GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

20/11/2020

Input

```
symbol = "^GSPC"#"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

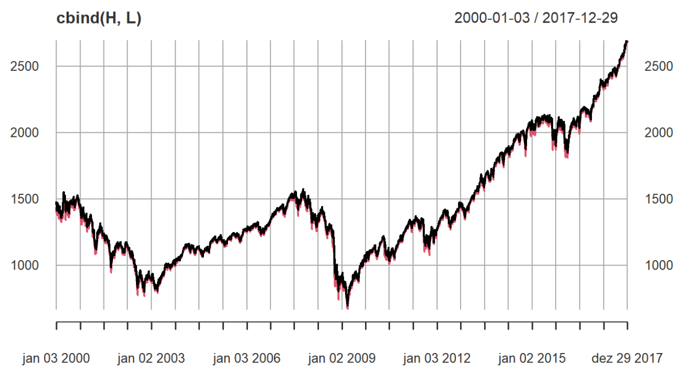
```
getSymbols.yahoo(symbol, from=from, to=to,env=globalenv())
```

```
## [1] "^GSPC"
```

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

High and Low

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

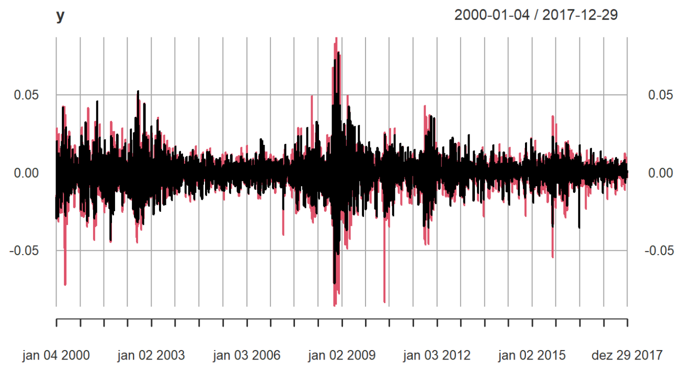


Returns

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

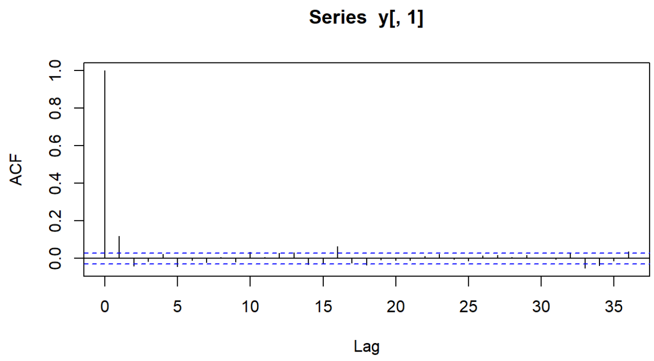
```
##           GSPC.High  GSPC.Low
## GSPC.High 1.0000000 0.6716581
## GSPC.Low  0.6716581 1.0000000
```

```
plot(y)
```

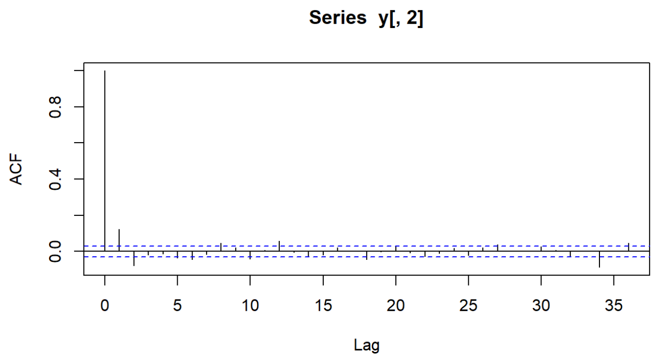


Autocorrelation

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

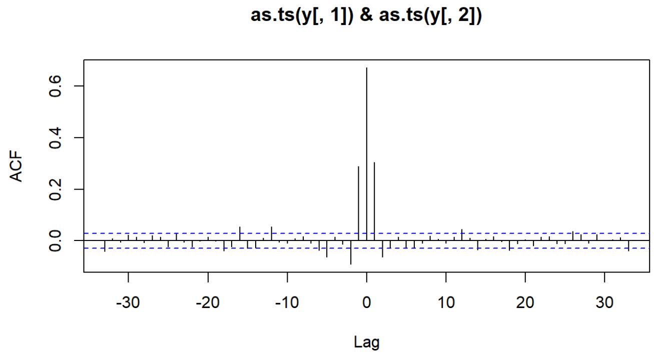



```
acf(y[,2])
```



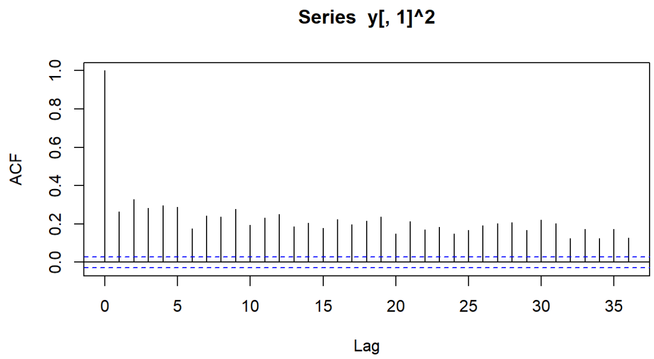
Cross correlation

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

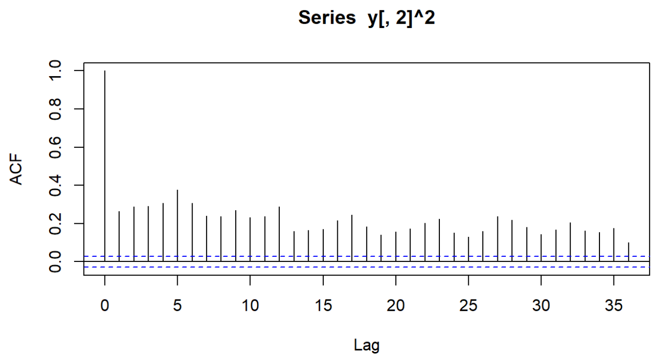


Volatility verification

```
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.
```

```
## Warning in if (class(control$cholCov) != "try-error") {: a condição tem
## comprimento > 1 e somente o primeiro elemento será usado
```

```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.18
## lambda: 0.32
## Accept Rate: 0.25
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

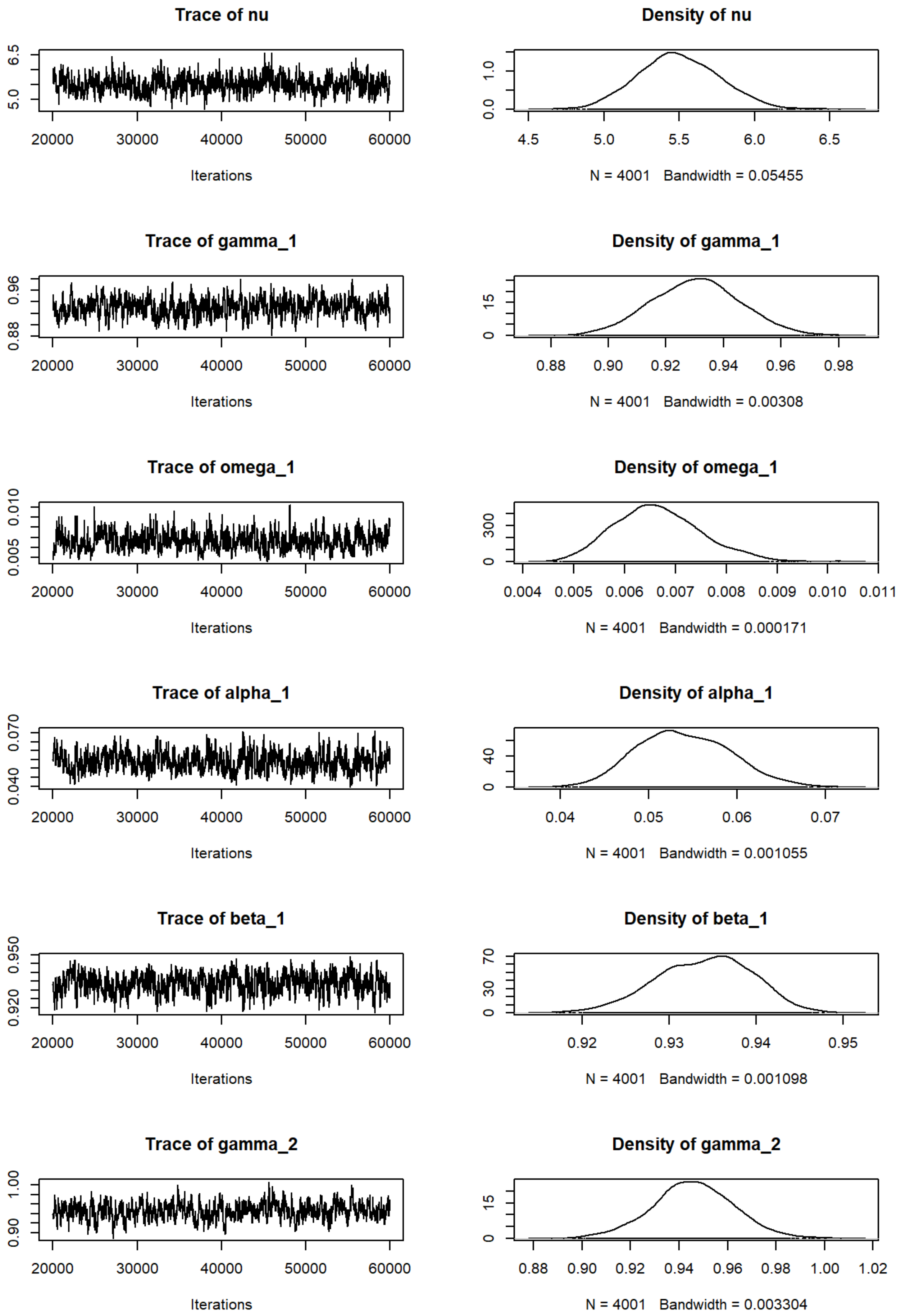
```
out2 <- increaseSim(out, nSim=50000)
```

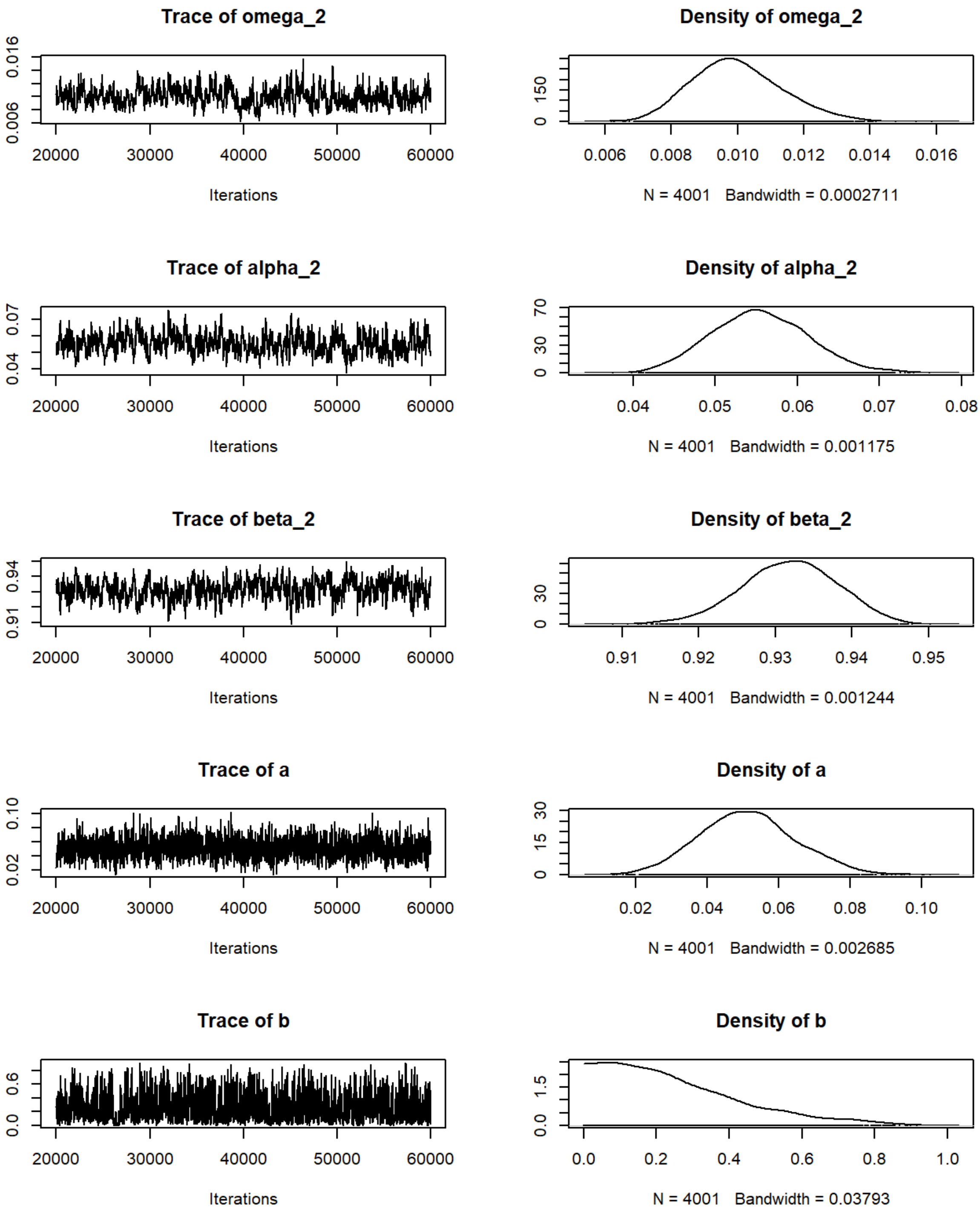
```
## Calibrating the Lambda coefficient:
## lambda: 0.32
## Accept Rate: 0.22
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

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

end <- Sys.time()

# elapsed time
end-start
```

```
## Time difference of 2.417842 mins
```

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


Trace of nu
5.0
6.5
20000
30000
40000
50000
60000
Iterations
Density of nu
0.0
1.0
4.5
5.0
5.5
6.0
6.5
N = 4001 Bandwidth = 0.05455
Trace of gamma_1
0.88
0.96
Iterations
Density of gamma_1
0
15
0.88
0.90
0.92
0.94
0.96
0.98
N = 4001 Bandwidth = 0.00308
Trace of omega_1
0.005
0.010
Iterations
Density of omega_1
0
300
0.004
0.005
0.006
0.007
0.008
0.009
0.010
0.011
N = 4001 Bandwidth = 0.000171
Trace of alpha_1
0.040
0.070
Iterations
Density of alpha_1
0
40
0.04
0.05
0.06
0.07
N = 4001 Bandwidth = 0.001055
Trace of beta_1
0.920
0.950
Iterations
Density of beta_1
0
30
70
0.92
0.93
0.94
0.95
N = 4001 Bandwidth = 0.001098
Trace of gamma_2
0.90
1.00
Iterations
Density of gamma_2
0
15
0.88
0.90
0.92
0.94
0.96
0.98
1.00
1.02
N = 4001 Bandwidth = 0.003304




```
## 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      5.496798 0.2703590 4.274e-03      1.558e-02
## gamma_1 0.930474 0.0152653 2.413e-04      9.833e-04
## omega_1 0.006656 0.0008493 1.343e-05      5.146e-05
## alpha_1 0.053950 0.0052290 8.267e-05      3.115e-04
## beta_1  0.934015 0.0054409 8.602e-05      3.241e-04
## gamma_2 0.945731 0.0168046 2.657e-04      1.173e-03
## omega_2 0.009981 0.0013435 2.124e-05      9.564e-05
## alpha_2 0.055439 0.0058240 9.207e-05      3.718e-04
## beta_2  0.931812 0.0061671 9.750e-05      3.850e-04
## a       0.051306 0.0133479 2.110e-04      4.024e-04
## b       0.250619 0.1879662 2.972e-03      6.299e-03
##
## 2. Quantiles for each variable:
##
##             2.5%      25%      50%      75%   97.5%
## nu      4.991876 5.312603 5.482831 5.675017 6.03404
## gamma_1 0.900632 0.919792 0.930639 0.940554 0.96034
## omega_1 0.005157 0.006052 0.006603 0.007188 0.00846
## alpha_1 0.044437 0.050147 0.053659 0.057620 0.06457
## beta_1  0.922970 0.930180 0.934364 0.937927 0.94369
## gamma_2 0.911626 0.935015 0.945560 0.956958 0.97916
## omega_2 0.007585 0.009042 0.009881 0.010854 0.01281
## alpha_2 0.044482 0.051338 0.055268 0.059390 0.06714
## beta_2  0.919110 0.927687 0.931956 0.936126 0.94330
## a       0.026048 0.042049 0.050915 0.059881 0.07818
## b       0.010671 0.099590 0.211239 0.357412 0.72333
```

```
# 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  1.913
## High_UB_S1    0.591
## Low_LB_B1    -0.564
## Low_LB_LBOP  -2.057
## High_omega    0.007
## High_alpha    0.054
## High_beta     0.934
## Low_omega     0.010
## Low_alpha     0.055
## Low_beta      0.932
```