

GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

05/02/2021

Input

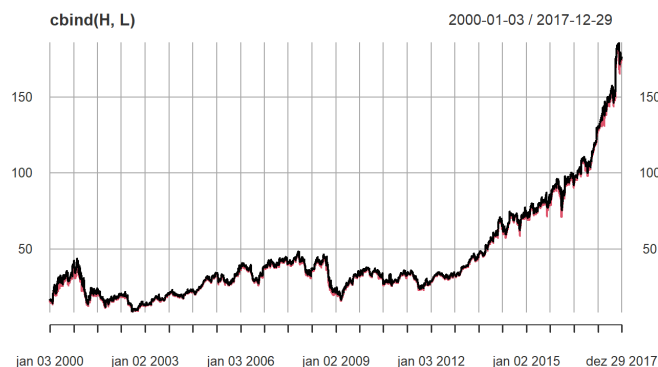
```
symbol = "ADBE"
from=as.Date('2000-01-01')
to=as.Date('2017-12-31')
C_Trend = 0.95
C_Reaction = 0.50
```

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))
```

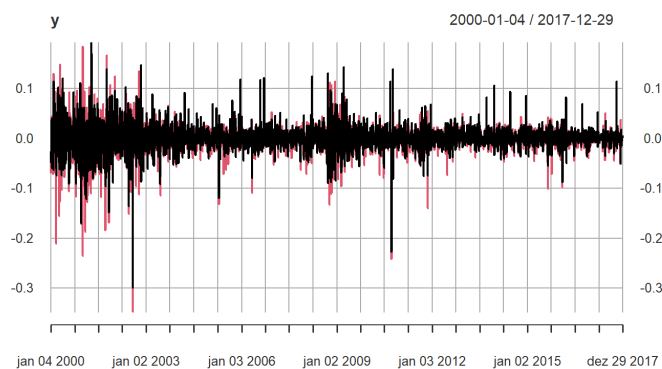


Returns

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

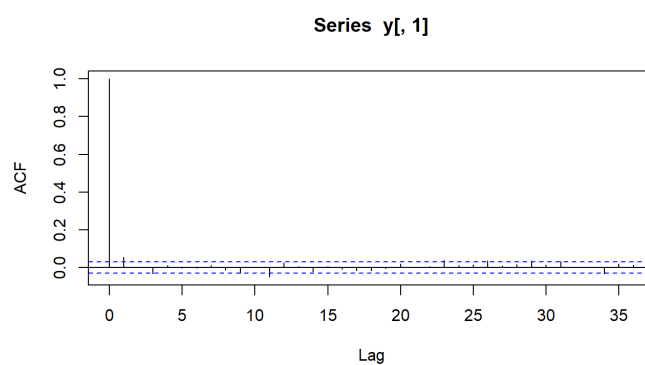
```
##          ADBE.High  ADBE.Low
## ADBE.High 1.0000000 0.7151885
## ADBE.Low  0.7151885 1.0000000
```

```
plot(y)
```

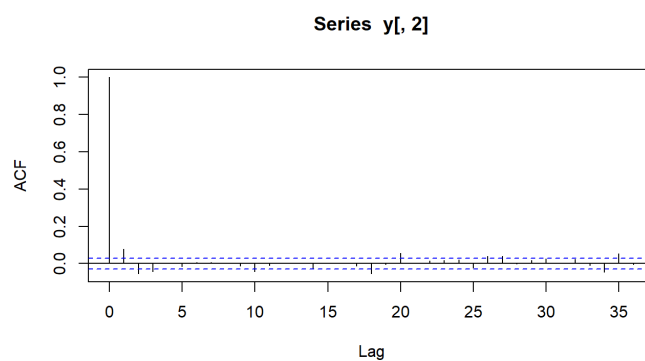


Autocorrelation

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

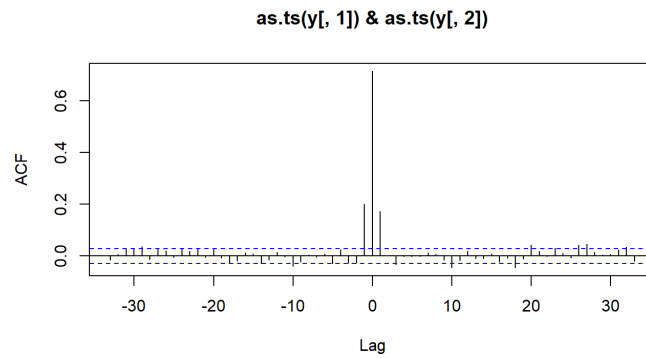


```
acf(y[,2])
```



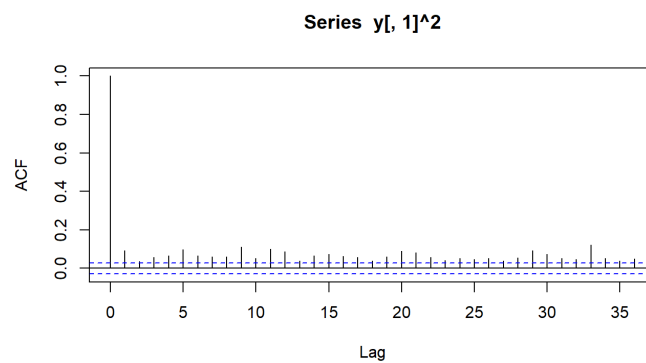
Cross correlation

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

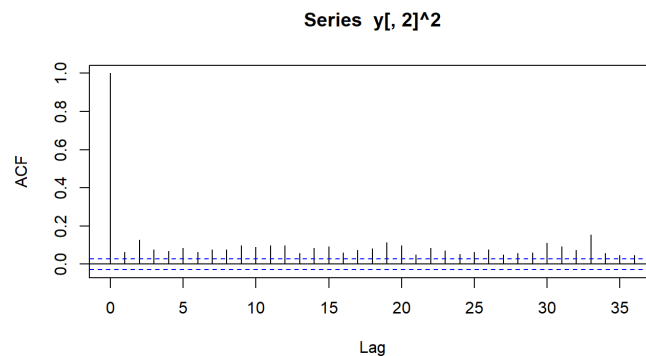


Volatility verification

```
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 high's and low's observations.

```
# 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 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
##   0.28   0.09   0.21   0.04   0.07   0.09   0.19   0.06   0.08   0.28   0.38
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9  phi_10  phi_11
##   0.27   0.17   0.21   0.07   0.10   0.17   0.20   0.07   0.10   0.30   0.42
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9  phi_10  phi_11
##   0.28   0.15   0.21   0.09   0.15   0.16   0.20   0.14   0.20   0.29   0.39
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9  phi_10  phi_11
##   0.28   0.15   0.21   0.17   0.27   0.18   0.20   0.20   0.19   0.31   0.41
## 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: 0.37
## 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: 0.37
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

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

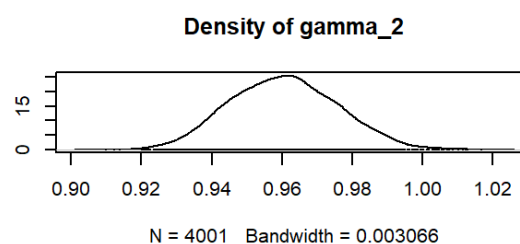
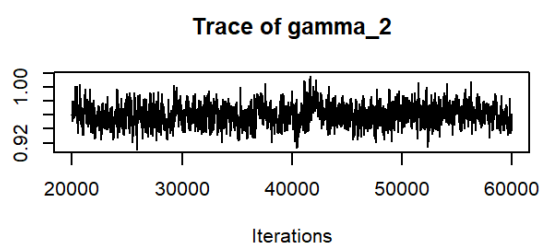
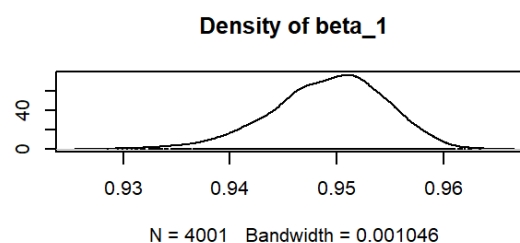
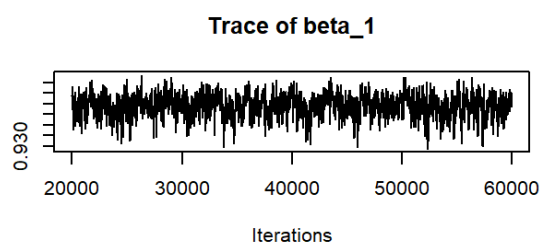
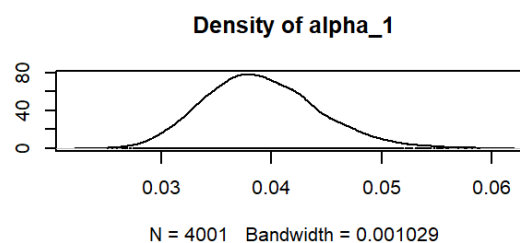
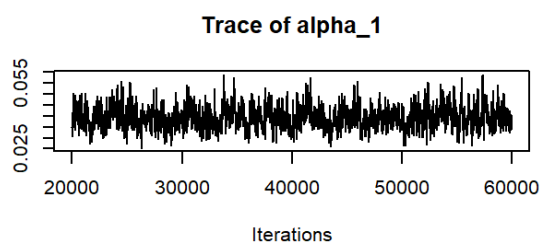
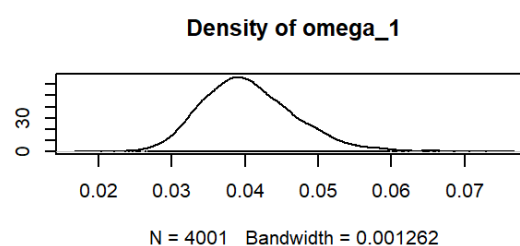
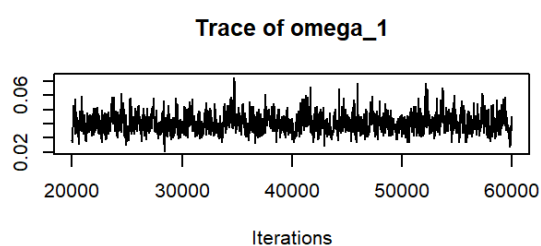
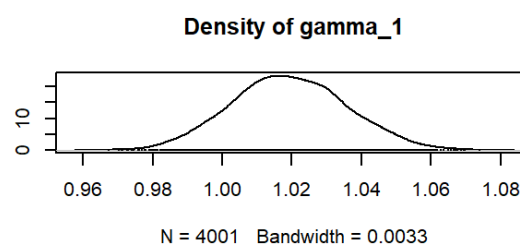
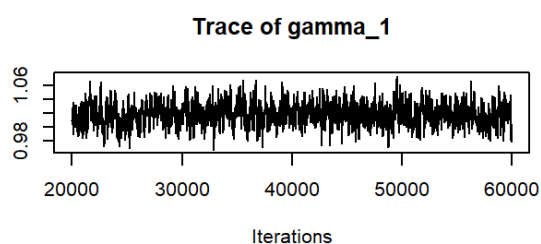
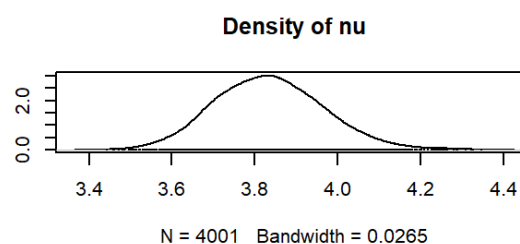
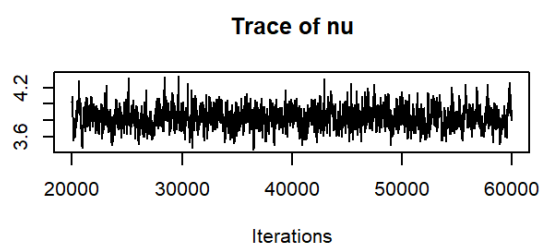
```
end <- Sys.time()
```

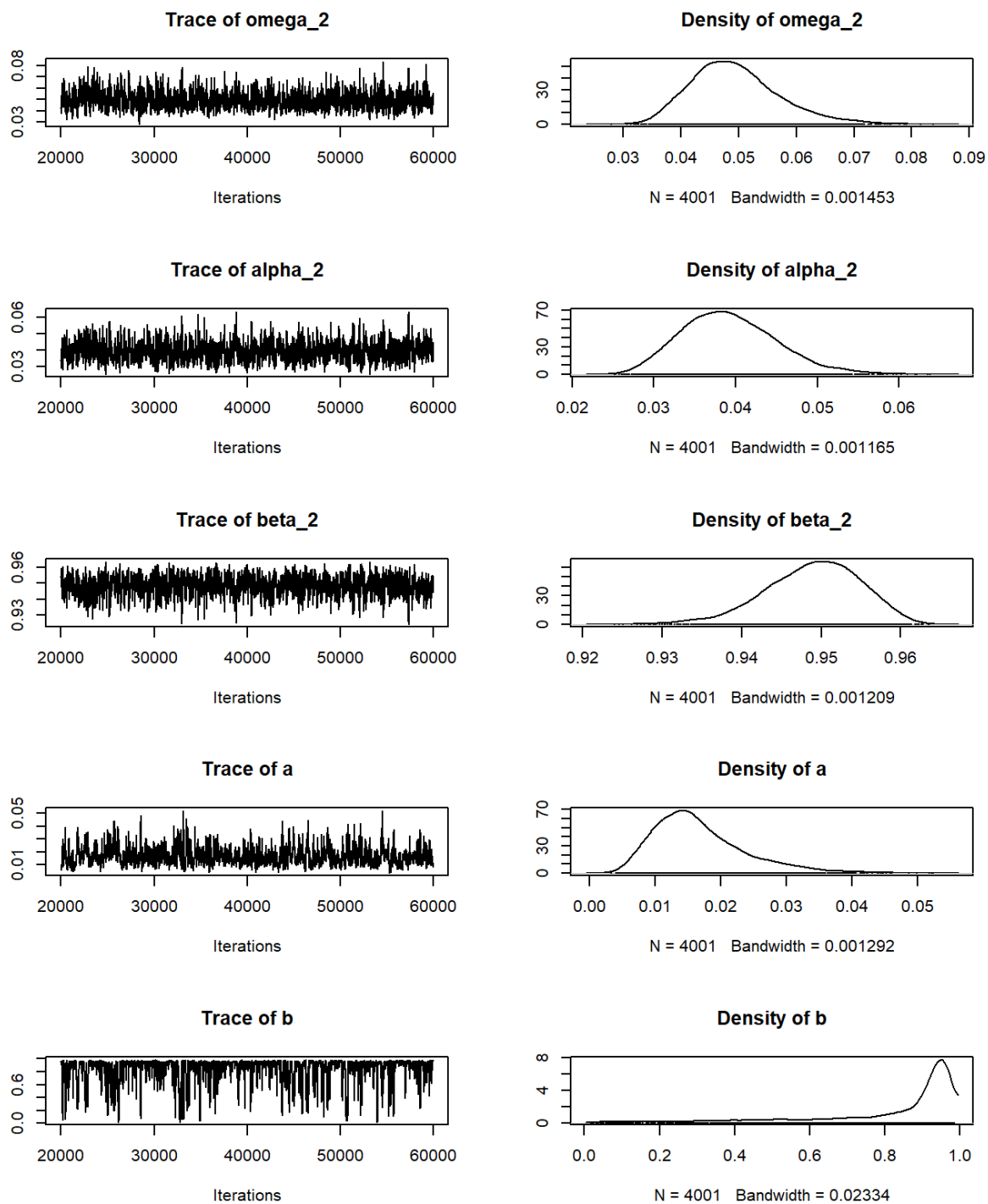
```
# elapsed time
end-start
```

```
## Time difference of 16.86253 mins
```

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

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





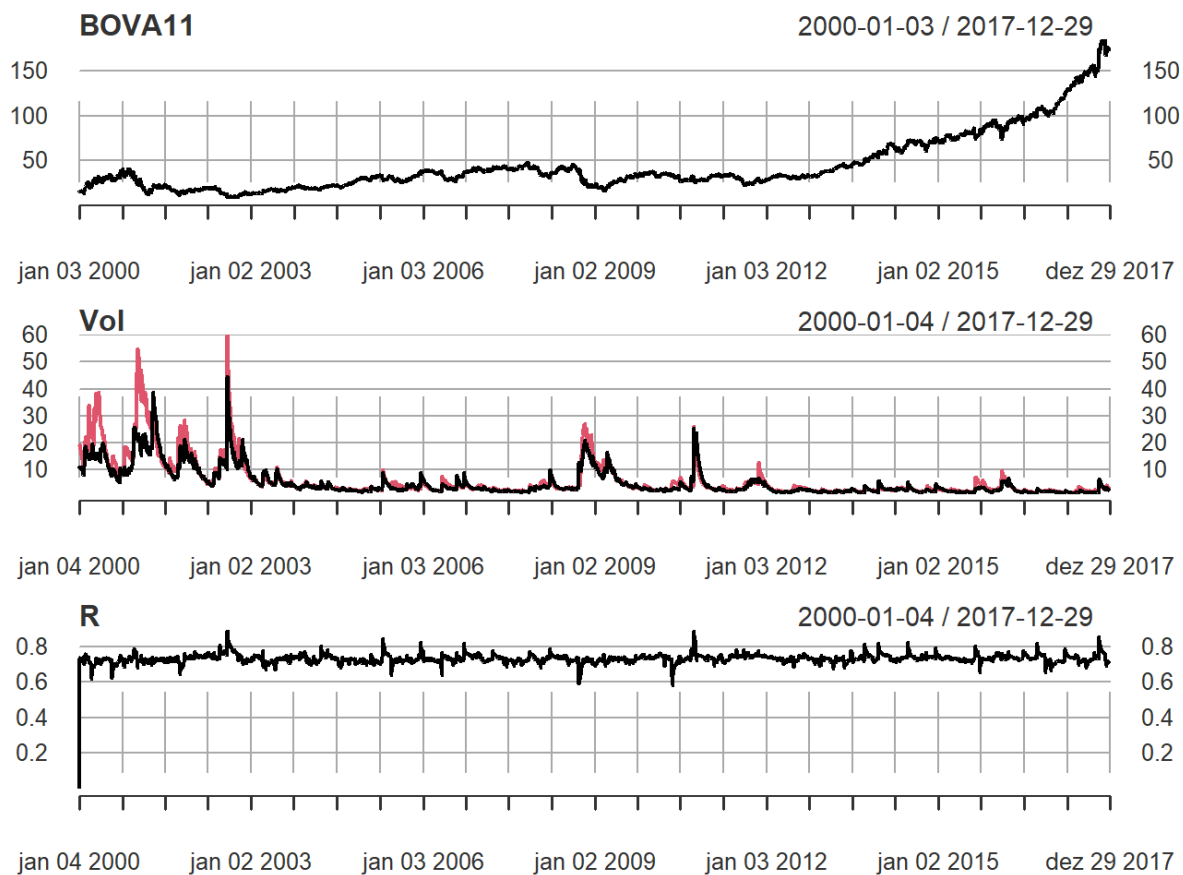
```
## Estimative of parameters
out$MC %>% summary()
```

```
##
## 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       3.83546 0.131442 2.078e-03    0.0059848
## gamma_1  1.01916 0.016356 2.586e-04    0.0007178
## omega_1   0.04056 0.006335 1.001e-04    0.0003311
## alpha_1   0.03910 0.005131 8.112e-05    0.0002540
## beta_1    0.94920 0.005294 8.370e-05    0.0002565
## gamma_2   0.96055 0.015193 2.402e-04    0.0006216
## omega_2   0.04937 0.007626 1.206e-04    0.0002817
## alpha_2   0.03920 0.005774 9.128e-05    0.0002127
## beta_2    0.94892 0.005993 9.474e-05    0.0002160
## a         0.01666 0.007042 1.113e-04    0.0003512
## b         0.82789 0.205685 3.252e-03    0.0117642
##
## 2. Quantiles for each variable:
##
##          2.5%      25%      50%      75%     97.5%
## nu       3.592960 3.74435 3.83103 3.92037 4.10947
## gamma_1  0.987898 1.00787 1.01876 1.03014 1.05131
## omega_1   0.029953 0.03611 0.03999 0.04449 0.05453
## alpha_1   0.030080 0.03551 0.03875 0.04234 0.05021
## beta_1    0.937720 0.94594 0.94963 0.95289 0.95846
## gamma_2   0.932474 0.94974 0.96049 0.97093 0.99042
## omega_2   0.036821 0.04400 0.04857 0.05365 0.06682
## alpha_2   0.029285 0.03505 0.03874 0.04285 0.05200
## beta_2    0.935447 0.94504 0.94934 0.95318 0.95919
## a         0.006633 0.01163 0.01531 0.02021 0.03397
## b         0.197215 0.79754 0.92525 0.95255 0.97466
```

```
## 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")
```



```
## 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")
```

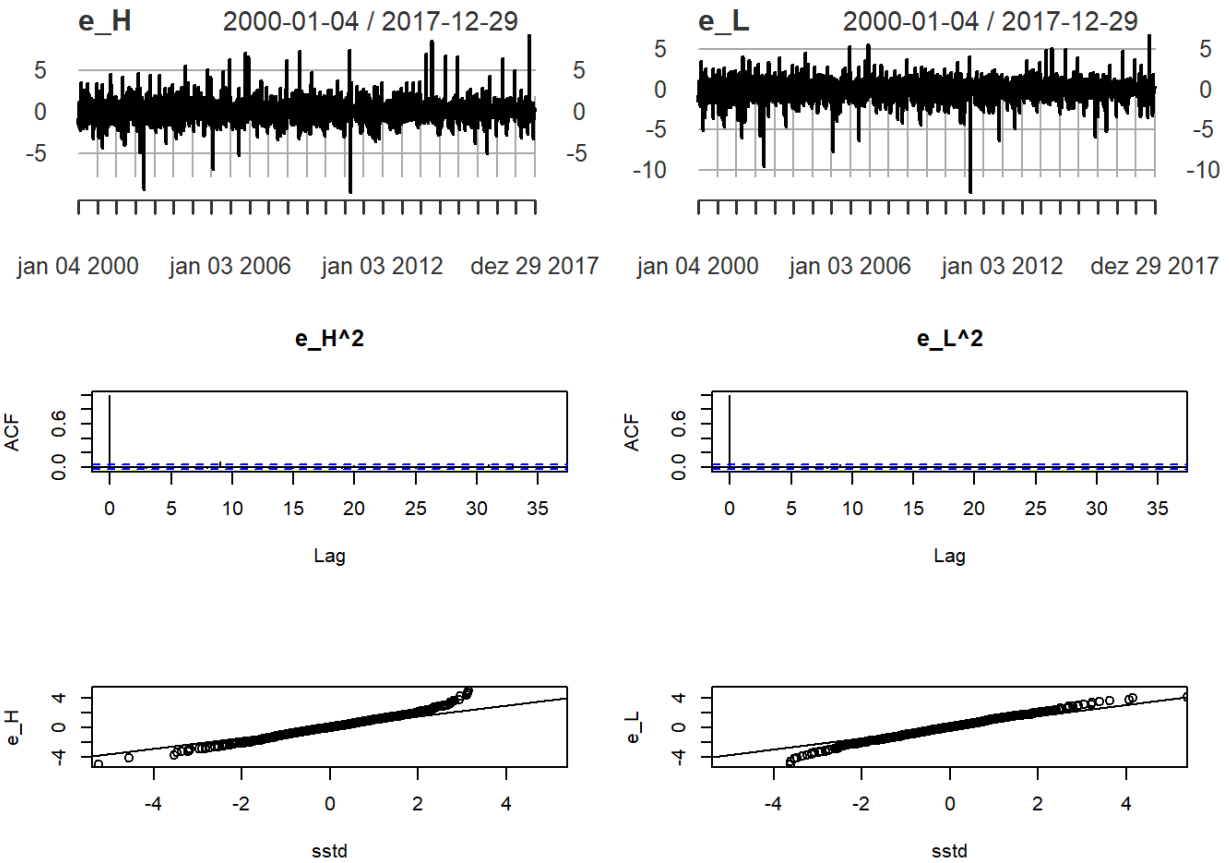
```
qqline(r1)
```

```
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)
```

```

# Prepare input for the expert advisor

## High
#HBOP
High_UB_HBOP = qstd(p=1-(1-C_Trend)/2, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['gamma_1'])
#S1
High_UB_S1 = qstd(p=1-(1-C_Reaction)/2, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['gamma_1'])

## Low
#B1
Low_LB_B1 = qstd(p=(1-C_Reaction)/2, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['gamma_2'])
#LBOP
Low_LB_LBOP = qstd(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(qstd(p=pH, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['gamma_1']),3)
names(qH) <- paste0(100*pH,"%")
pL <- 1 - pH
qL <- round(qstd(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	0.18	0.280	0.389	0.510	0.652	0.829	1.074	1.507	1.977	2.694
Low_LB	-0.17	-0.272	-0.382	-0.505	-0.649	-0.829	-1.080	-1.522	-2.003	-2.737
	0.5%									
High_UB	3.334									
Low_LB	-3.393									

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

	Value
High_UB_HBOP	1.977
High_UB_S1	0.510
Low_LB_B1	-0.505
Low_LB_LBOP	-2.003
High_omega	0.041
High_alpha	0.039
High_beta	0.949
Low_omega	0.049
Low_alpha	0.039
Low_beta	0.949