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
```

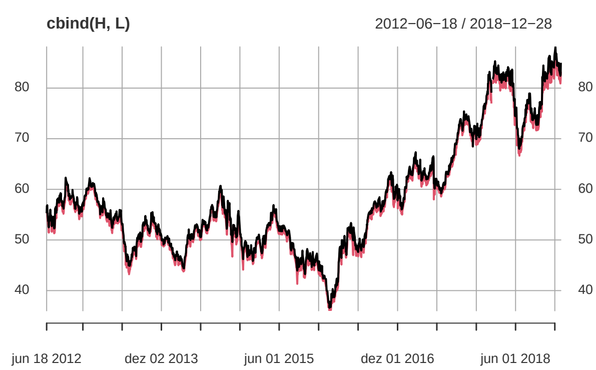
Data download

```
x <- getSymbols.yahoo(symbol,auto.assign = FALSE, from=from, to=to)
```

```
## 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.
```

High and Low

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



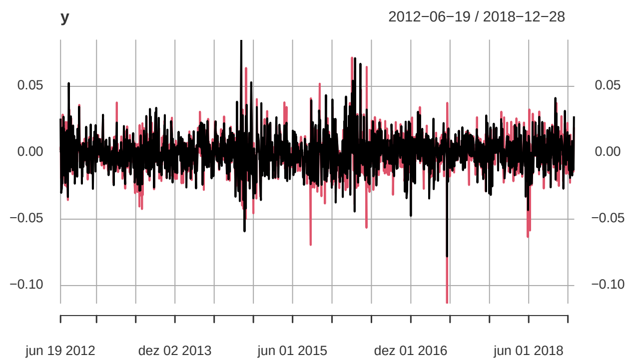
Returns

```
y <- cbind( diff(log(H)),  diff(log(L)) )
y <- na.omit(y)
```

```r
y %>% cor() # Returns correlation
```

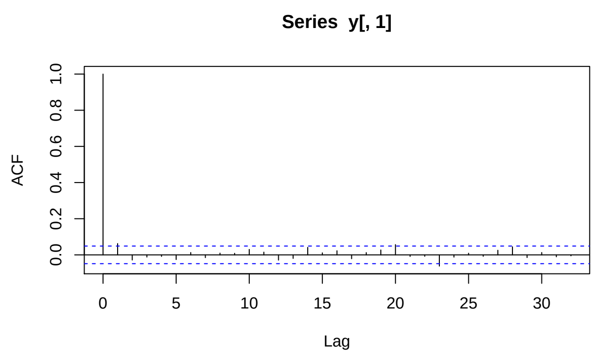
```
##                BOVA11.SA.High BOVA11.SA.Low
## BOVA11.SA.High      1.0000000     0.7256971
## BOVA11.SA.Low       0.7256971     1.0000000
```

```r
plot(y)
```

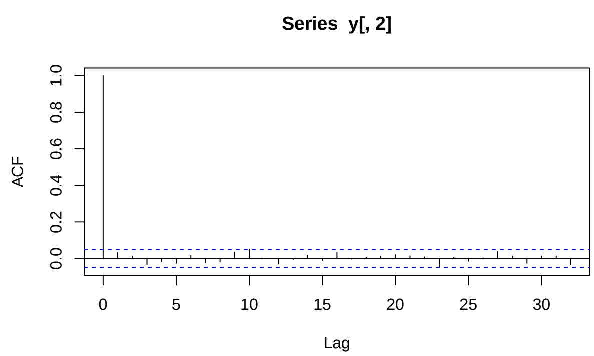


Autocorrelation

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

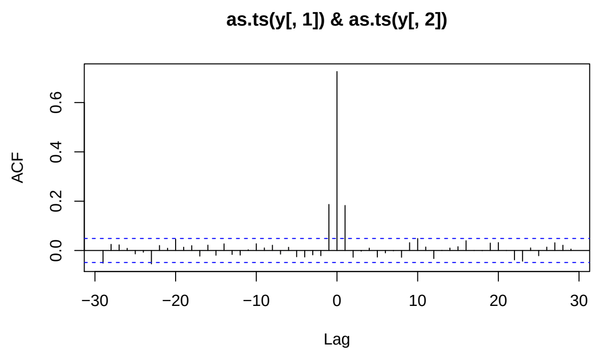



```r
acf(y[,2])
```



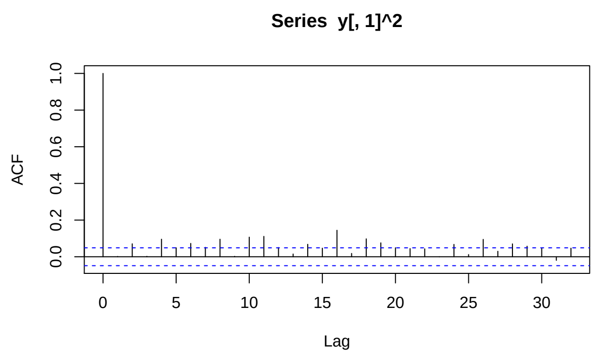
Cross correlation

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

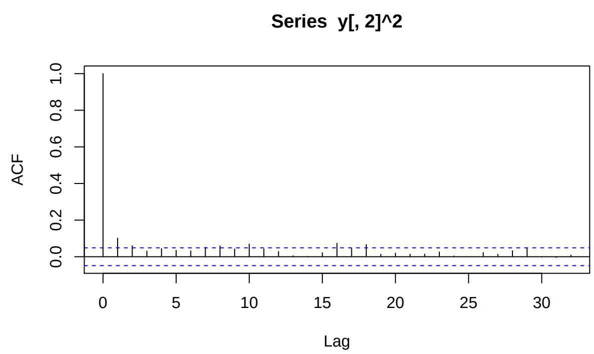


Volatility verification

```
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.

```r
# 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
## 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
##   0.46   0.19   0.18   0.21   0.22   0.18   0.18   0.22   0.22   0.28   0.32
## 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.
```

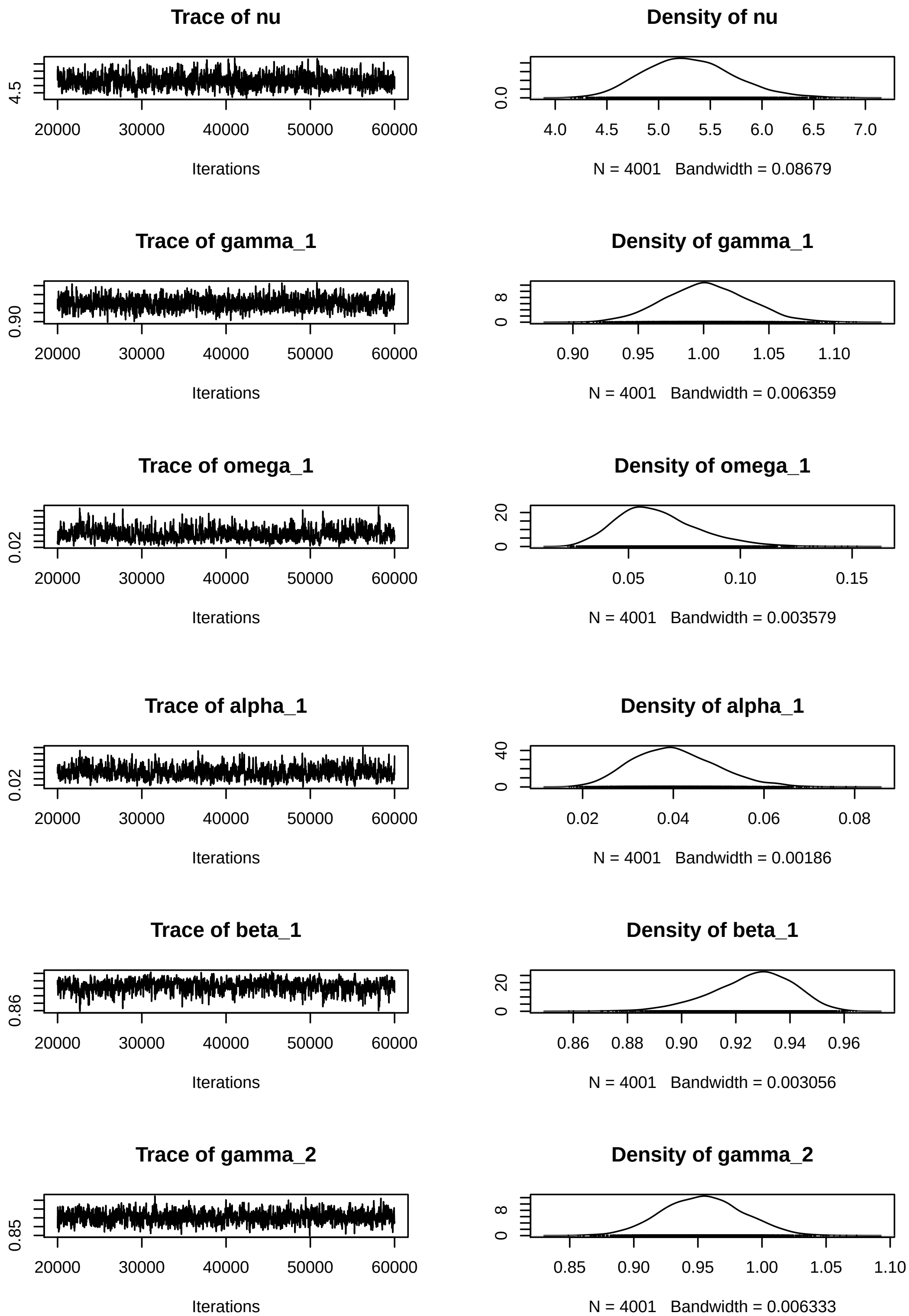
```r
out2 <- increaseSim(out, nSim=50000)
```

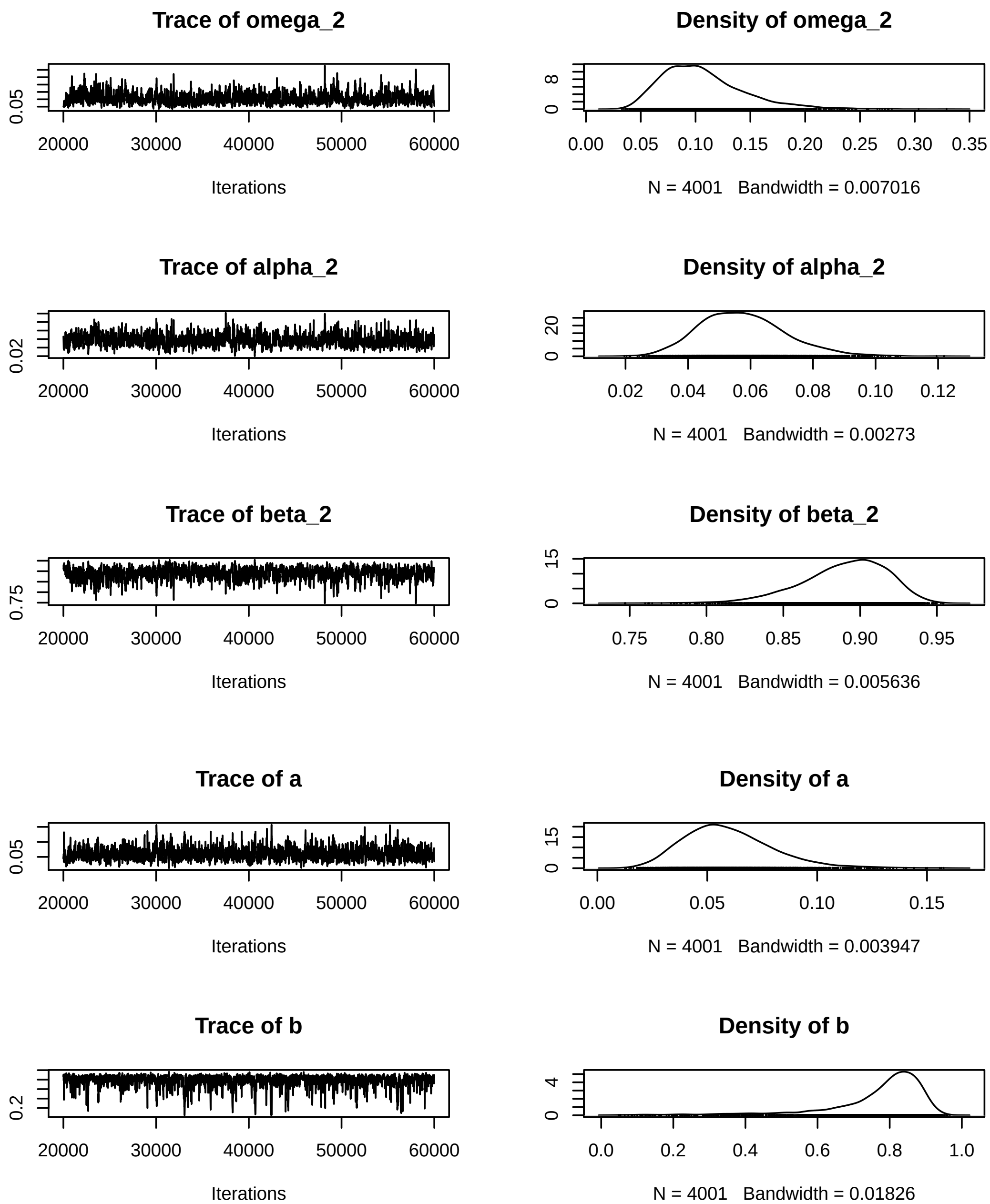
```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.52
## lambda: 0.48
## Accept Rate: 0.42
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

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

end <- Sys.time()

# elapsed time
end-start
```

```
## Time difference of 1.623644 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
4.0 4.5 5.0 5.5 6.0 6.5 7.0
N = 4001 Bandwidth = 0.08679
Trace of gamma_1
0.90
20000 30000 40000 50000 60000
Iterations
Density of gamma_1
0 8
0.90 0.95 1.00 1.05 1.10
N = 4001 Bandwidth = 0.006359
Trace of omega_1
0.02
20000 30000 40000 50000 60000
Iterations
Density of omega_1
0 20
0.05 0.10 0.15
N = 4001 Bandwidth = 0.003579
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 = 0.00186
Trace of beta_1
0.86
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 = 0.003056
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 1.10
N = 4001 Bandwidth = 0.006333

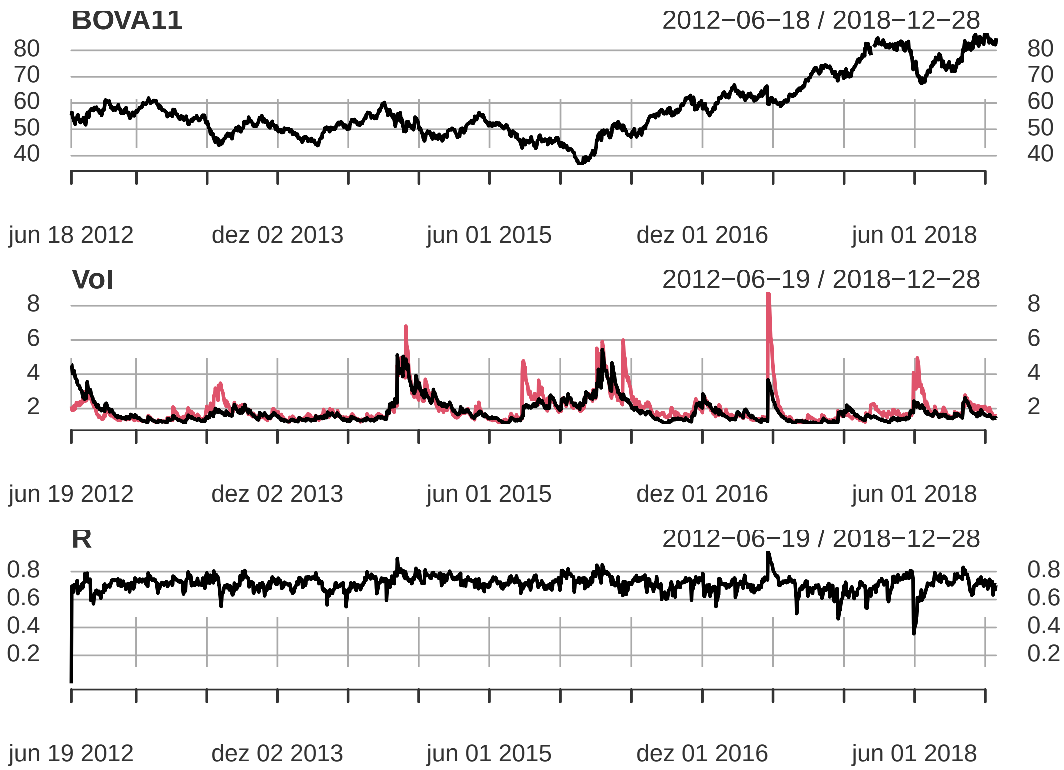



```r
## Estimative of parameters
out$MC %>% summary()
```

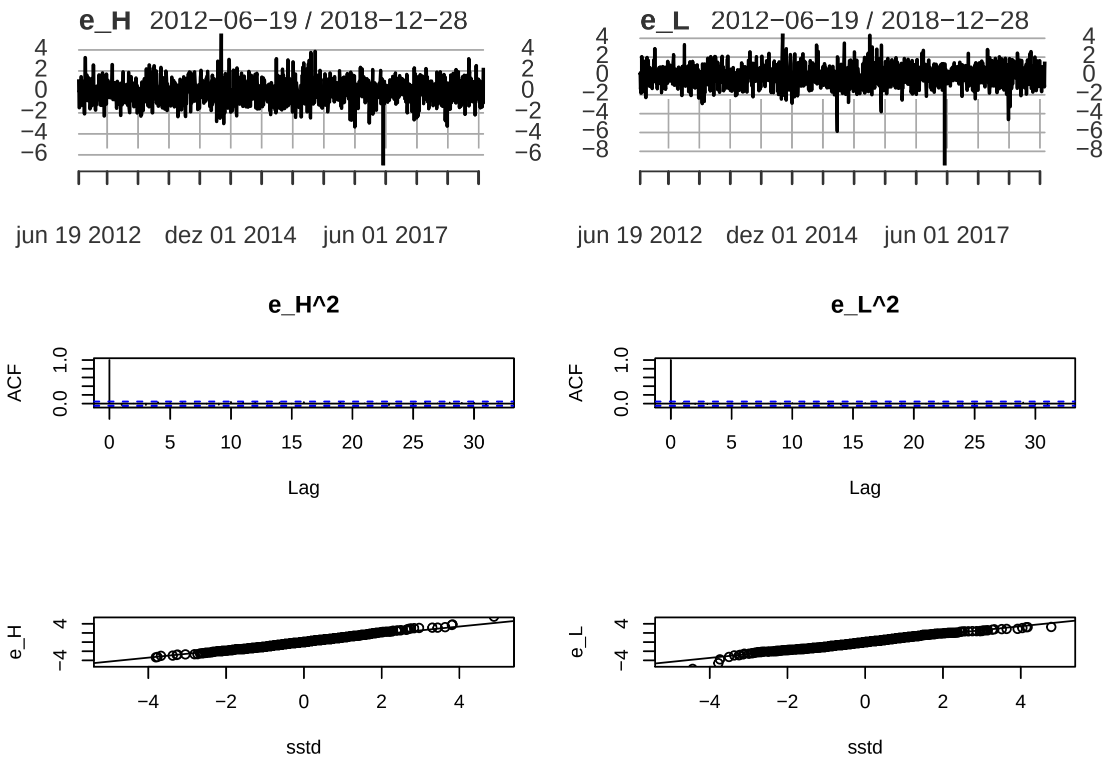
```
##
```

```
## 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      5.29855 0.430790 0.0068105      0.0146018
## gamma_1 1.00238 0.031919 0.0005046      0.0010599
## omega_1 0.06316 0.018494 0.0002924      0.0008458
## alpha_1 0.04034 0.009216 0.0001457      0.0003844
## beta_1  0.92618 0.015145 0.0002394      0.0007062
## gamma_2 0.95412 0.031388 0.0004962      0.0011328
## omega_2 0.10636 0.037812 0.0005978      0.0015569
## alpha_2 0.05761 0.013730 0.0002171      0.0005260
## beta_2  0.88939 0.028598 0.0004521      0.0012215
## a       0.05965 0.020422 0.0003229      0.0007044
## b       0.76680 0.142784 0.0022573      0.0068405
##
## 2. Quantiles for each variable:
##
##            2.5%     25%     50%     75%   97.5%
## nu      4.53190 4.99495 5.27292 5.57130 6.21321
## gamma_1 0.94128 0.98086 1.00167 1.02309 1.06692
## omega_1 0.03317 0.05005 0.06073 0.07382 0.10568
## alpha_1 0.02433 0.03371 0.03967 0.04635 0.06087
## beta_1  0.89274 0.91675 0.92776 0.93710 0.95124
## gamma_2 0.89367 0.93250 0.95380 0.97469 1.01599
## omega_2 0.05159 0.07919 0.10048 0.12578 0.19822
## alpha_2 0.03371 0.04781 0.05666 0.06595 0.08737
## beta_2  0.82471 0.87267 0.89305 0.91009 0.93394
## a       0.02697 0.04519 0.05723 0.07140 0.10611
## b       0.33388 0.73440 0.80903 0.85568 0.91372
```

```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")
```




```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)
```




```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",
```

```r
                      "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  0.209  0.320  0.439  0.570  0.722  0.906  1.155  1.572  1.997  2.598
Low_LB  -0.191 -0.304 -0.426 -0.561 -0.717 -0.908 -1.167 -1.602 -2.047 -2.676
          0.5%
High_UB  3.097
Low_LB  -3.200
```

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

```
              Value
High_UB_HBOP  1.997
High_UB_S1    0.570
Low_LB_B1    -0.561
Low_LB_LBOP  -2.047
High_omega    0.063
High_alpha    0.040
High_beta     0.926
Low_omega     0.106
Low_alpha     0.058
Low_beta      0.889
```