# GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

20/11/2020

# Input

```
symbol = "PETR4.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: PETR4.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] "PETR4.SA"
```

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

## High and Low

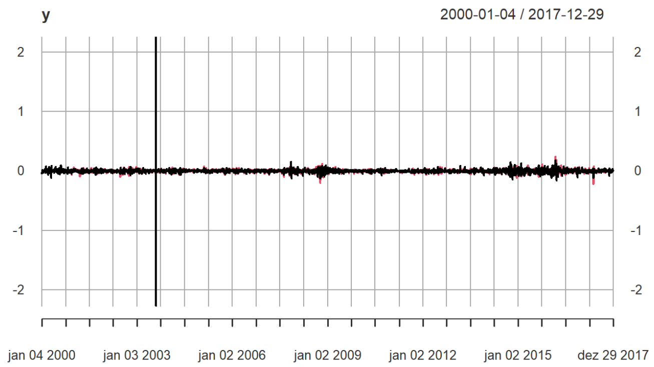
```
H <- Hi(x)
L <- Lo(x)
plot(cbind(H,L))
```


# Returns

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

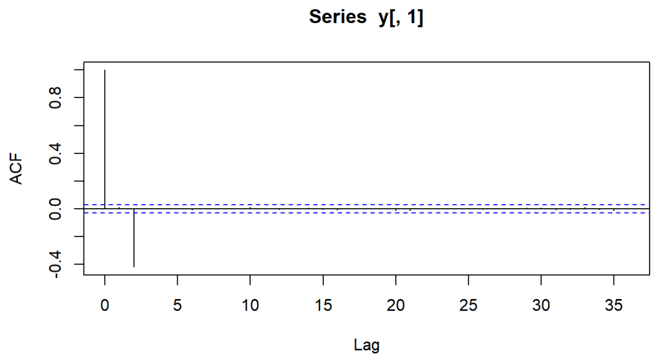
```
##               PETR4.SA.High PETR4.SA.Low
## PETR4.SA.High     1.0000000    0.3290882
## PETR4.SA.Low      0.3290882    1.0000000
```

```
plot(y)
```

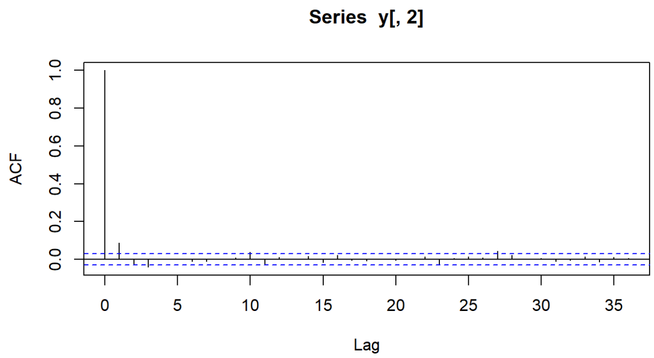



# Autocorrelation

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

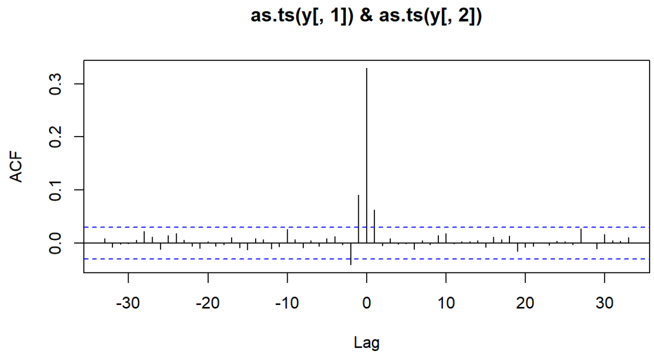



```
acf(y[,2])
```



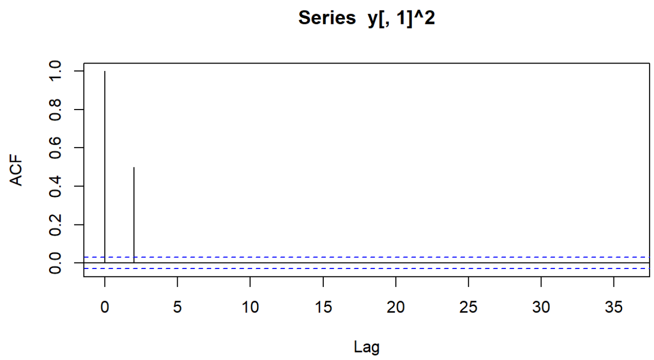

# Cross correlation

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

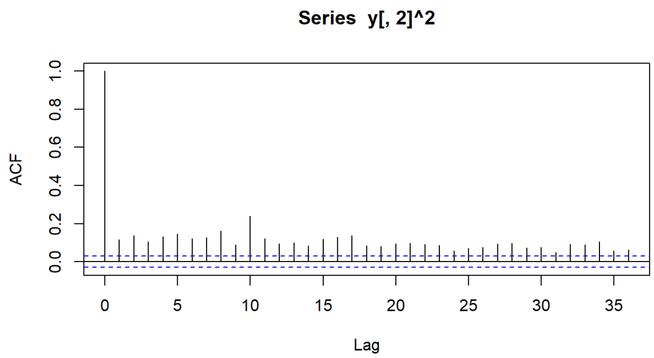



# Volatility verification

```
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
##   0.30   0.09   0.12   0.18   0.17   0.11   0.13   0.15   0.13   0.00   0.00
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.29   0.16   0.17   0.18   0.20   0.16   0.23   0.23   0.25   0.00   0.00
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.29   0.16   0.19   0.16   0.15   0.15   0.20   0.24   0.24   0.00   0.00
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.30   0.17   0.21   0.20   0.29   0.29   0.24   0.22   0.25   0.01   0.00
## 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   0.16   0.19   0.19   0.26   0.27   0.20   0.22   0.25   0.01   0.01
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.30   0.17   0.19   0.17   0.28   0.31   0.25   0.24   0.25   0.01   0.01
## 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: 0.29
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

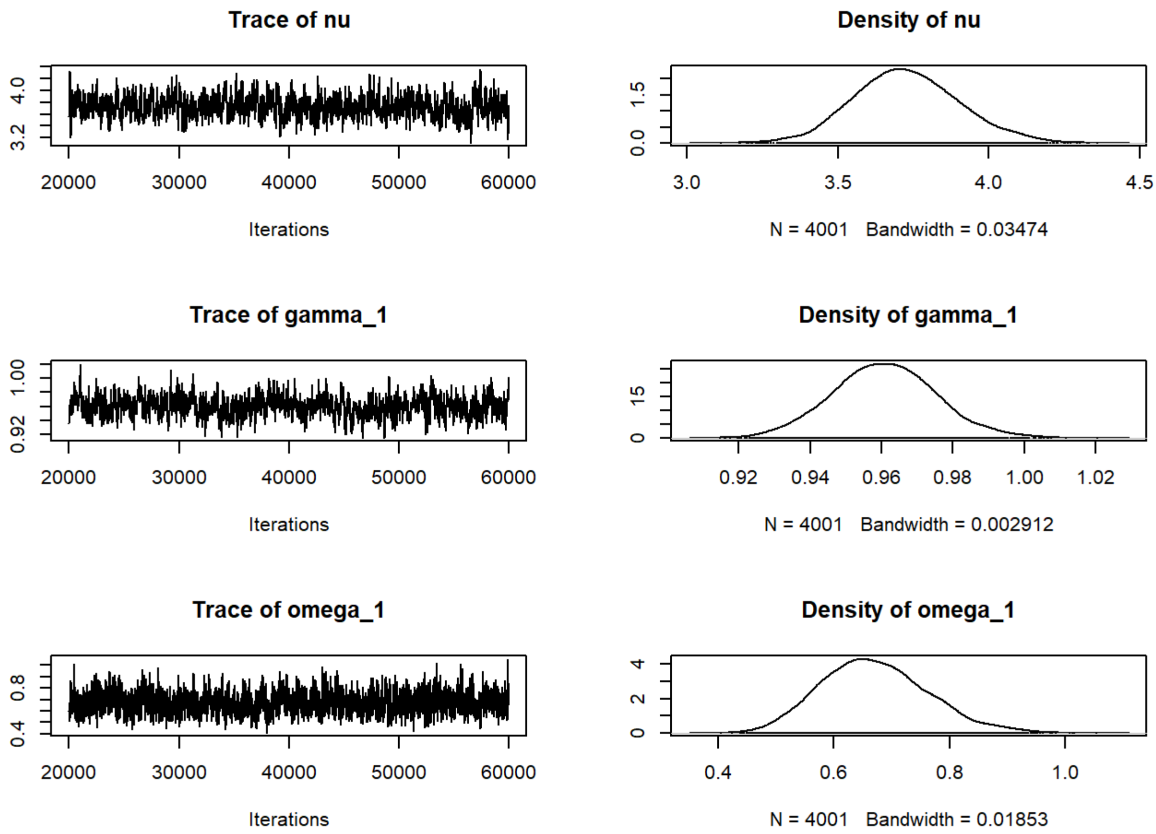
```
out2 <- increaseSim(out, nSim=50000)
```

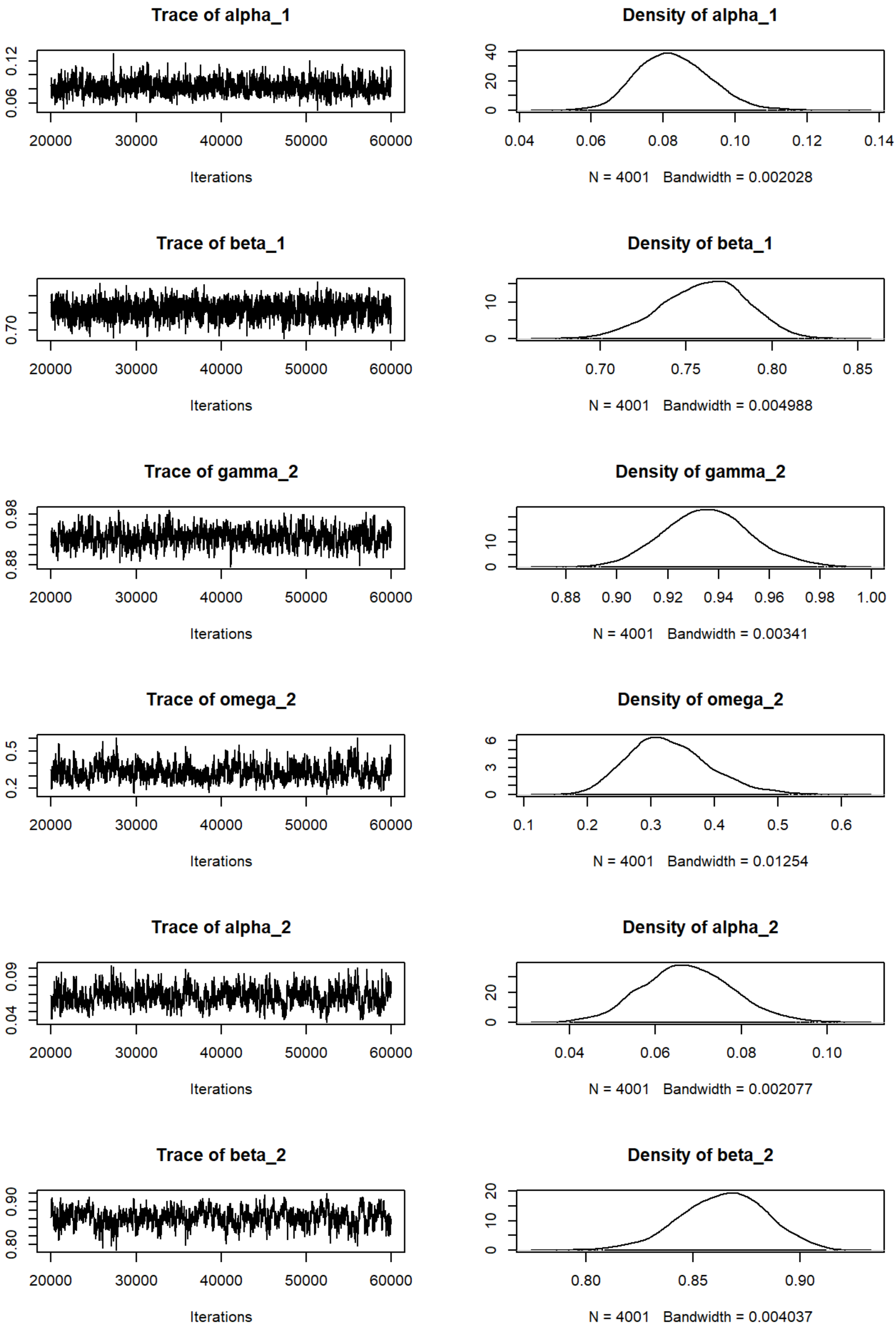
```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.27
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

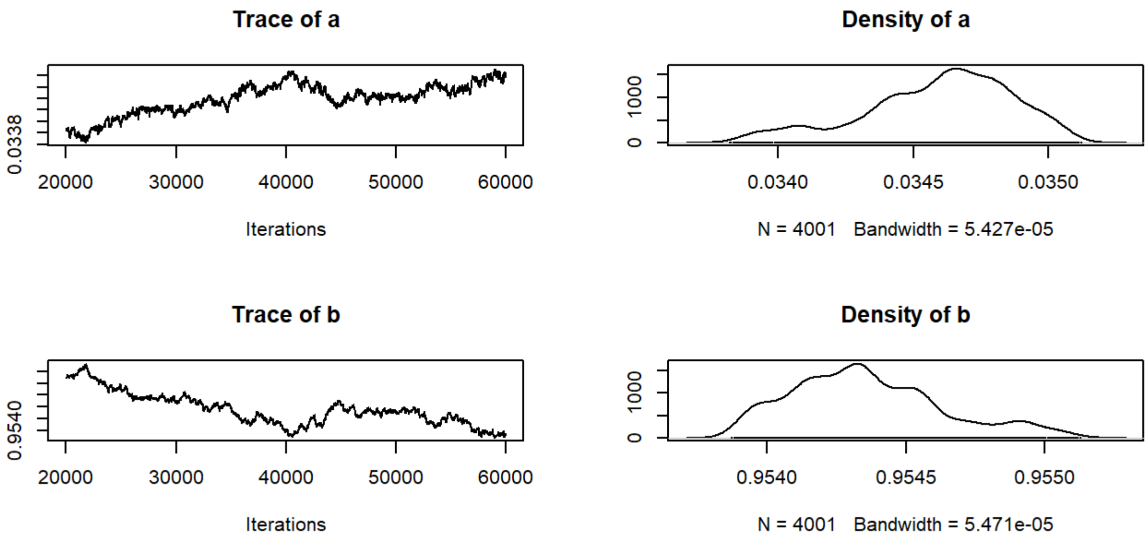
```
out <- window(out2, start=20000, thin=10)
rm(out2)

end <- Sys.time()

# elapsed time
end-start
```

```
## Time difference of 6.452315 mins
```

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




Trace of alpha_1
Iterations
Density of alpha_1
N = 4001 Bandwidth = 0.002028
Trace of beta_1
Iterations
Density of beta_1
N = 4001 Bandwidth = 0.004988
Trace of gamma_2
Iterations
Density of gamma_2
N = 4001 Bandwidth = 0.00341
Trace of omega_2
Iterations
Density of omega_2
N = 4001 Bandwidth = 0.01254
Trace of alpha_2
Iterations
Density of alpha_2
N = 4001 Bandwidth = 0.002077
Trace of beta_2
Iterations
Density of beta_2
N = 4001 Bandwidth = 0.004037




```
## 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      3.72786 0.1738106 2.748e-03      0.0079330
## gamma_1 0.96075 0.0145047 2.293e-04      0.0007069
## omega_1 0.67070 0.0918512 1.452e-03      0.0032479
## alpha_1 0.08292 0.0100531 1.589e-04      0.0003438
## beta_1  0.76062 0.0247224 3.908e-04      0.0007499
## gamma_2 0.93553 0.0168998 2.672e-04      0.0007781
## omega_2 0.32835 0.0641803 1.015e-03      0.0040590
## alpha_2 0.06754 0.0103471 1.636e-04      0.0006030
## beta_2  0.86419 0.0200059 3.163e-04      0.0014200
## a       0.03459 0.0002724 4.307e-06      0.0001127
## b       0.95436 0.0002712 4.287e-06      0.0001387
##
## 2. Quantiles for each variable:
##
##             2.5%     25%     50%     75%   97.5%
## nu      3.40332 3.60974 3.72106 3.84047 4.08707
## gamma_1 0.93259 0.95090 0.96050 0.97024 0.99044
## omega_1 0.50918 0.60582 0.66398 0.72930 0.87023
## alpha_1 0.06519 0.07569 0.08232 0.08949 0.10382
## beta_1  0.70973 0.74424 0.76191 0.77775 0.80518
## gamma_2 0.90377 0.92389 0.93535 0.94655 0.97007
## omega_2 0.21846 0.28414 0.32168 0.36740 0.47210
## alpha_2 0.04815 0.06057 0.06731 0.07437 0.08872
## beta_2  0.82251 0.85097 0.86515 0.87828 0.90117
## a       0.03395 0.03443 0.03463 0.03479 0.03503
## b       0.95393 0.95416 0.95433 0.95453 0.95497
```

```
# 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  1.894
## High_UB_S1    0.518
## Low_LB_B1    -0.491
## Low_LB_LBOP  -2.027
## High_omega    0.671
## High_alpha    0.083
## High_beta     0.761
## Low_omega     0.328
## Low_alpha     0.068
## Low_beta      0.864
```