GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

20/11/2020

Input

```
symbol = "ITSA4.SA"#"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())
```

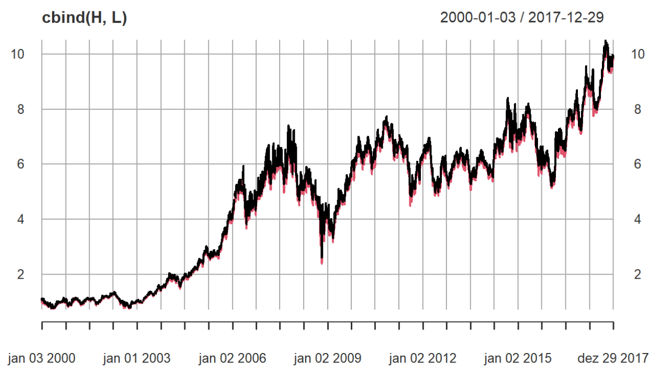
```
## Warning: ITSA4.SA contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## [1] "ITSA4.SA"
```

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

High and Low

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

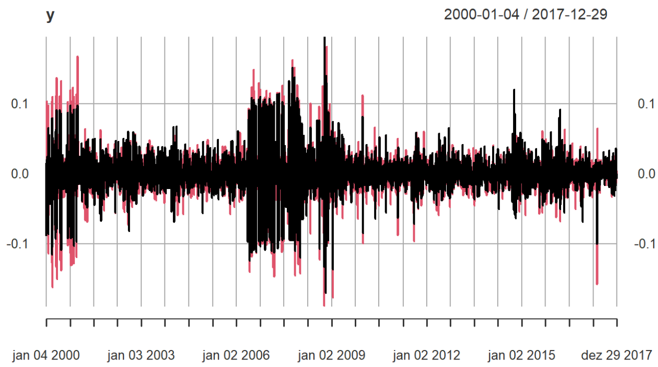


Returns

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

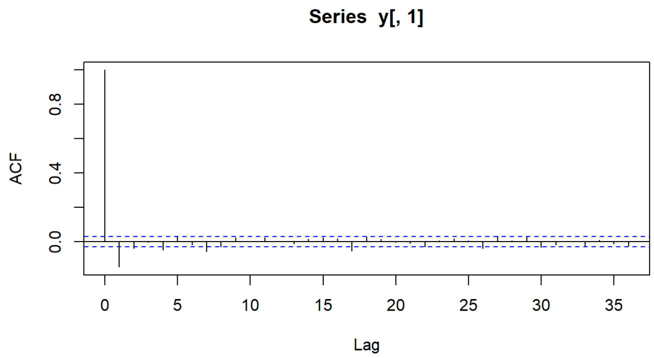
```
##                ITSA4.SA.High ITSA4.SA.Low
## ITSA4.SA.High     1.0000000    0.8336175
## ITSA4.SA.Low      0.8336175    1.0000000
```

```
plot(y)
```

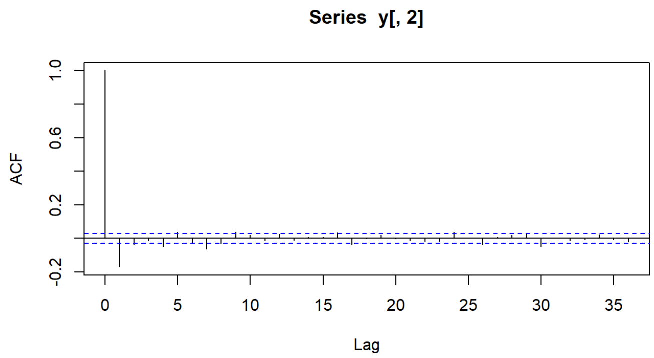


Autocorrelation

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

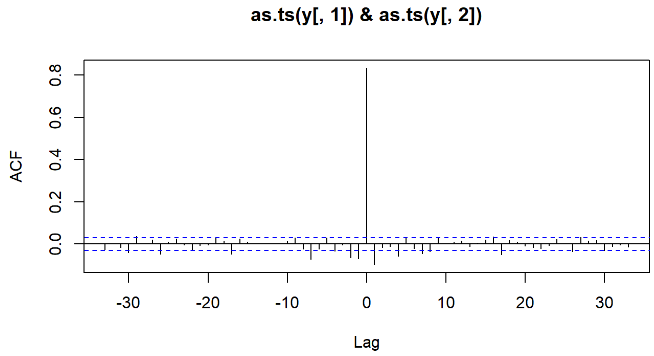



```
acf(y[,2])
```



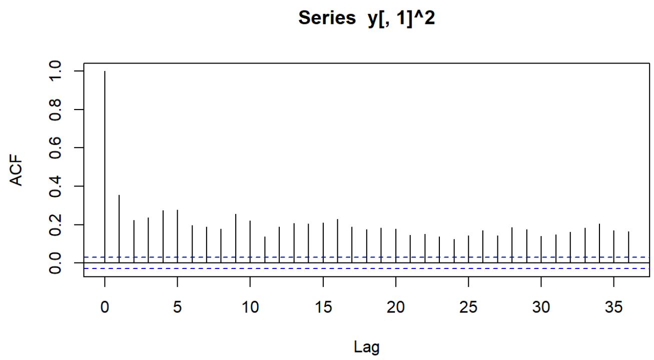
Cross correlation

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

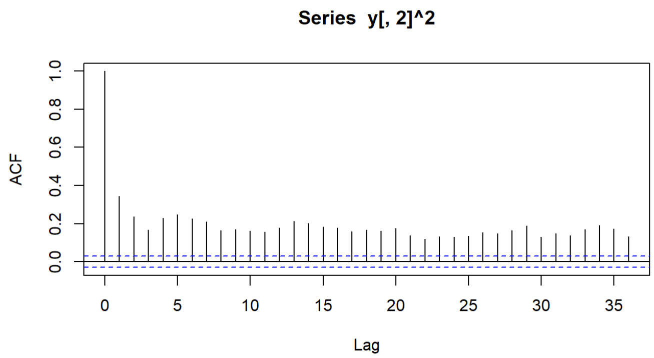


Volatility verification

```
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
##   0.35   0.09   0.19   0.09   0.12   0.11   0.15   0.09   0.11   0.03   0.02
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.36   0.19   0.16   0.14   0.22   0.18   0.28   0.14   0.20   0.05   0.03
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.35   0.16   0.18   0.21   0.19   0.16   0.25   0.22   0.21   0.09   0.04
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.35   0.18   0.20   0.22   0.21   0.21   0.28   0.21   0.19   0.11   0.06
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.38   0.18   0.18   0.25   0.19   0.18   0.24   0.22   0.20   0.23   0.11
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.35   0.19   0.16   0.24   0.20   0.20   0.27   0.23   0.20   0.20   0.19
## 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: 0.31
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

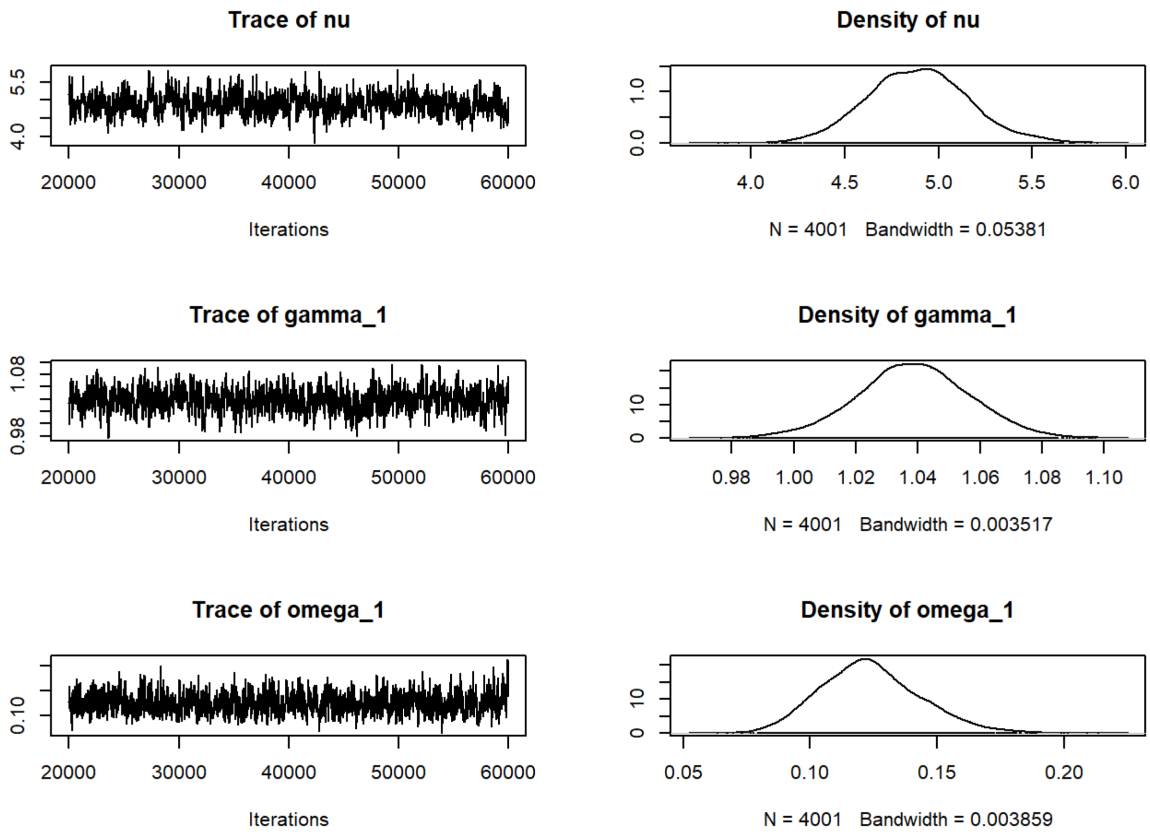
```
out2 <- increaseSim(out, nSim=50000)
```

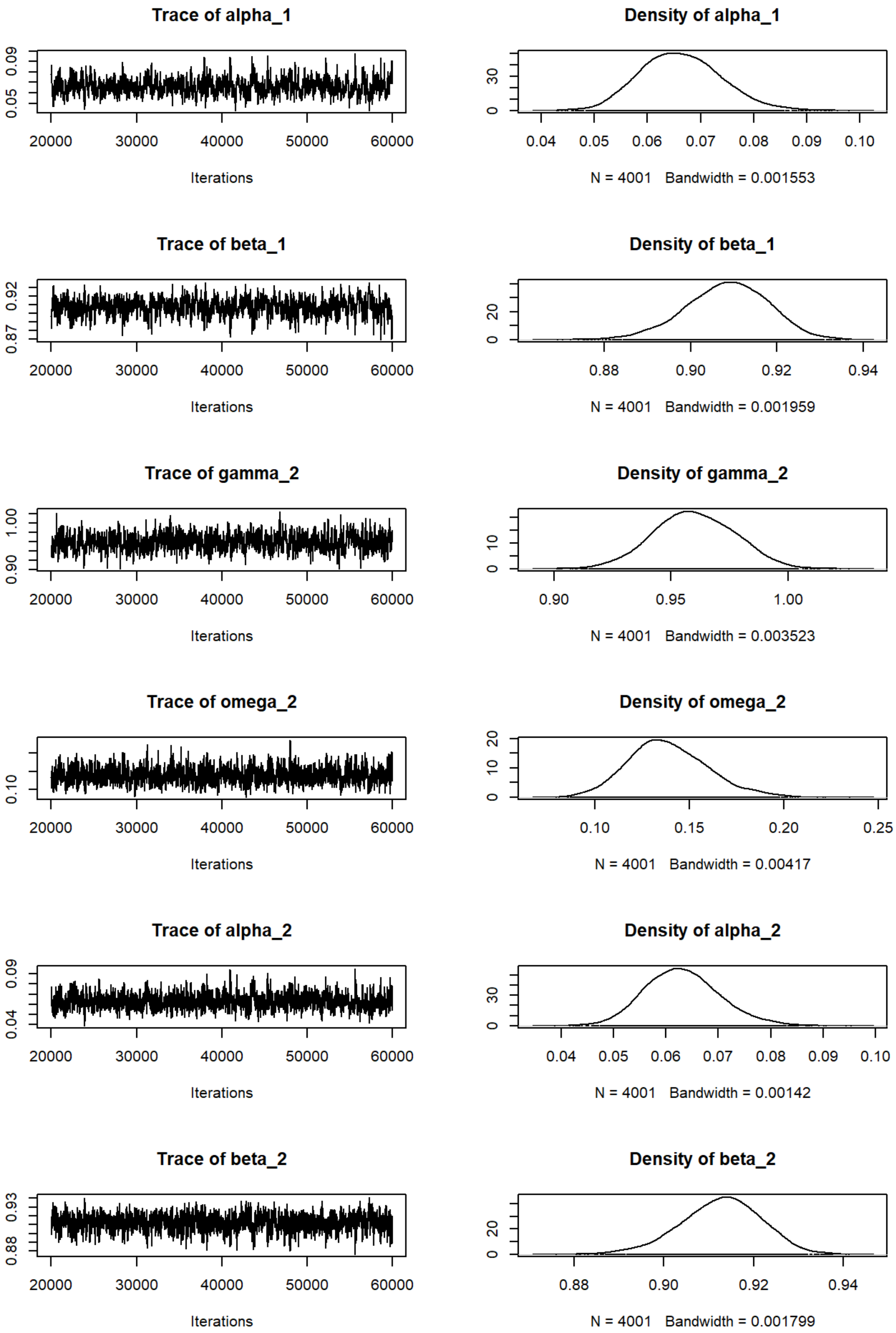
```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.36
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

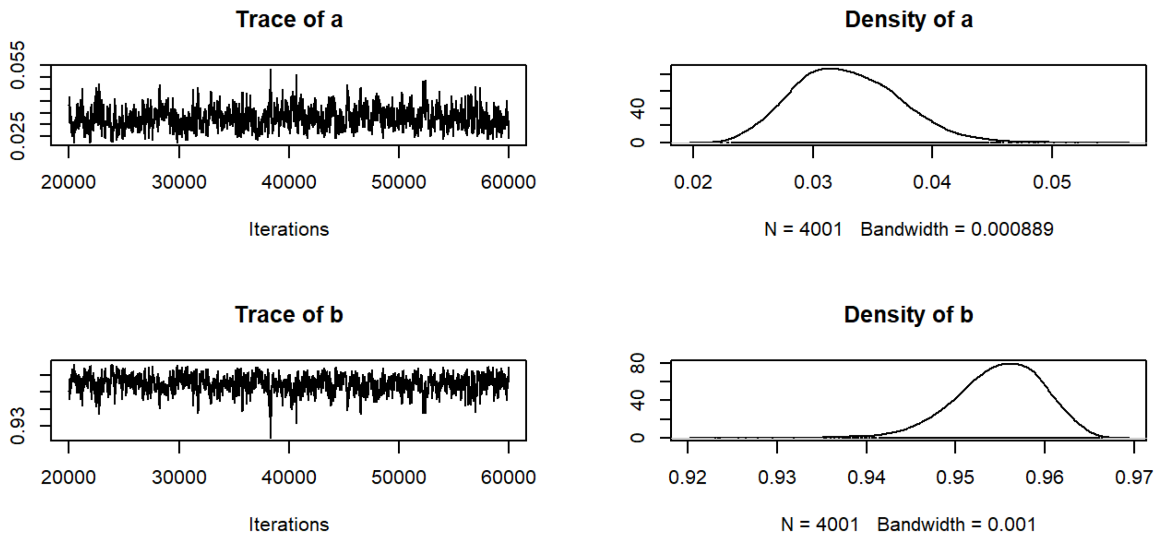
```
out <- window(out2, start=20000, thin=10)
rm(out2)

end <- Sys.time()

# elapsed time
end-start
```

```
## Time difference of 6.70874 mins
```

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




Trace of alpha_1
Density of alpha_1
N = 4001 Bandwidth = 0.001553
Trace of beta_1
Density of beta_1
N = 4001 Bandwidth = 0.001959
Trace of gamma_2
Density of gamma_2
N = 4001 Bandwidth = 0.003523
Trace of omega_2
Density of omega_2
N = 4001 Bandwidth = 0.00417
Trace of alpha_2
Density of alpha_2
N = 4001 Bandwidth = 0.00142
Trace of beta_2
Density of beta_2
N = 4001 Bandwidth = 0.001799
Iterations
20000 30000 40000 50000 60000




```
## 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      4.88888 0.269710 4.264e-03      0.0133301
## gamma_1 1.03852 0.018003 2.846e-04      0.0008000
## omega_1 0.12373 0.019528 3.087e-04      0.0007748
## alpha_1 0.06625 0.007695 1.216e-04      0.0004013
## beta_1  0.90827 0.009709 1.535e-04      0.0004712
## gamma_2 0.95983 0.017460 2.760e-04      0.0007274
## omega_2 0.13877 0.020872 3.300e-04      0.0007678
## alpha_2 0.06315 0.007169 1.133e-04      0.0002868
## beta_2  0.91263 0.009008 1.424e-04      0.0003567
## a       0.03285 0.004406 6.965e-05      0.0002591
## b       0.95482 0.005065 8.007e-05      0.0003039
##
## 2. Quantiles for each variable:
##
##            2.5%     25%     50%     75%   97.5%
## nu      4.37223 4.70817 4.88912 5.06554 5.45340
## gamma_1 1.00208 1.02695 1.03872 1.05030 1.07384
## omega_1 0.08894 0.11020 0.12237 0.13583 0.16493
## alpha_1 0.05255 0.06081 0.06589 0.07116 0.08228
## beta_1  0.88798 0.90201 0.90864 0.91503 0.92569
## gamma_2 0.92634 0.94790 0.95930 0.97188 0.99383
## omega_2 0.10110 0.12443 0.13729 0.15213 0.18455
## alpha_2 0.05025 0.05821 0.06275 0.06764 0.07849
## beta_2  0.89358 0.90685 0.91312 0.91880 0.92930
## a       0.02515 0.02965 0.03252 0.03577 0.04216
## b       0.94402 0.95177 0.95528 0.95841 0.96328
```

```
# 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  2.033
## High_UB_S1    0.551
## Low_LB_B1    -0.551
## Low_LB_LBOP  -2.037
## High_omega    0.124
## High_alpha    0.066
## High_beta     0.908
## Low_omega     0.139
## Low_alpha     0.063
## Low_beta      0.913
```