
##              2.5%     25%     50%     75%   97.5%
## nu      4.69180 5.12251 5.37491 5.66817 6.26880
## gamma_1 0.92499 0.95726 0.97365 0.99035 1.02467
## omega_1 0.03122 0.04415 0.05274 0.06204 0.08585
## alpha_1 0.02896 0.03710 0.04231 0.04786 0.05982
## beta_1  0.90342 0.92256 0.93052 0.93801 0.95081
## gamma_2 0.88482 0.91987 0.93709 0.95479 0.98835
## omega_2 0.03797 0.05335 0.06399 0.07632 0.10532
## alpha_2 0.03743 0.04808 0.05451 0.06197 0.07806
## beta_2  0.88203 0.90650 0.91708 0.92668 0.94183
## a       0.04220 0.06464 0.07986 0.09918 0.13992
## b       0.31689 0.64358 0.76210 0.82818 0.90143
```

```r
## 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")
```




```
# 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['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'])

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  0.222  0.332  0.450  0.579  0.727  0.908  1.151  1.555  1.965  2.541
Low_LB  -0.185 -0.300 -0.423 -0.560 -0.718 -0.913 -1.176 -1.616 -2.066 -2.702
          0.5%
High_UB  3.018
Low_LB  -3.228
```

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

```
              Value
High_UB_HBOP  1.965
High_UB_S1    0.579
Low_LB_B1    -0.560
Low_LB_LBOP  -2.066
High_omega    0.054
High_alpha    0.043
High_beta     0.930
Low_omega     0.066
Low_alpha     0.055
Low_beta      0.916
```