GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

20/11/2020

Input

```
symbol = "GE"#"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

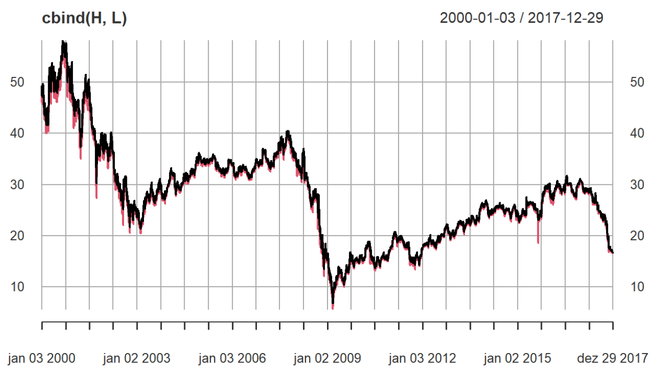
```
getSymbols.yahoo(symbol, from=from, to=to,env=globalenv())
```

```
## [1] "GE"
```

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

High and Low

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

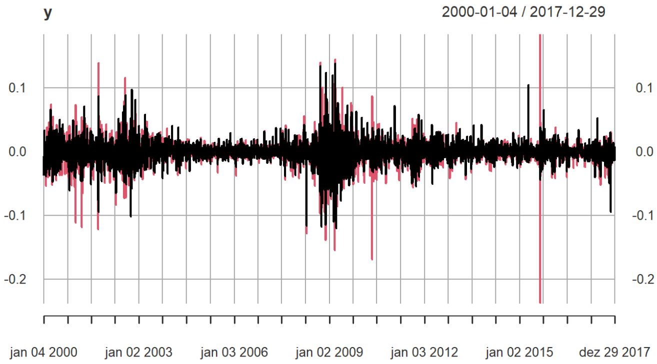


Returns

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

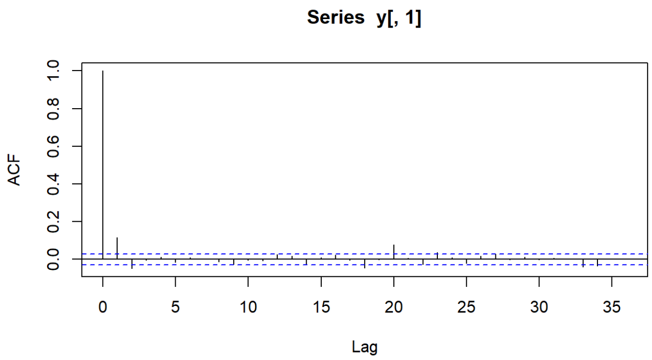
```
##           GE.High    GE.Low
## GE.High 1.0000000 0.6933611
## GE.Low  0.6933611 1.0000000
```

```
plot(y)
```

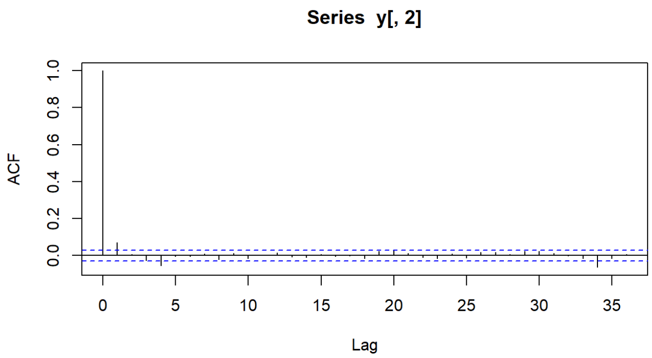


Autocorrelation

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

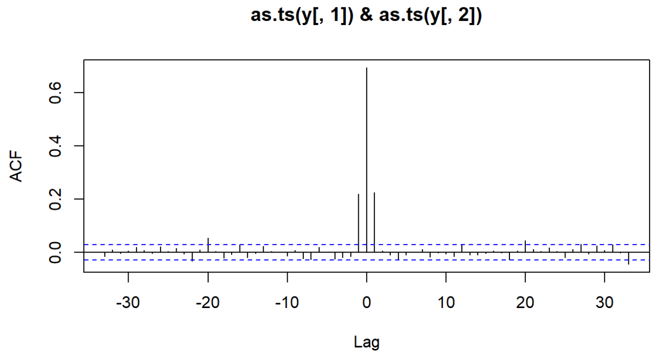



```
acf(y[,2])
```



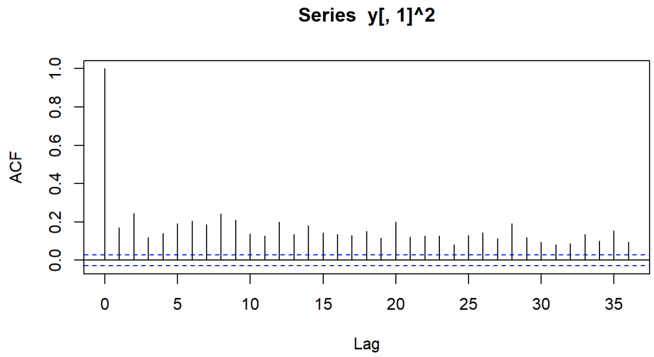
Cross correlation

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

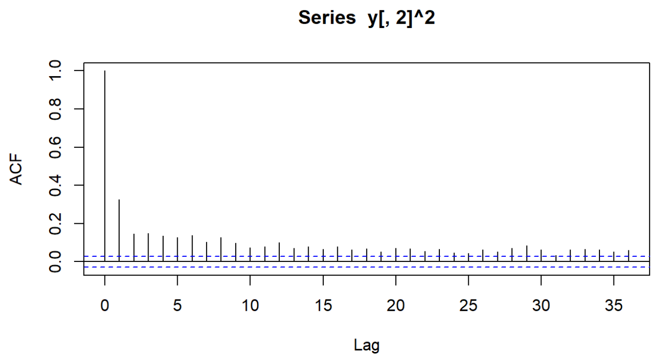


Volatility verification

```
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
##   0.32   0.10   0.23   0.04   0.05   0.11   0.19   0.07   0.10   0.26   0.35
## 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   0.17   0.20   0.08   0.14   0.18   0.22   0.15   0.20   0.28   0.37
## 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   0.16   0.23   0.10   0.18   0.16   0.18   0.13   0.19   0.26   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
##   0.30   0.18   0.23   0.19   0.18   0.19   0.21   0.20   0.17   0.26   0.33
## 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: 0.34
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

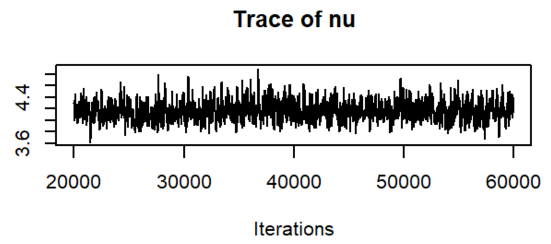
```
out2 <- increaseSim(out, nSim=50000)
```

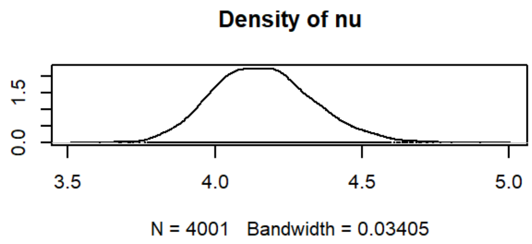
```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.33
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

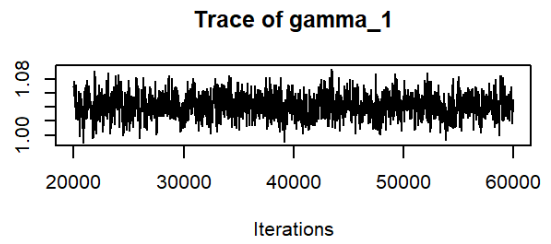
```
out <- window(out2, start=20000, thin=10)
rm(out2)

end <- Sys.time()

# elapsed time
end-start
```

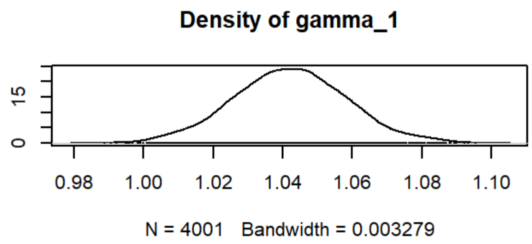
```
## Time difference of 4.989913 mins
```

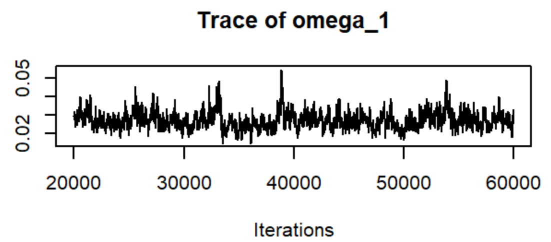
```
# plot Markov Chain
plot(out$MC)
```

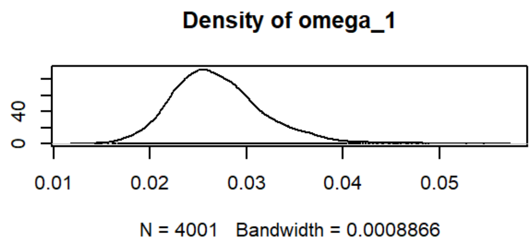

Trace of nu
3.6
4.4
20000 30000 40000 50000 60000
Iterations

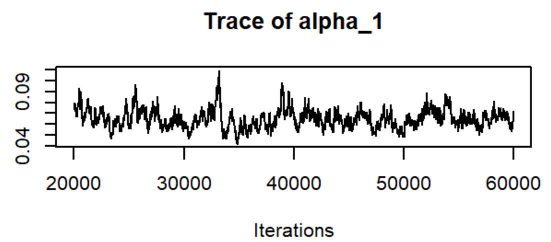

Density of nu
0.0
1.5
3.5 4.0 4.5 5.0
N = 4001 Bandwidth = 0.03405

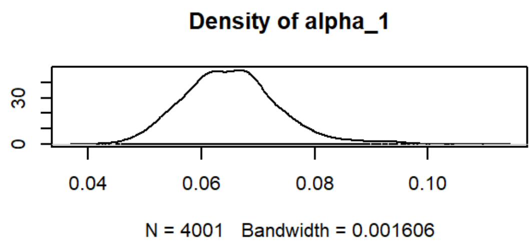

Trace of gamma_1
1.00
1.08
20000 30000 40000 50000 60000
Iterations

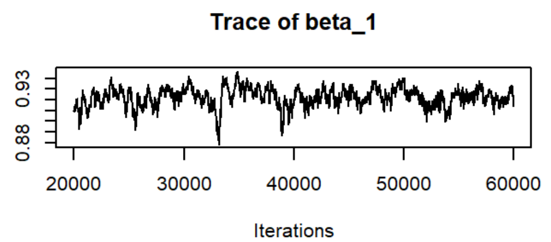

Density of gamma_1
0
15
0.98 1.00 1.02 1.04 1.06 1.08 1.10
N = 4001 Bandwidth = 0.003279

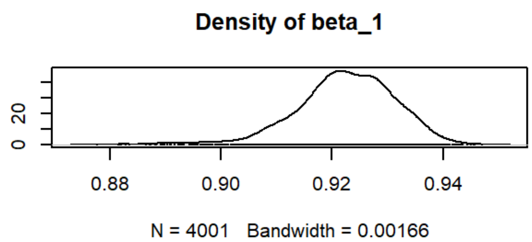

Trace of omega_1
0.02
0.05
20000 30000 40000 50000 60000
Iterations

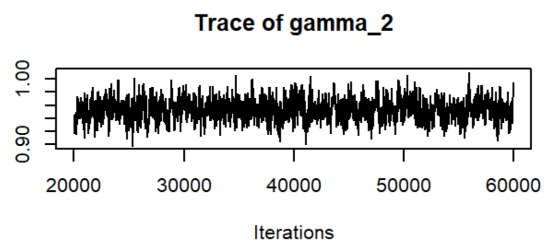

Density of omega_1
0
40
0.01 0.02 0.03 0.04 0.05
N = 4001 Bandwidth = 0.0008866

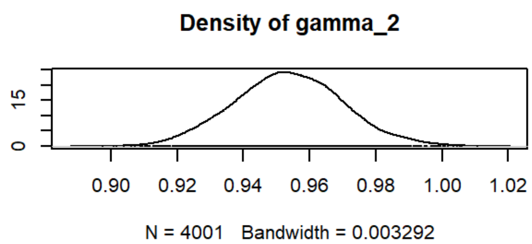

Trace of alpha_1
0.04
0.09
20000 30000 40000 50000 60000
Iterations

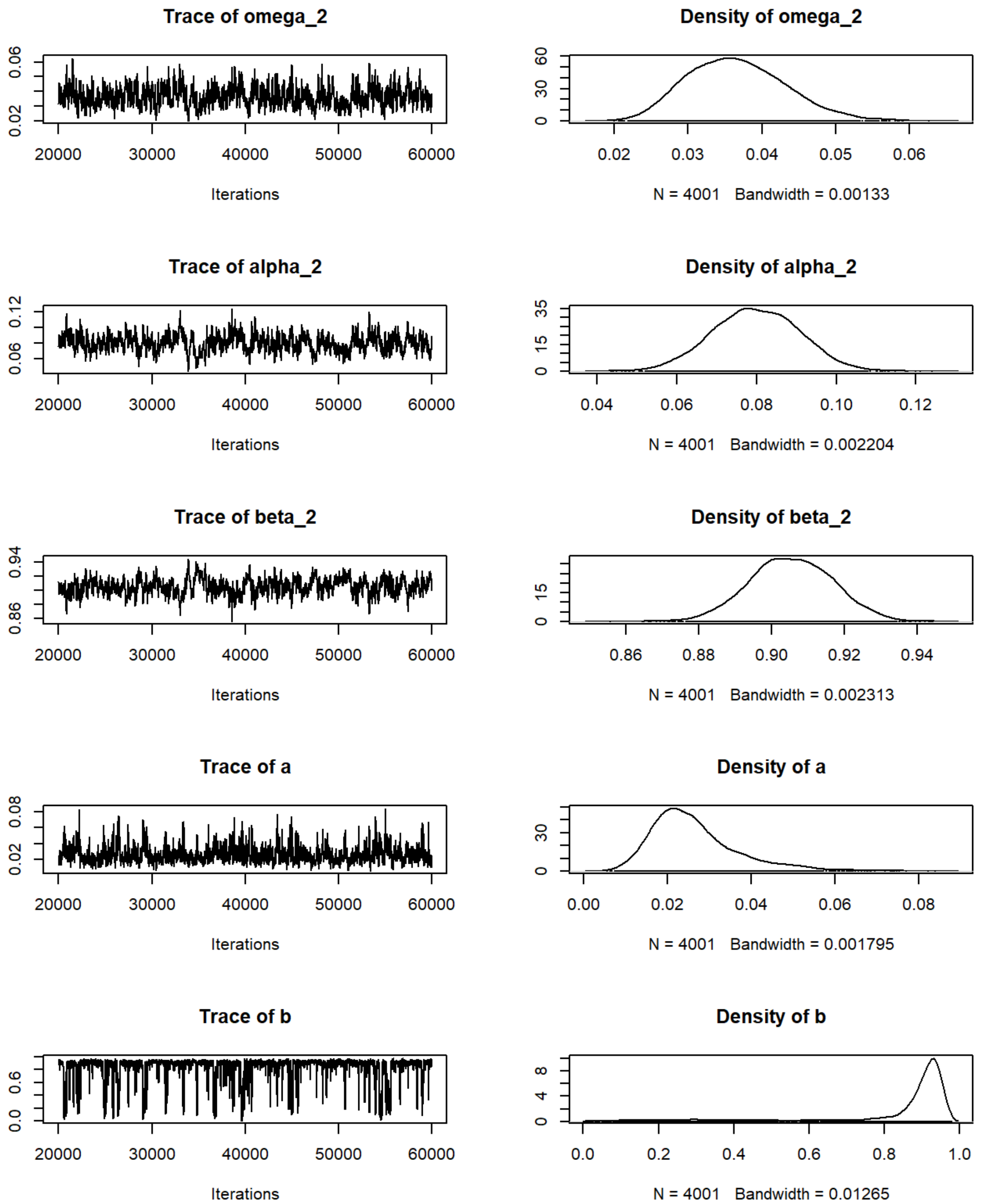

Density of alpha_1
0
30
0.04 0.06 0.08 0.10
N = 4001 Bandwidth = 0.001606


Trace of beta_1
0.88
0.93
20000 30000 40000 50000 60000
Iterations


Density of beta_1
0
20
0.88 0.90 0.92 0.94
N = 4001 Bandwidth = 0.00166


Trace of gamma_2
0.90
1.00
20000 30000 40000 50000 60000
Iterations


Density of gamma_2
0
15
0.90 0.92 0.94 0.96 0.98 1.00 1.02
N = 4001 Bandwidth = 0.003292




```
## 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      4.16357 0.168733 2.668e-03      0.0071212
## gamma_1 1.04212 0.016291 2.576e-04      0.0006066
## omega_1 0.02703 0.004792 7.576e-05      0.0003382
## alpha_1 0.06525 0.008582 1.357e-04      0.0010267
## beta_1  0.92244 0.008923 1.411e-04      0.0010954
## gamma_2 0.95351 0.016320 2.580e-04      0.0006766
## omega_2 0.03657 0.006590 1.042e-04      0.0003843
## alpha_2 0.07984 0.010924 1.727e-04      0.0007961
## beta_2  0.90539 0.011461 1.812e-04      0.0008288
## a       0.02642 0.010592 1.675e-04      0.0005501
## b       0.81608 0.224545 3.550e-03      0.0156555
##
## 2. Quantiles for each variable:
##
##            2.5%     25%     50%     75%   97.5%
## nu      3.85681 4.04371 4.15685 4.27038 4.52015
## gamma_1 1.00971 1.03123 1.04206 1.05300 1.07532
## omega_1 0.01906 0.02375 0.02642 0.02963 0.03787
## alpha_1 0.05033 0.05943 0.06480 0.07009 0.08472
## beta_1  0.90274 0.91743 0.92283 0.92845 0.93774
## gamma_2 0.92171 0.94255 0.95335 0.96441 0.98675
## omega_2 0.02516 0.03171 0.03616 0.04090 0.05053
## alpha_2 0.05865 0.07227 0.07967 0.08730 0.10094
## beta_2  0.88315 0.89769 0.90534 0.91325 0.92728
## a       0.01138 0.01922 0.02438 0.03114 0.05319
## b       0.13206 0.85113 0.91018 0.93514 0.96191
```

```r
# 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  2.020
## High_UB_S1    0.522
## Low_LB_B1    -0.521
## Low_LB_LBOP  -2.028
## High_omega    0.027
## High_alpha    0.065
## High_beta     0.922
## Low_omega     0.037
## Low_alpha     0.080
## Low_beta      0.905
```