GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

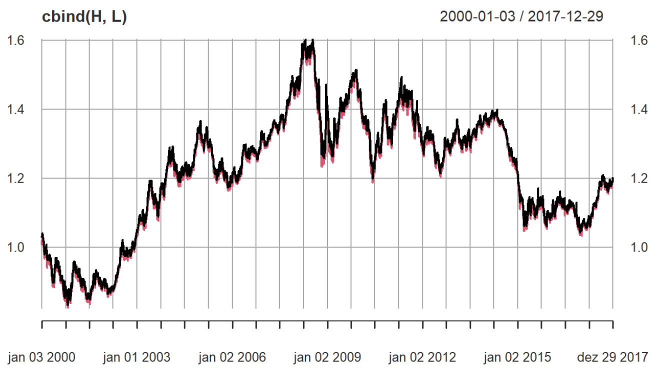
20/11/2020

Input

```
x<- read.csv("~/R/EURUSD/EURUSD.csv", header=F)
colnames(x)=c("Date", "Open","High","Low", "Close","Volume","Adjusted")
x$Date=as.Date(gsub("[.]","-",x$Date))
row.names(x)=x$Date
x=xts(x[,-1],order.by = x[,1])
C_Trend = 0.95
C_Reaction = 0.50
```

High and Low

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

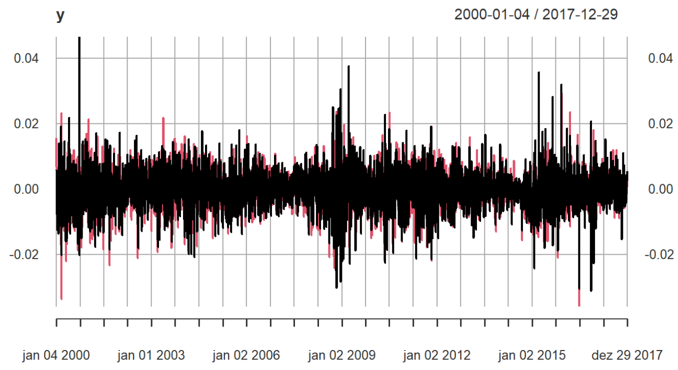


Returns

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

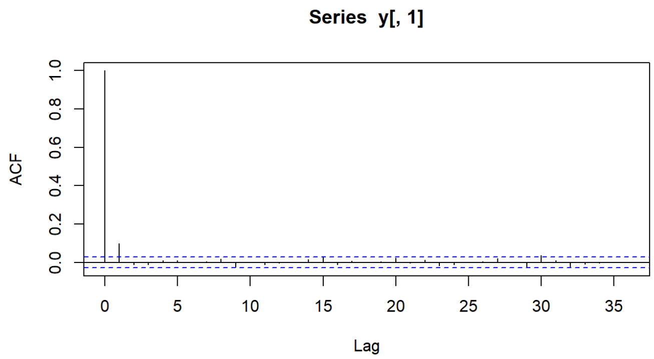
```
##           High       Low
## High 1.0000000 0.5579326
## Low  0.5579326 1.0000000
```

```
plot(y)
```

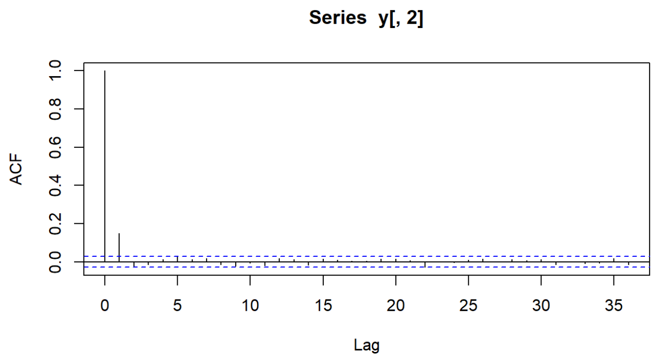


Autocorrelation

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

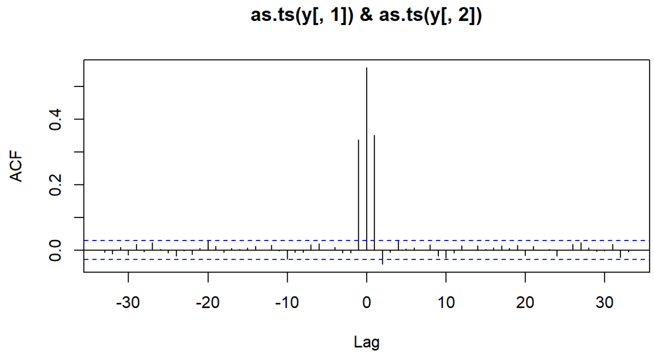



```
acf(y[,2])
```



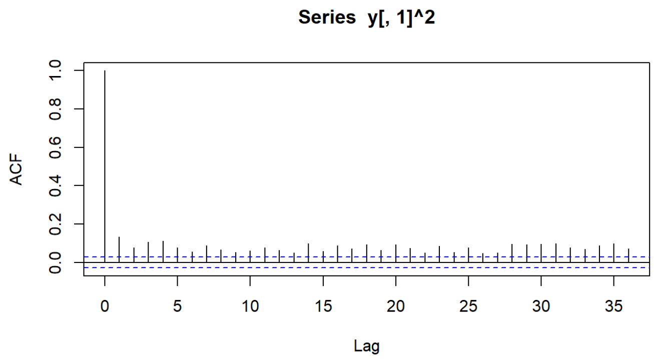
Cross correlation

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

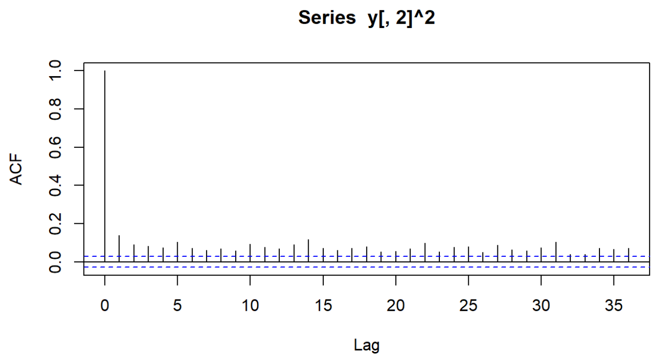


Volatility verification

```
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
##   0.33   0.10   0.21   0.09   0.12   0.10   0.22   0.09   0.10   0.26   0.54
## Accept Rate:
##  phi_1  phi_2  phi_3  phi_4  phi_5  phi_6  phi_7  phi_8  phi_9 phi_10 phi_11
##   0.37   0.19   0.22   0.14   0.16   0.18   0.23   0.12   0.17   0.29   0.42
## 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   0.15   0.17   0.21   0.18   0.15   0.21   0.19   0.14   0.28   0.40
## 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   0.17   0.20   0.19   0.16   0.19   0.21   0.23   0.28   0.26   0.38
## 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: 0.15
## lambda: 0.32
## Accept Rate: 0.23
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

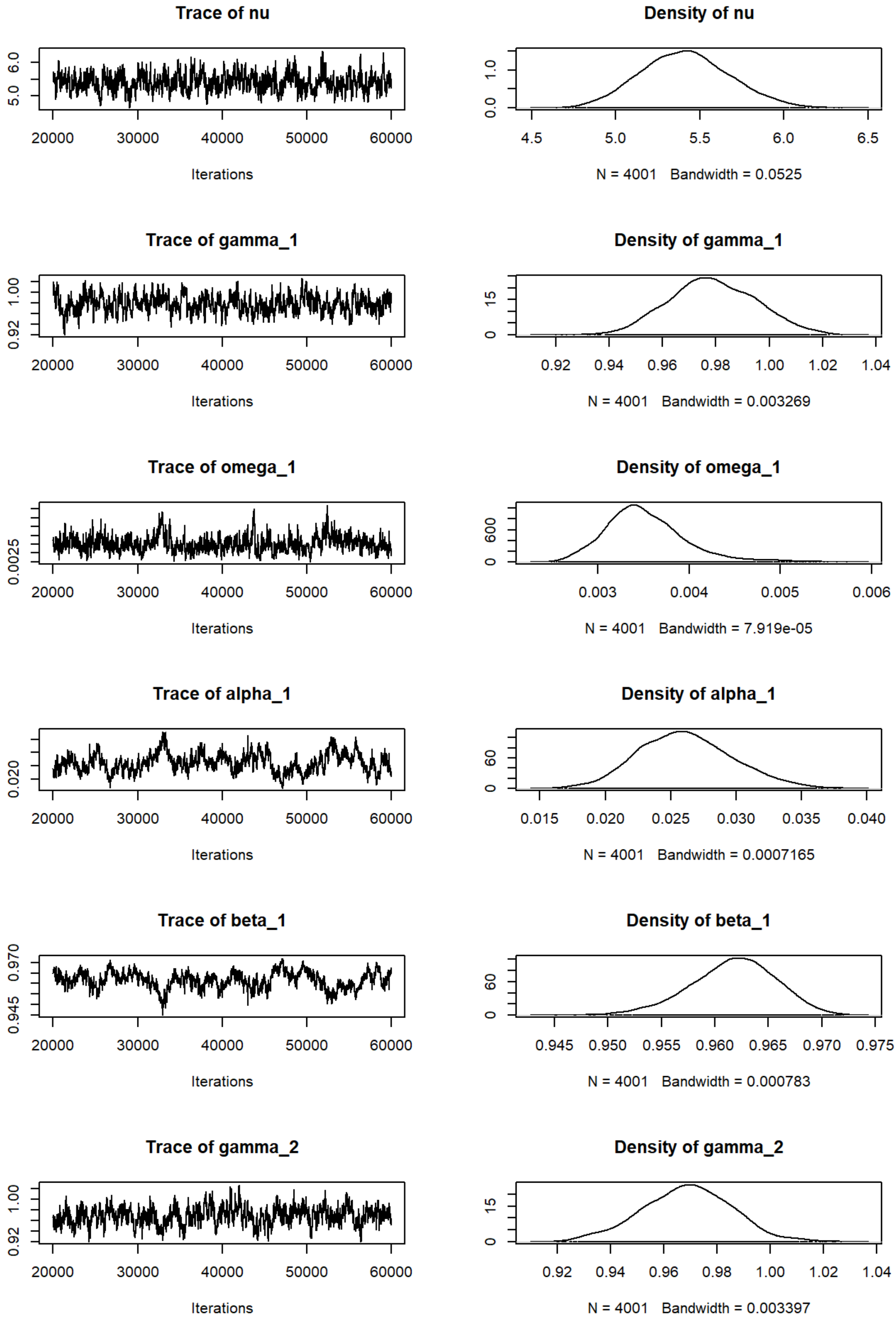
```
out2 <- increaseSim(out, nSim=50000)
```

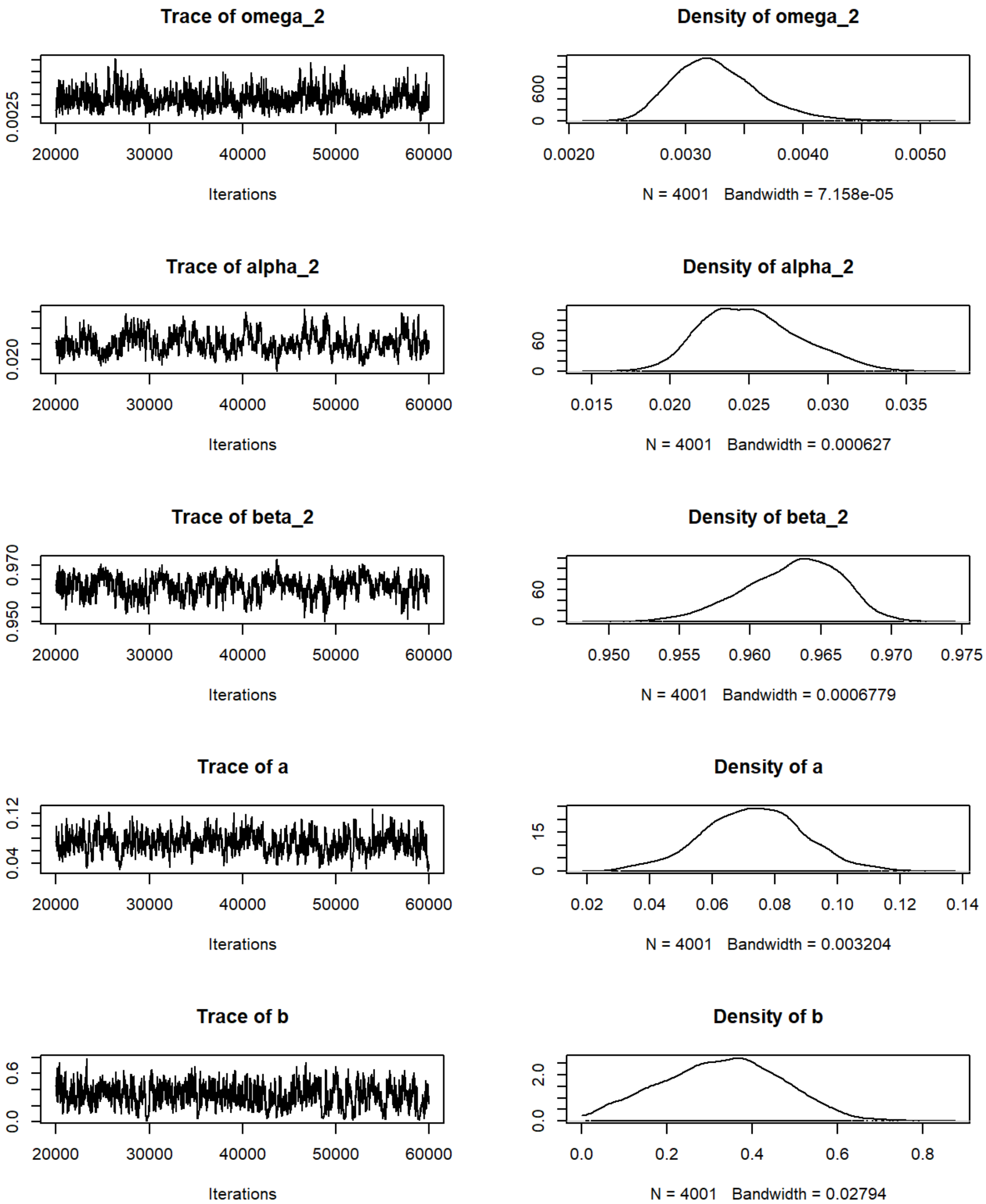
```
## Calibrating the Lambda coefficient:
## lambda: 0.32
## Accept Rate: 0.19
## lambda: 0.256
## Accept Rate: 0.25
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

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

end <- Sys.time()

# elapsed time
end-start
```

```
## Time difference of 4.404351 mins
```

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






```
## 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      5.412773 0.2601698 4.113e-03      1.757e-02
## gamma_1 0.978768 0.0162001 2.561e-04      1.249e-03
## omega_1 0.003521 0.0004440 7.019e-06      3.481e-05
## alpha_1 0.026142 0.0035511 5.614e-05      5.196e-04
## beta_1  0.961705 0.0039553 6.253e-05      6.108e-04
## gamma_2 0.968648 0.0168364 2.662e-04      1.323e-03
## omega_2 0.003260 0.0003674 5.808e-06      1.997e-05
## alpha_2 0.025180 0.0031074 4.913e-05      2.697e-04
## beta_2  0.962987 0.0033597 5.311e-05      2.795e-04
## a       0.073105 0.0158781 2.510e-04      9.943e-04
## b       0.328361 0.1384858 2.189e-03      8.678e-03
##
## 2. Quantiles for each variable:
##
##             2.5%      25%      50%      75%    97.5%
## nu      4.920120 5.231977 5.409097 5.584851 5.938925
## gamma_1 0.948338 0.967616 0.978299 0.990394 1.010386
## omega_1 0.002792 0.003225 0.003460 0.003751 0.004617
## alpha_1 0.019722 0.023583 0.025967 0.028423 0.033595
## beta_1  0.953229 0.959219 0.961888 0.964419 0.968818
## gamma_2 0.934407 0.957154 0.968922 0.979887 1.001104
## omega_2 0.002665 0.002997 0.003222 0.003472 0.004094
## alpha_2 0.019939 0.022904 0.024913 0.027179 0.031758
## beta_2  0.955796 0.960782 0.963339 0.965448 0.968660
## a       0.040734 0.062370 0.073216 0.083784 0.104769
## b       0.062019 0.229617 0.332876 0.426934 0.587236
```

```
# 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  1.971
## High_UB_S1    0.578
## Low_LB_B1    -0.567
## Low_LB_LBOP  -2.031
## High_omega    0.004
## High_alpha    0.026
## High_beta     0.962
## Low_omega     0.003
## Low_alpha     0.025
## Low_beta      0.963
```