

# GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

05/02/2021

## Input

```
symbol = "BIDU"
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))
```

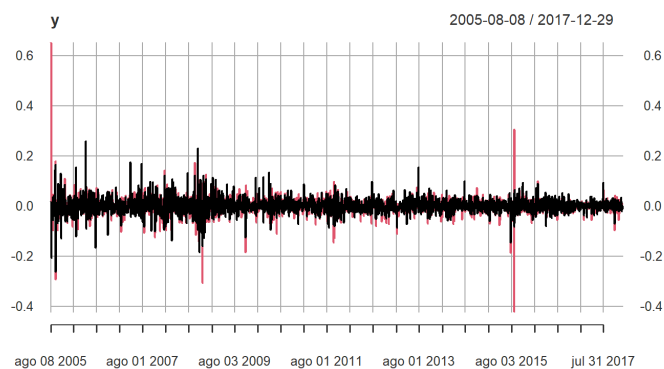


## Returns

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

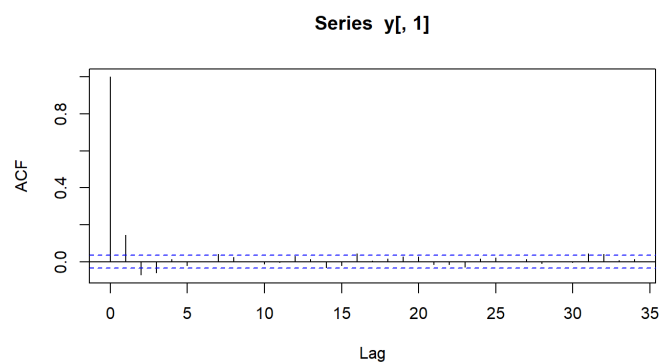
```
##          BIDU.High BIDU.Low
## BIDU.High 1.0000000 0.6907737
## BIDU.Low  0.6907737 1.0000000
```

```
plot(y)
```

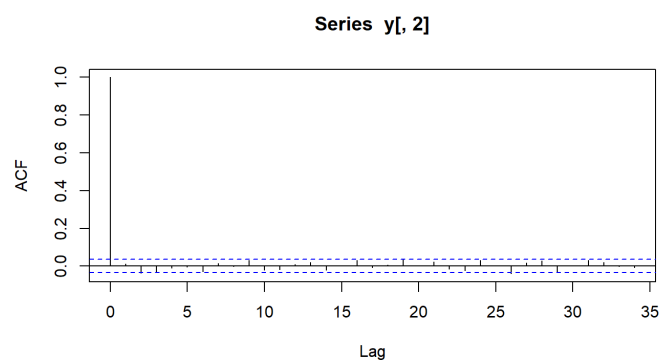


## Autocorrelation

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

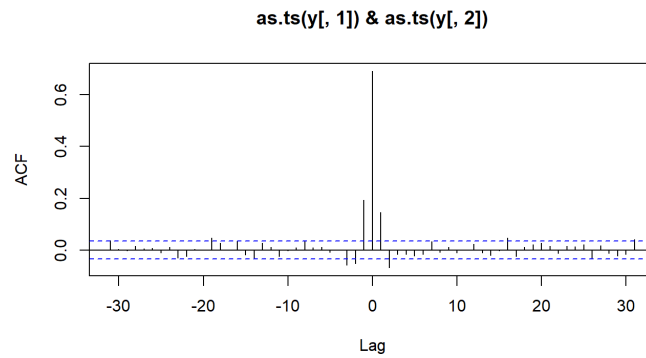


```
acf(y[,2])
```



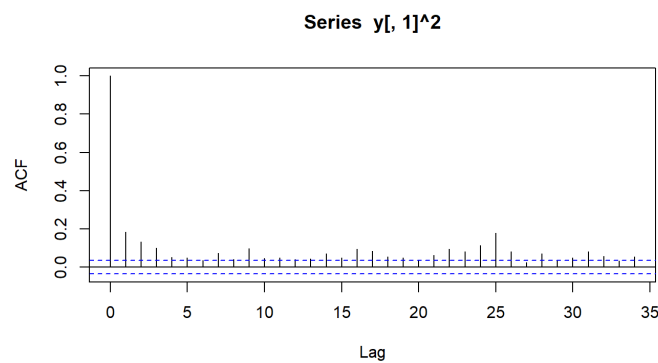
## Cross correlation

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

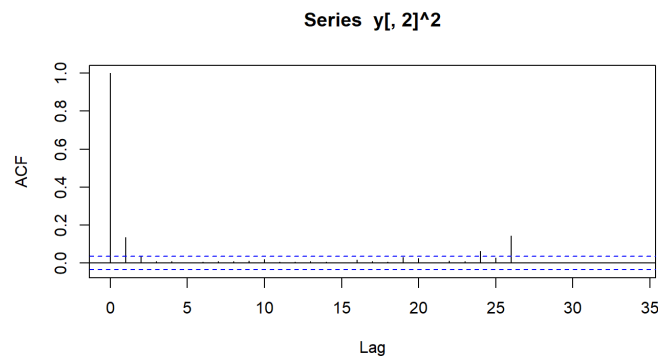


## Volatility verification

```
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
##   0.31   0.11   0.19   0.12   0.14   0.11   0.18   0.14   0.16   0.45   0.62
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9  phi_10  phi_11
##   0.32   0.20   0.20   0.20   0.23   0.20   0.18   0.22   0.15   0.46   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
##   0.32   0.19   0.19   0.19   0.22   0.20   0.18   0.23   0.17   0.48   0.42
## 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: 0.44
## 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: 0.47
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

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

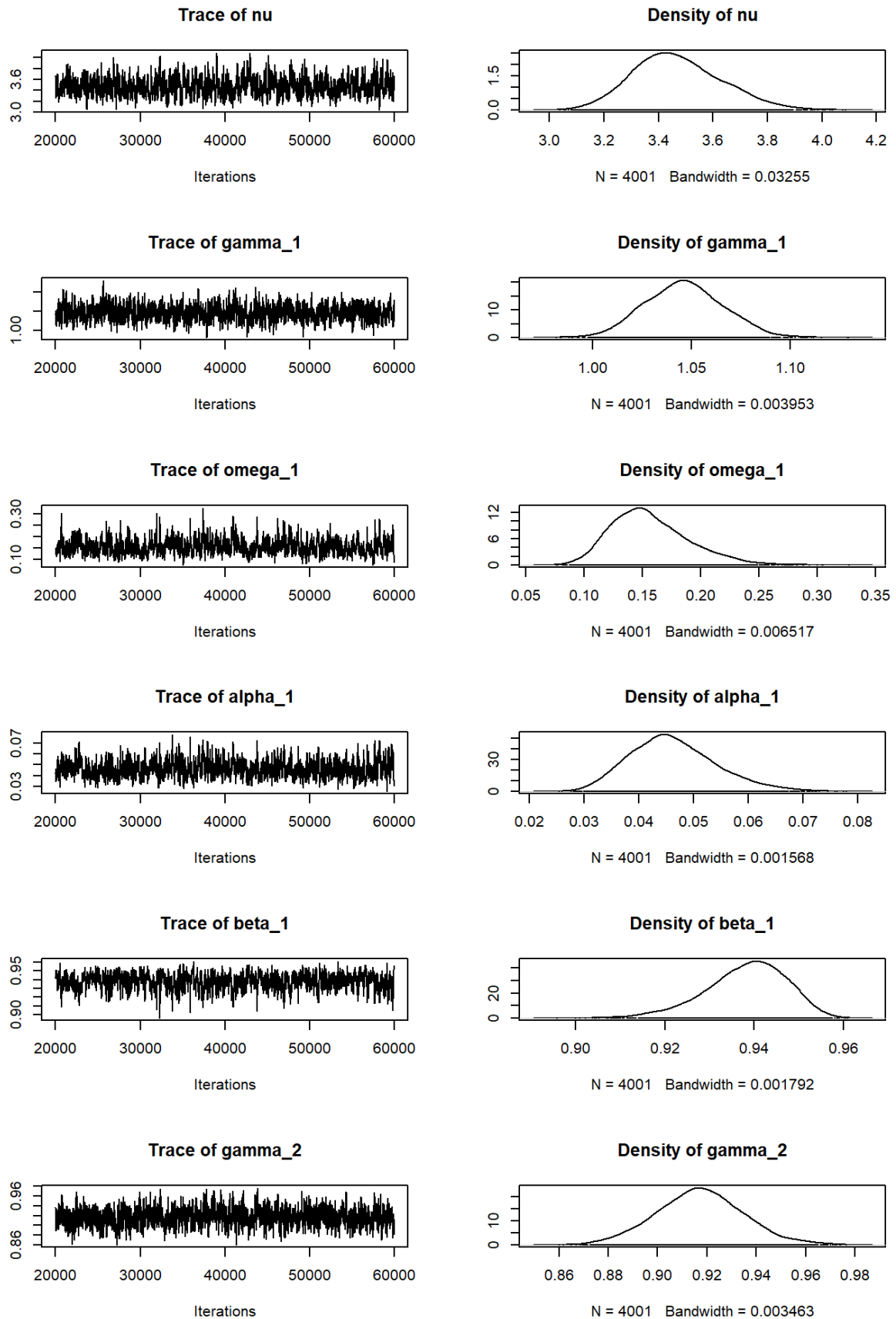
end <- Sys.time()

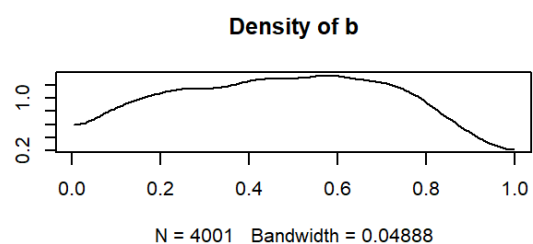
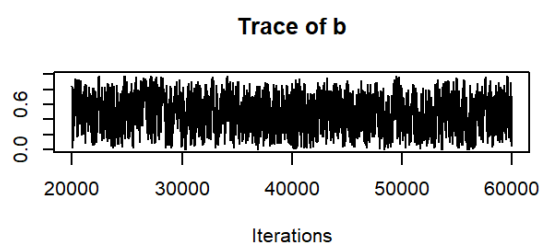
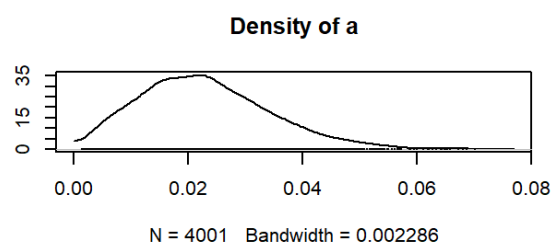
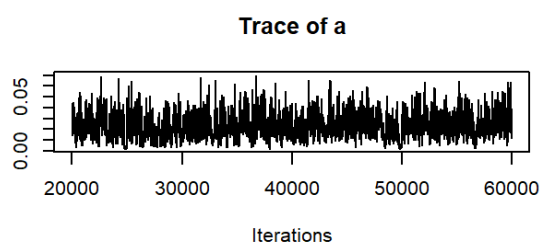
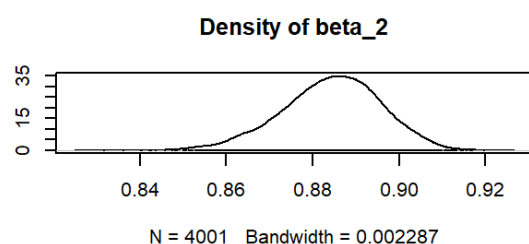
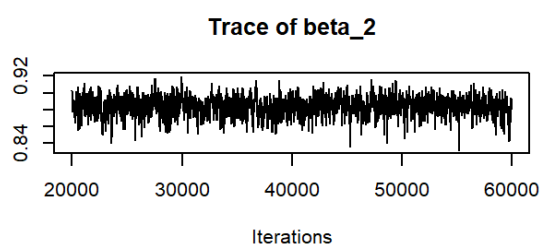
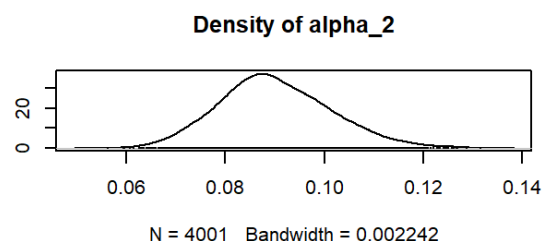
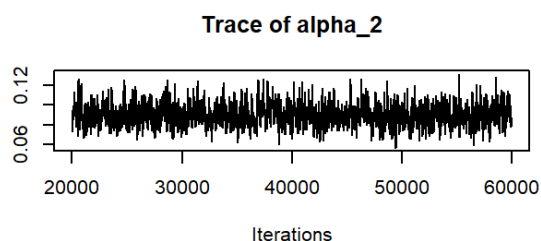
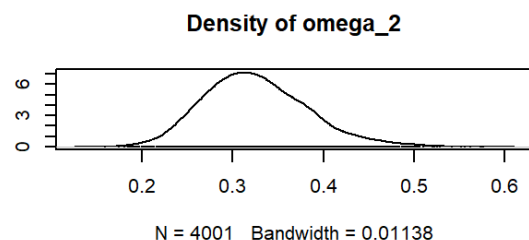
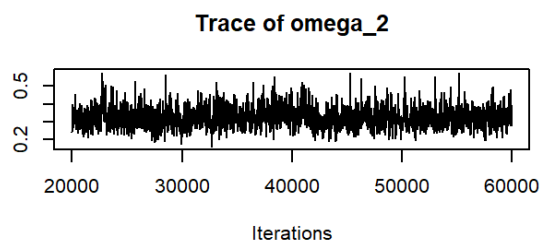
# elapsed time
end-start
```

```
## Time difference of 7.961156 mins
```

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

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





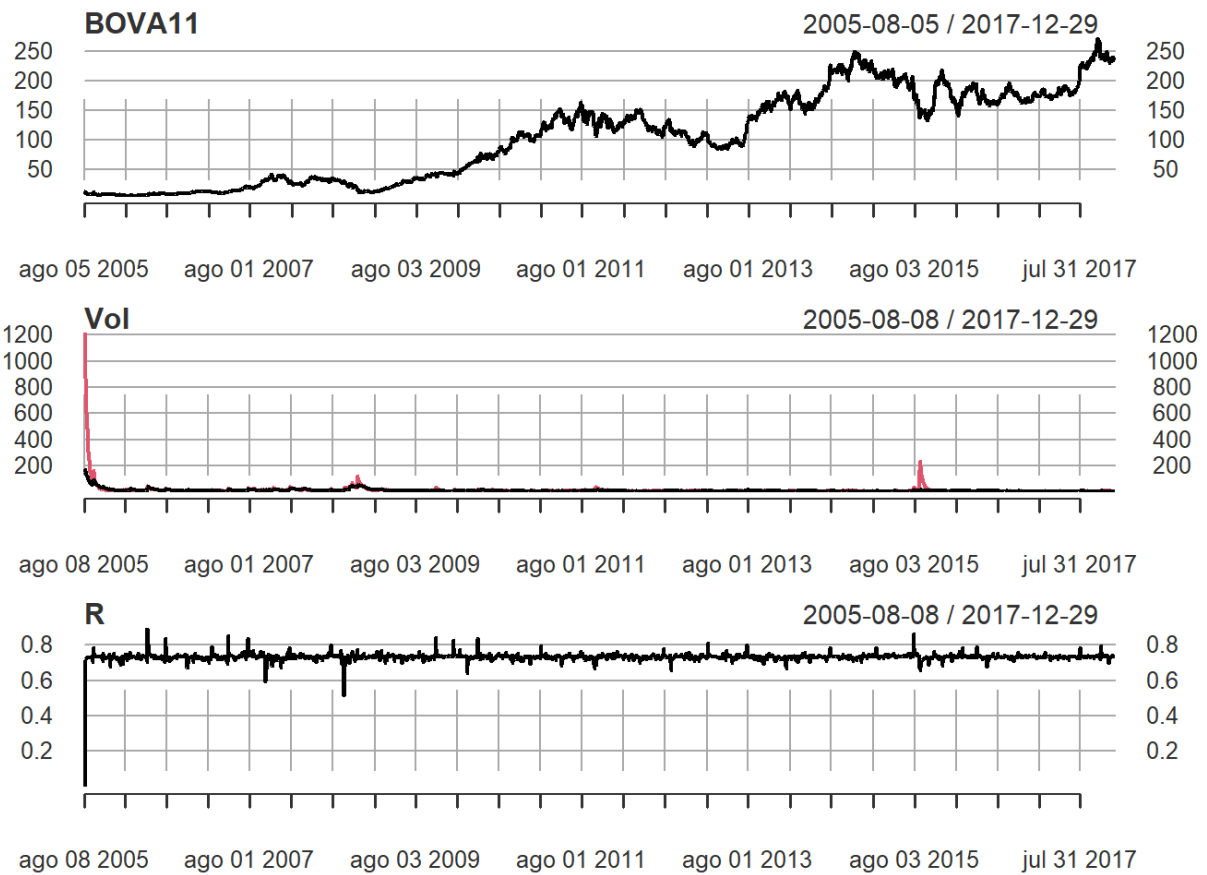
```
## Estimative of parameters
out$MC %>% summary()
```

```
##
## 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      3.46953 0.161295 0.0025500    0.0070830
## gamma_1 1.04634 0.019685 0.0003112    0.0007400
## omega_1 0.15656 0.033671 0.0005323    0.0016143
## alpha_1 0.04628 0.007827 0.0001237    0.0003444
## beta_1  0.93806 0.009085 0.0001436    0.0004384
## gamma_2 0.91714 0.017624 0.0002786    0.0006315
## omega_2 0.32553 0.057318 0.0009062    0.0019317
## alpha_2 0.09006 0.011183 0.0001768    0.0004378
## beta_2  0.88423 0.011487 0.0001816    0.0004219
## a       0.02300 0.011366 0.0001797    0.0004206
## b       0.47363 0.242272 0.0038302    0.0087305
##
## 2. Quantiles for each variable:
##
##          2.5%      25%      50%      75%      97.5%
## nu      3.182443 3.35403 3.45468 3.57429 3.81363
## gamma_1 1.009616 1.03277 1.04611 1.05902 1.08500
## omega_1 0.102939 0.13249 0.15214 0.17577 0.23124
## alpha_1 0.032849 0.04073 0.04558 0.05114 0.06375
## beta_1  0.917443 0.93263 0.93905 0.94452 0.95311
## gamma_2 0.882842 0.90543 0.91696 0.92843 0.95364
## omega_2 0.228322 0.28534 0.32091 0.36091 0.45425
## alpha_2 0.069952 0.08238 0.08922 0.09727 0.11381
## beta_2  0.860157 0.87698 0.88495 0.89217 0.90495
## a       0.004653 0.01479 0.02202 0.02997 0.04853
## b       0.049212 0.27342 0.47727 0.67022 0.90255
```

```
## 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)
```

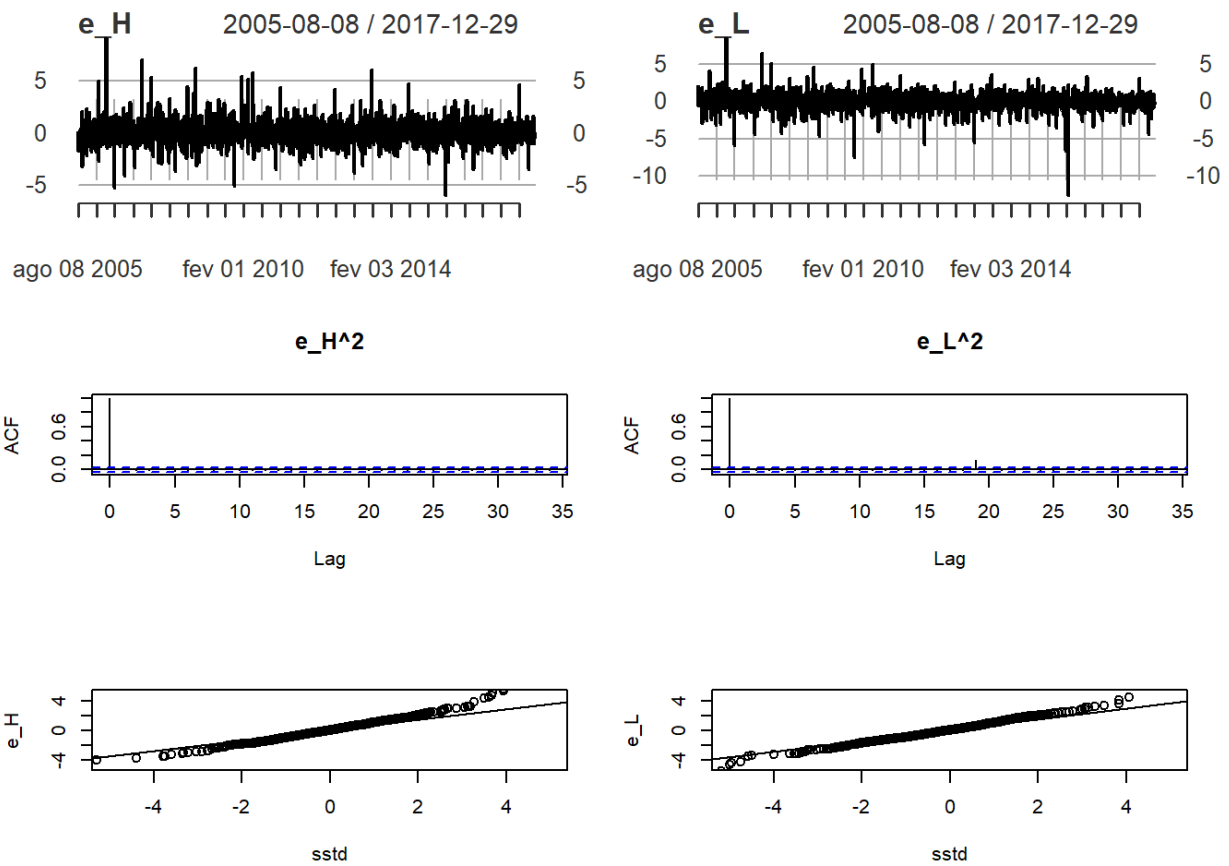
```
r2 <- as.numeric(r[,2])
```

```
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	0.158	0.254	0.359	0.478	0.617	0.792	1.039	1.483	1.978	2.755
Low_LB	-0.140	-0.238	-0.346	-0.467	-0.610	-0.792	-1.048	-1.510	-2.027	-2.839
	0.5%									
High_UB	3.468									
Low_LB	-3.585									

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

	Value
High_UB_HBOP	1.978
High_UB_S1	0.478
Low_LB_B1	-0.467
Low_LB_LBOP	-2.027
High_omega	0.157
High_alpha	0.046
High_beta	0.938
Low_omega	0.326
Low_alpha	0.090
Low_beta	0.884