GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

20/11/2020

Input

```
symbol = "ITSA4.SA"#"BOVA11.SA"
from=as.Date('2012-06-18')#2012
to=as.Date('2017-12-31')#'2018-12-31'
C_Trend = 0.95
C_Reaction = 0.50
```

Data download

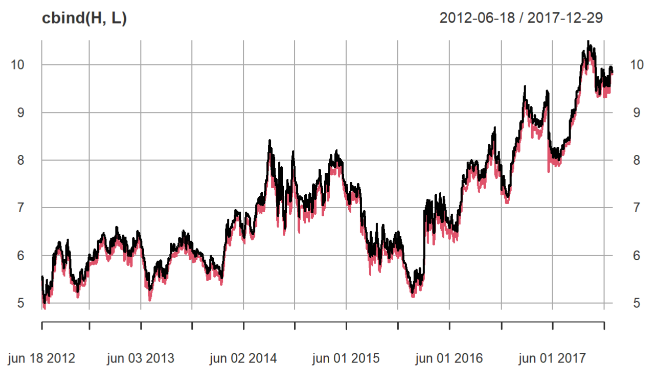
```
getSymbols.yahoo(symbol, from=from, to=to, env=globalenv())
```

```
## [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))
```

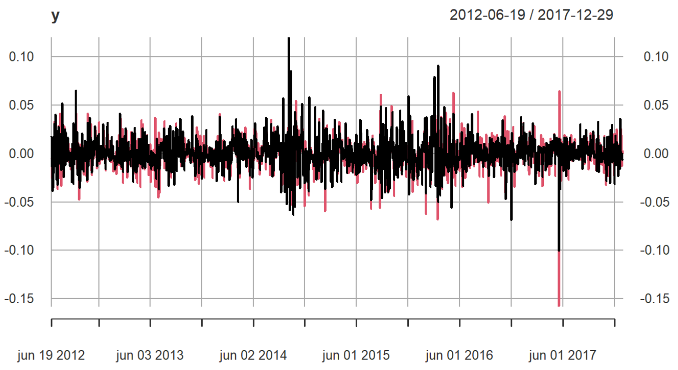


Returns

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

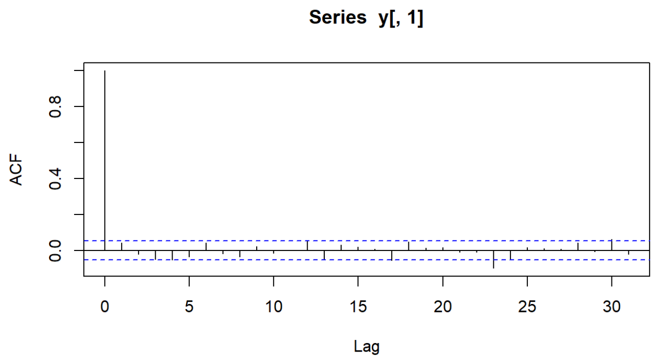
```
##              ITSA4.SA.High ITSA4.SA.Low
## ITSA4.SA.High     1.0000000    0.7070102
## ITSA4.SA.Low      0.7070102    1.0000000
```

```
plot(y)
```

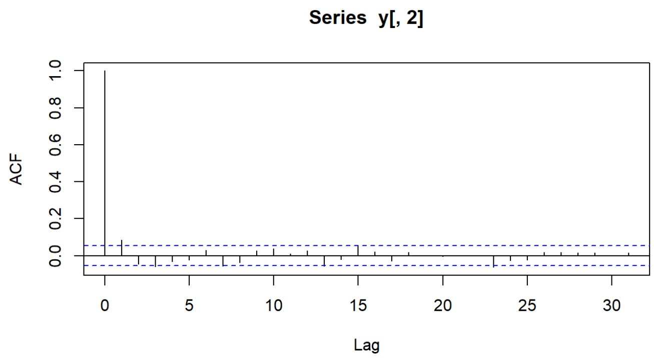


Autocorrelation

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

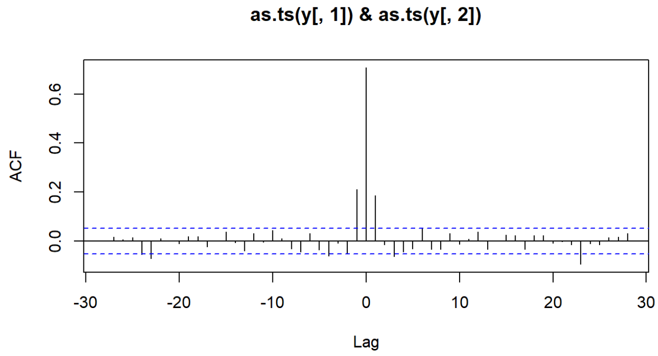



```
acf(y[,2])
```



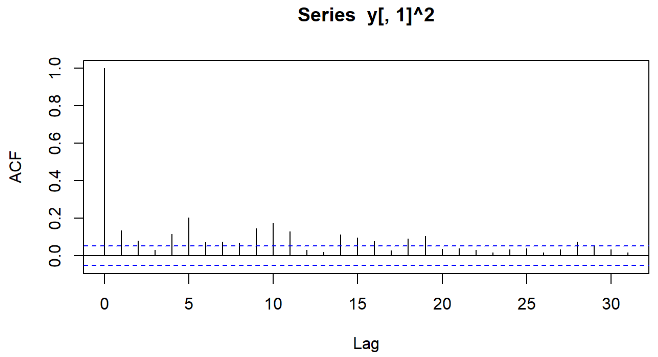
Cross correlation

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

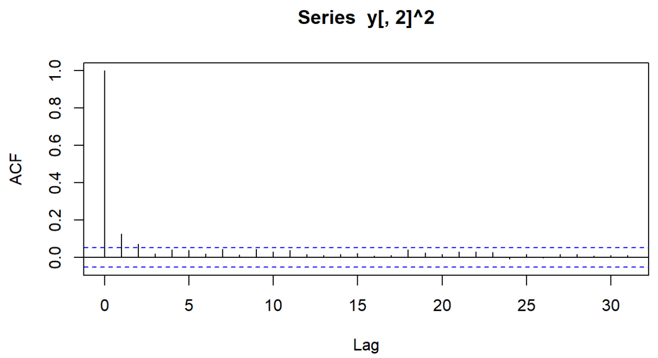


Volatility verification

```
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
##   0.49   0.20   0.25   0.22   0.23   0.20   0.22   0.24   0.23   0.47   0.61
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.50   0.21   0.21   0.23   0.25   0.20   0.22   0.24   0.26   0.49   0.50
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.39   0.18   0.26   0.19   0.25   0.18   0.19   0.23   0.25   0.44   0.43
## 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: 0.48
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

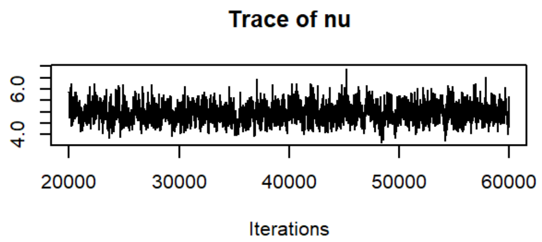
```
out2 <- increaseSim(out, nSim=50000)
```

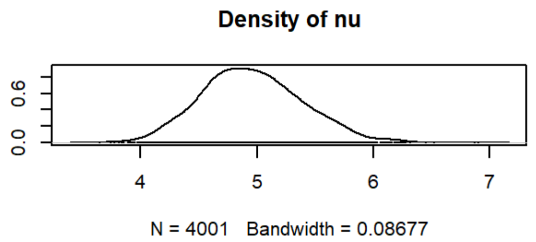
```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.52
## lambda: 0.48
## Accept Rate: 0.44
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

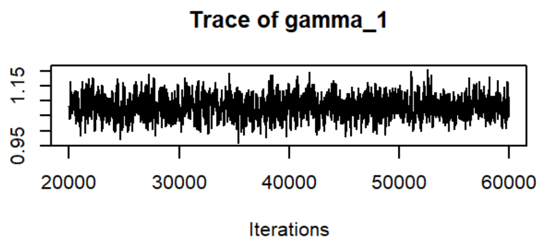
```
out <- window(out2, start=20000, thin=10)
rm(out2)

end <- Sys.time()

# elapsed time
end-start
```

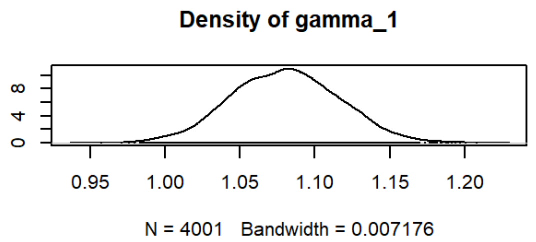
```
## Time difference of 1.641824 mins
```

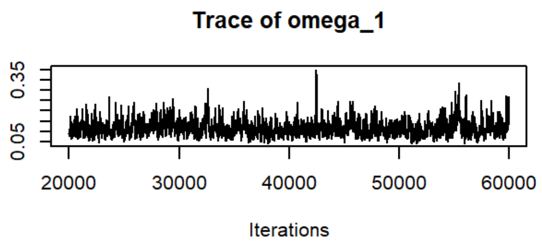
```
# plot Markov Chain
plot(out$MC)
```

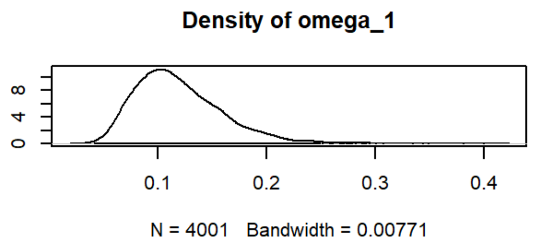

Trace of nu
4.0
6.0
20000
30000
40000
50000
60000
Iterations

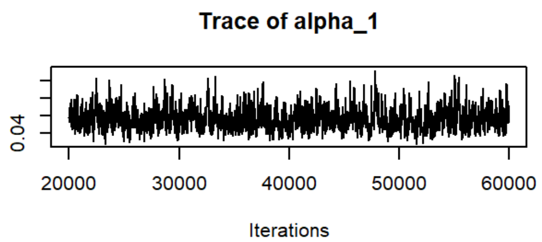

Density of nu
0.0
0.6
4
5
6
7
N = 4001 Bandwidth = 0.08677

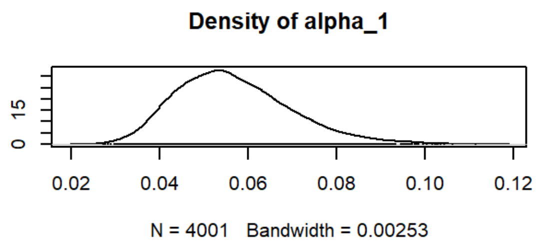

Trace of gamma_1
0.95
1.15
20000
30000
40000
50000
60000
Iterations

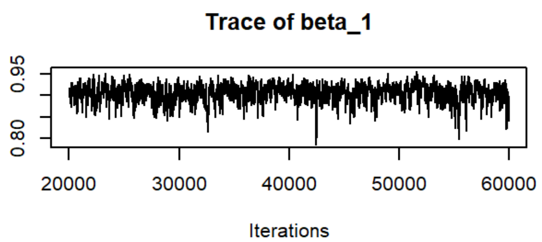

Density of gamma_1
0
4
8
0.95
1.00
1.05
1.10
1.15
1.20
N = 4001 Bandwidth = 0.007176

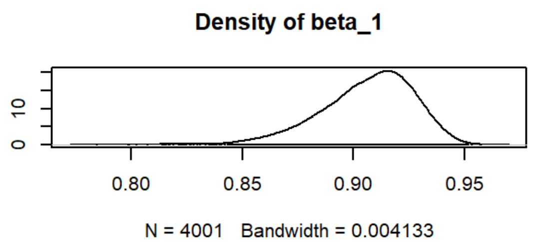

Trace of omega_1
0.05
0.35
20000
30000
40000
50000
60000
Iterations

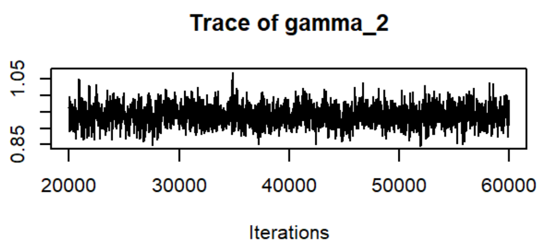

Density of omega_1
0
4
8
0.1
0.2
0.3
0.4
N = 4001 Bandwidth = 0.00771

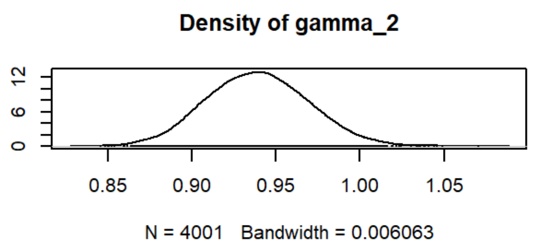

Trace of alpha_1
0.04
20000
30000
40000
50000
60000
Iterations

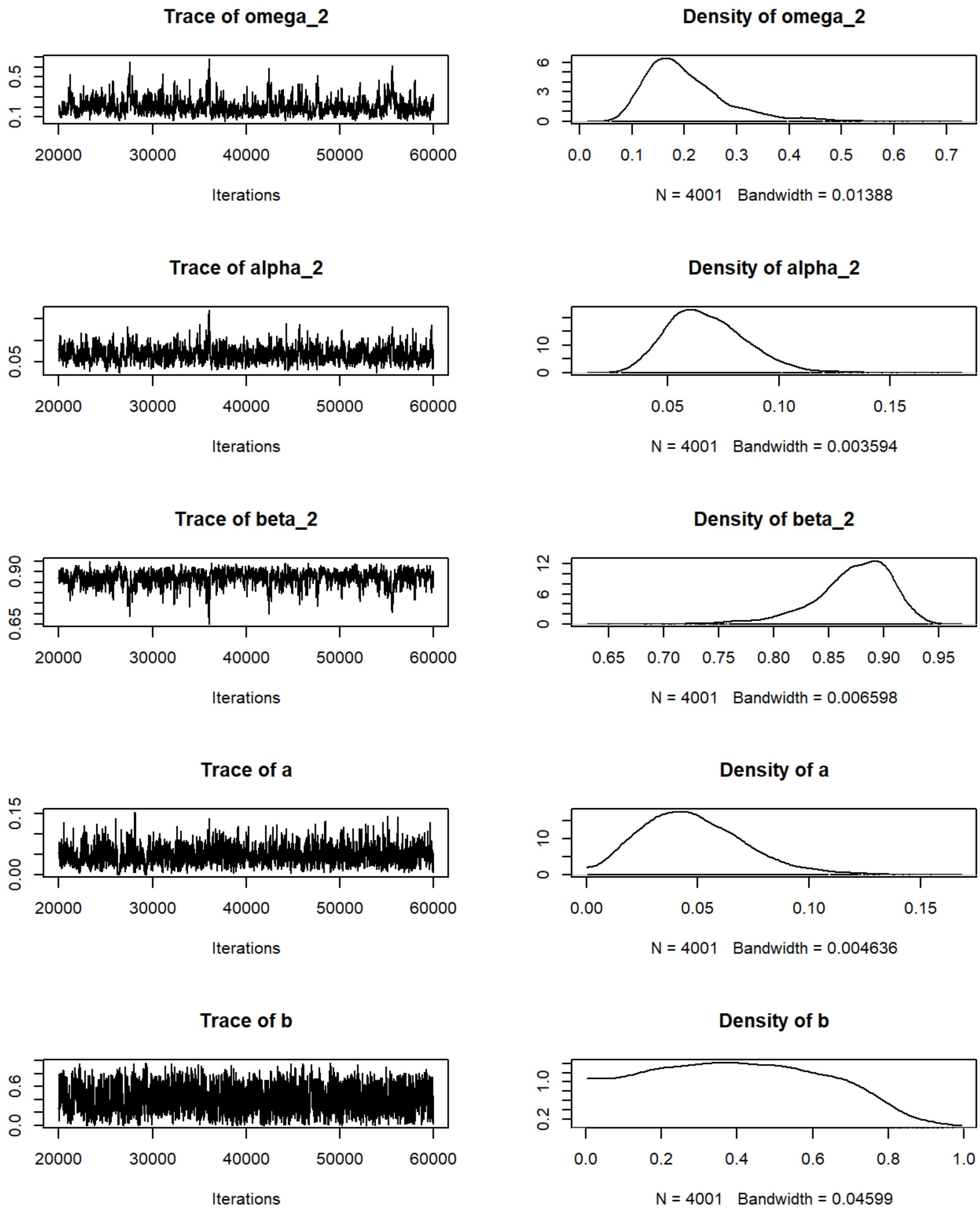

Density of alpha_1
0
15
0.02
0.04
0.06
0.08
0.10
0.12
N = 4001 Bandwidth = 0.00253


Trace of beta_1
0.80
0.95
20000
30000
40000
50000
60000
Iterations


Density of beta_1
0
10
0.80
0.85
0.90
0.95
N = 4001 Bandwidth = 0.004133


Trace of gamma_2
0.85
1.05
20000
30000
40000
50000
60000
Iterations


Density of gamma_2
0
6
12
0.85
0.90
0.95
1.00
1.05
N = 4001 Bandwidth = 0.006063




```
## 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      4.95693 0.43001 0.0067983      0.0155184
## gamma_1 1.08018 0.03556 0.0005622      0.0011795
## omega_1 0.11902 0.04023 0.0006360      0.0019594
## alpha_1 0.05674 0.01272 0.0002010      0.0005628
## beta_1  0.90574 0.02142 0.0003386      0.0010548
## gamma_2 0.93908 0.03005 0.0004751      0.0009442
## omega_2 0.20304 0.08027 0.0012690      0.0045725
## alpha_2 0.06774 0.01781 0.0002816      0.0007658
## beta_2  0.87199 0.03641 0.0005756      0.0019450
## a       0.04760 0.02297 0.0003632      0.0007690
## b       0.40386 0.22793 0.0036034      0.0077452
##
## 2. Quantiles for each variable:
##
##            2.5%     25%     50%     75%   97.5%
## nu      4.18220 4.65700 4.93108 5.23847 5.83386
## gamma_1 1.01104 1.05523 1.08011 1.10403 1.14983
## omega_1 0.06105 0.09024 0.11244 0.14145 0.21143
## alpha_1 0.03635 0.04757 0.05518 0.06437 0.08577
## beta_1  0.85788 0.89351 0.90886 0.92096 0.93912
## gamma_2 0.88182 0.91824 0.93875 0.95891 0.99929
## omega_2 0.09619 0.14700 0.18560 0.23919 0.41900
## alpha_2 0.03855 0.05479 0.06582 0.07872 0.10615
## beta_2  0.78081 0.85387 0.87745 0.89768 0.92611
## a       0.01025 0.03087 0.04547 0.06205 0.09972
## b       0.02200 0.21584 0.39581 0.58143 0.81713
```

```
# 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  2.077
## High_UB_S1    0.545
## Low_LB_B1    -0.548
## Low_LB_LBOP  -2.061
## High_omega    0.119
## High_alpha    0.057
## High_beta     0.906
## Low_omega     0.203
## Low_alpha     0.068
## Low_beta      0.872
```