GARCH parameters and quantiles estimation

Jose Augusto Fiorucci

20/11/2020

Input

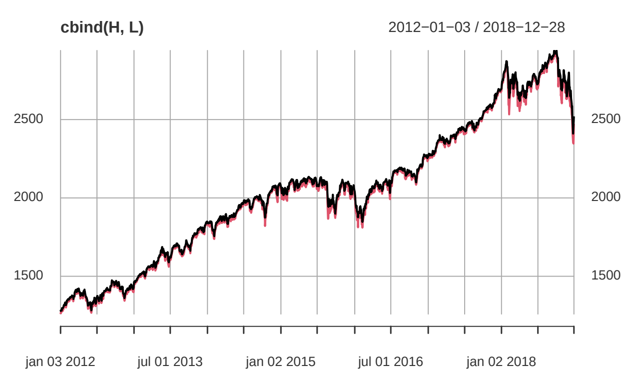
```
symbol = "^GSPC"
from=as.Date('2012-01-01')
to=as.Date('2018-12-31')
C_Trend = 0.95
C_Reaction = 0.50
```

Data download

```
x <- getSymbols.yahoo(symbol,auto.assign = FALSE, from=from, to=to)
```

High and Low

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



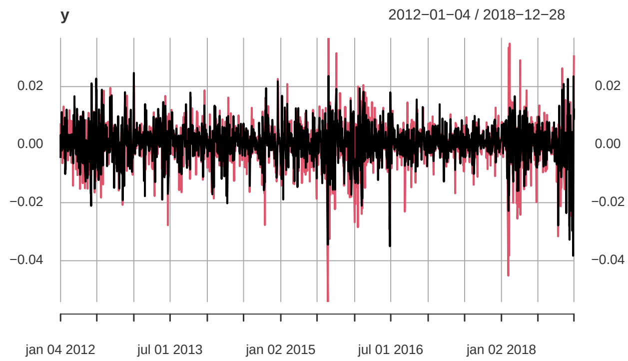
Returns

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

```
##           GSPC.High  GSPC.Low
## GSPC.High 1.0000000 0.7065524
```

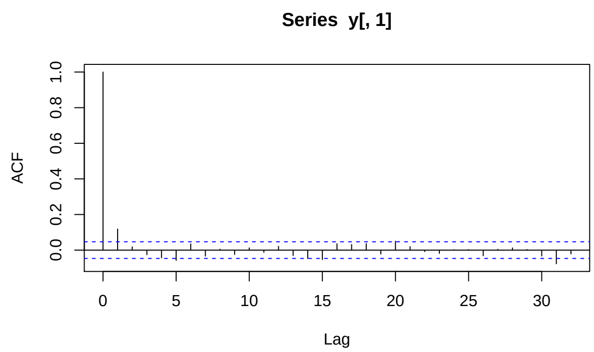
```
## GSPC.Low   0.7065524 1.0000000
```

```
plot(y)
```

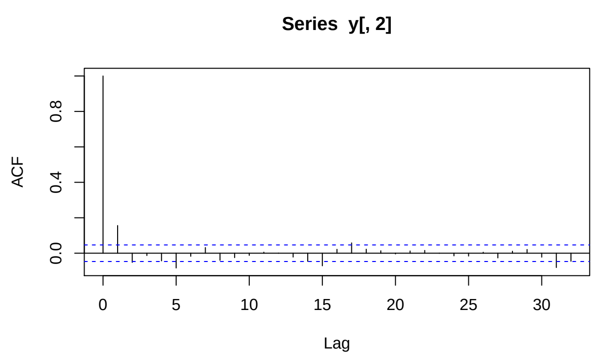


Autocorrelation

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

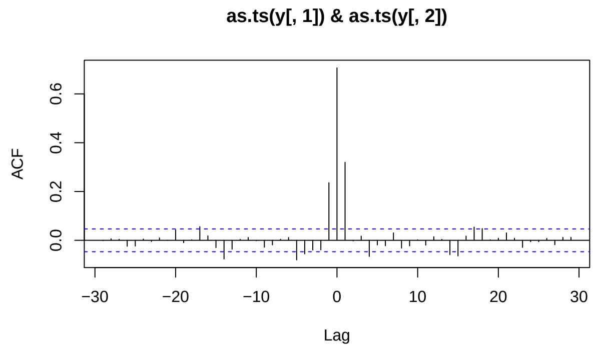



```
acf(y[,2])
```



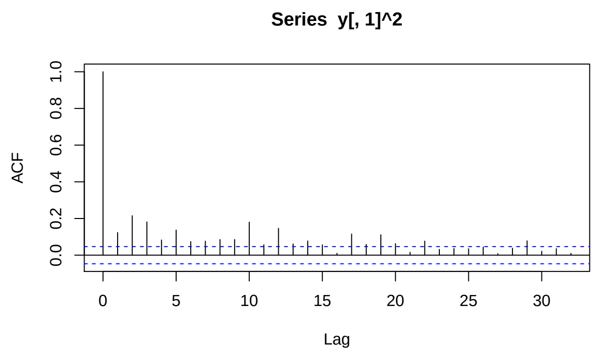
Cross correlation

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

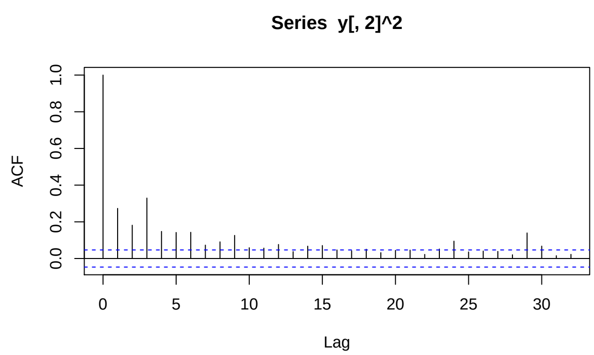


Volatility verification

```
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
```

```r
start <- Sys.time()

out <- bayesDccGarch(mY, control=list(print=FALSE, nPilotSim=3000))
```

```
## 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: 0.44
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

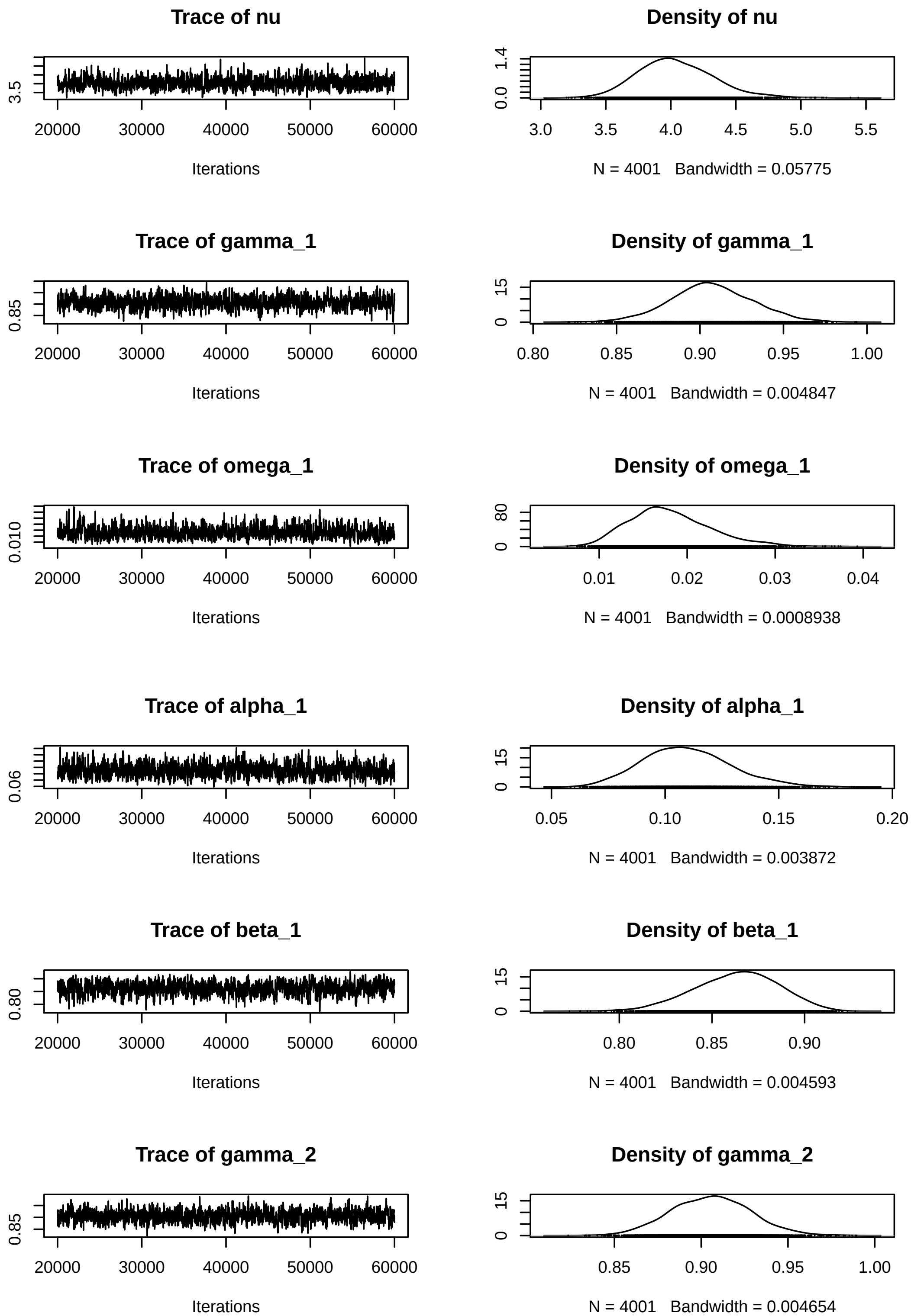
```r
out2 <- increaseSim(out, nSim=50000)
```

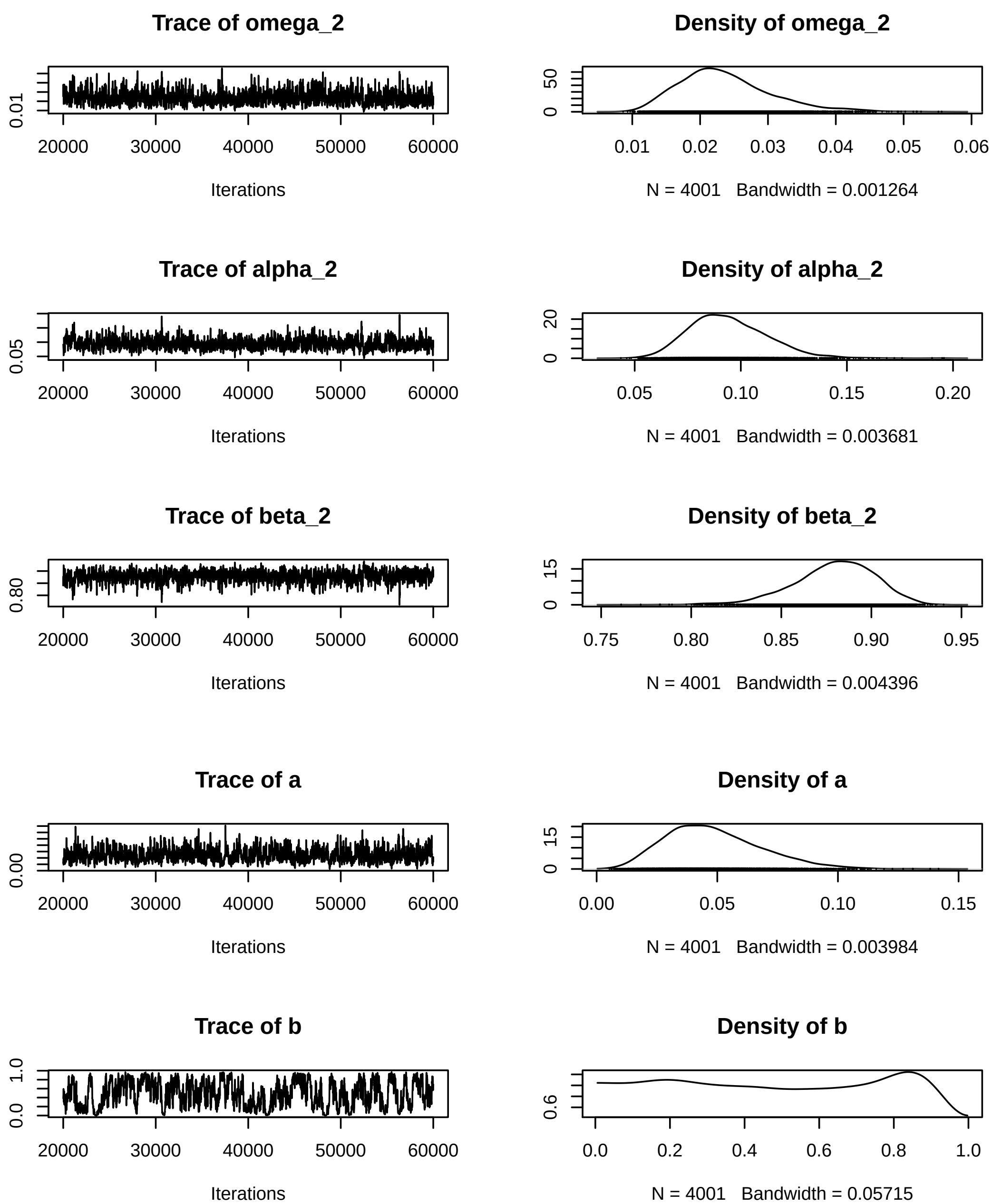
```
## Calibrating the Lambda coefficient:
## lambda: 0.4
## Accept Rate: 0.49
## Done.
## Starting the simulation by one-block random walk Metropolis-Hasting algorithm.
## Done.
```

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

end <- Sys.time()

# elapsed time
end-start
```

```
## Time difference of 1.064257 mins
```

```r
## Estimative of parameters
parEst <- summary(out)$statistics[,'Mean']

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


Trace of nu
3.5
20000 30000 40000 50000 60000
Iterations
Density of nu
0.0 1.4
3.0 3.5 4.0 4.5 5.0 5.5
N = 4001 Bandwidth = 0.05775
Trace of gamma_1
0.85
20000 30000 40000 50000 60000
Iterations
Density of gamma_1
0 15
0.80 0.85 0.90 0.95 1.00
N = 4001 Bandwidth = 0.004847
Trace of omega_1
0.010
20000 30000 40000 50000 60000
Iterations
Density of omega_1
0 80
0.01 0.02 0.03 0.04
N = 4001 Bandwidth = 0.0008938
Trace of alpha_1
0.06
20000 30000 40000 50000 60000
Iterations
Density of alpha_1
0 15
0.05 0.10 0.15 0.20
N = 4001 Bandwidth = 0.003872
Trace of beta_1
0.80
20000 30000 40000 50000 60000
Iterations
Density of beta_1
0 15
0.80 0.85 0.90
N = 4001 Bandwidth = 0.004593
Trace of gamma_2
0.85
20000 30000 40000 50000 60000
Iterations
Density of gamma_2
0 15
0.85 0.90 0.95 1.00
N = 4001 Bandwidth = 0.004654

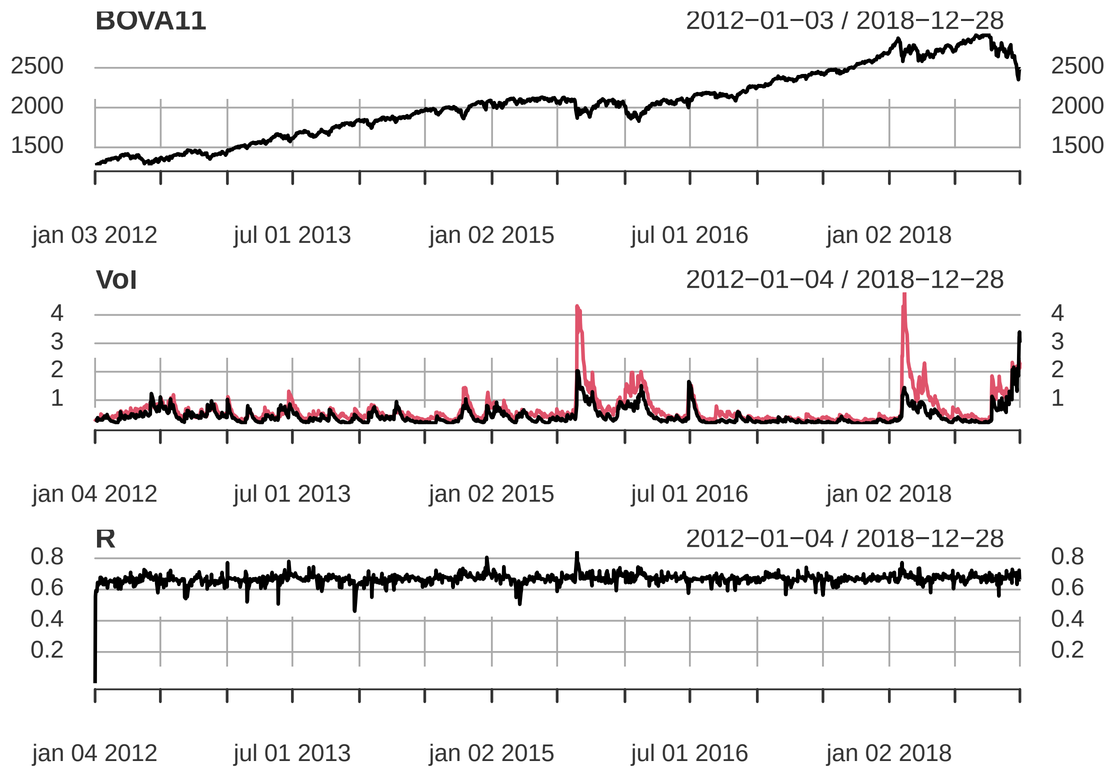



```r
## Estimative of parameters
out$MC %>% summary()
```

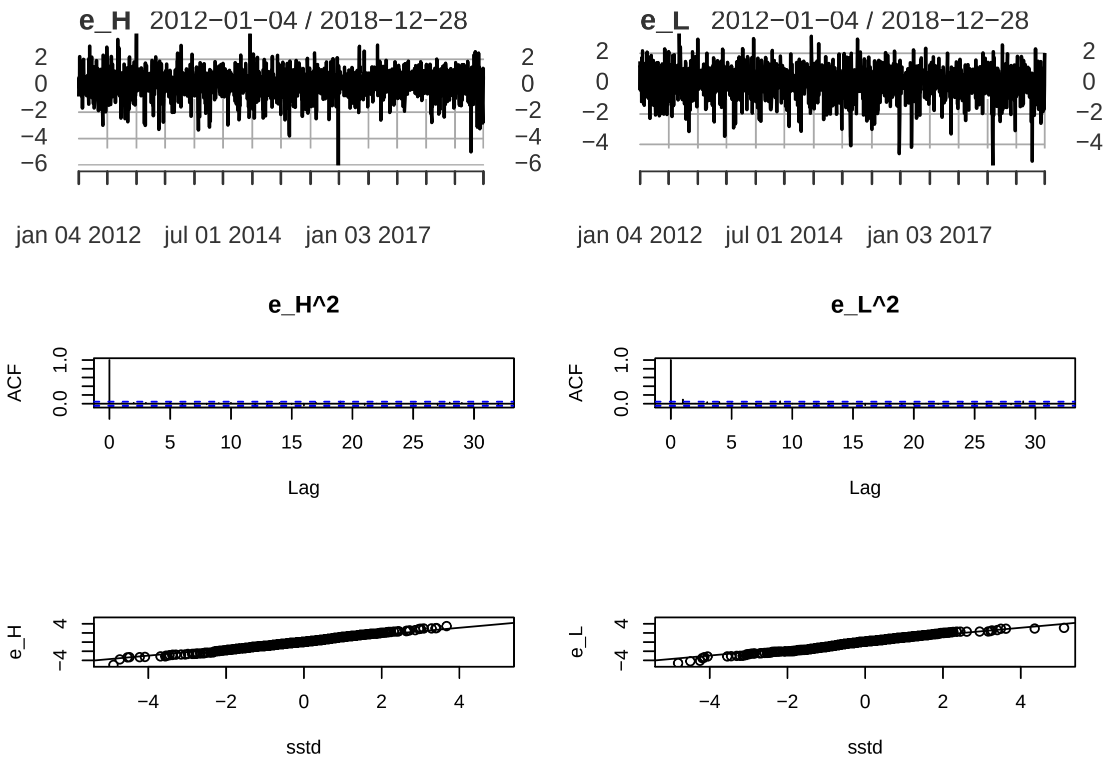
```
##
```

```
## 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      4.03845 0.290119 4.587e-03      0.0104835
## gamma_1 0.90698 0.024634 3.894e-04      0.0009179
## omega_1 0.01805 0.004620 7.304e-05      0.0001875
## alpha_1 0.10959 0.019301 3.051e-04      0.0007155
## beta_1  0.86317 0.022763 3.599e-04      0.0009104
## gamma_2 0.90569 0.023065 3.646e-04      0.0008471
## omega_2 0.02369 0.006782 1.072e-04      0.0002225
## alpha_2 0.09416 0.018244 2.884e-04      0.0006817
## beta_2  0.87956 0.022509 3.558e-04      0.0008072
## a       0.04849 0.019746 3.122e-04      0.0007619
## b       0.48573 0.283235 4.478e-03      0.0287163
##
## 2. Quantiles for each variable:
##
##            2.5%     25%     50%     75%   97.5%
## nu      3.52846 3.83732 4.01748 4.22081 4.67570
## gamma_1 0.85820 0.89106 0.90615 0.92325 0.95541
## omega_1 0.01055 0.01488 0.01751 0.02081 0.02866
## alpha_1 0.07493 0.09606 0.10852 0.12178 0.15095
## beta_1  0.81684 0.84805 0.86443 0.87931 0.90360
## gamma_2 0.86175 0.88975 0.90577 0.92124 0.95183
## omega_2 0.01303 0.01899 0.02274 0.02739 0.04033
## alpha_2 0.06376 0.08121 0.09250 0.10569 0.13295
## beta_2  0.83044 0.86620 0.88131 0.89540 0.91819
## a       0.01713 0.03410 0.04604 0.06056 0.09309
## b       0.02425 0.23159 0.48266 0.74852 0.92473
```

```r
## Conditional Correlation
R <-  xts(out$R[,2], order.by=index(y))



## Volatility
Vol <- xts(out$H[,c("H_1,1","H_2,2")], order.by=index(y))

par(mfrow=c(3,1))
plot(C, main="BOVA11")
plot(Vol)
plot(R, main="R")
```




```r
## Standard Residuals
r <- mY / sqrt(Vol)

par(mfrow=c(3,2))

plot(r[,1], main="e_H")
plot(r[,2], main="e_L")
acf(r[,1]^2, main="e_H^2")
acf(r[,2]^2, main="e_L^2")
r1 <- as.numeric(r[,1])
x <- rsstd(2000, mean = 0, sd = 1, nu = parEst['nu'], xi =parEst['gamma_1'])
qqplot(x=x, y=r1, xlim=c(-5, 5), ylim=c(-5, 5), ylab="e_H",xlab="sstd")
qqline(r1)
r2 <- as.numeric(r[,2])
x <- rsstd(2000, mean = 0, sd = 1, nu = parEst['nu'], xi =parEst['gamma_2'])
qqplot(x=x, y=r2 , xlim=c(-5, 5), ylim=c(-5, 5), ylab="e_L",xlab="sstd" )
qqline(r2)
```




```
# Prepare input for the expert advisor

## High
#HBOP
High_UB_HBOP = qsstd(p=1-(1-C_Trend)/2, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['gamma_1'])
#S1
High_UB_S1 = qsstd(p=1-(1-C_Reaction)/2, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['gamma_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'])

pH <- c(0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.975, 0.99, 0.995)
qH <- round(qsstd(p=pH, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['gamma_1']),3)
names(qH) <- paste0(100*pH,"%")
pL <- 1 - pH
qL <- round(qsstd(p=pL, mean = 0, sd = 1, nu = parEst['nu'], xi = parEst['gamma_2']),3)
names(qL) <- paste0(100*pL,"%")

qC <- rbind(qH, qL)
rownames(qC) <- c("High_UB", "Low_LB")
colnames(qC) <- paste0(100*pL,"%")

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",
```

```r
                     "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(qC)
```

```
           40%    35%    30%    25%    20%    15%    10%     5%   2.5%     1%
High_UB  0.235  0.332  0.435  0.549  0.679  0.840  1.059  1.437  1.840  2.443
Low_LB  -0.150 -0.256 -0.372 -0.502 -0.655 -0.846 -1.111 -1.577 -2.081 -2.840
          0.5%
High_UB  2.972
Low_LB  -3.511
```

```r
print(round(m,3))
```

```
              Value
High_UB_HBOP  1.840
High_UB_S1    0.549
Low_LB_B1    -0.502
Low_LB_LBOP  -2.081
High_omega    0.018
High_alpha    0.110
High_beta     0.863
Low_omega     0.024
Low_alpha     0.094
Low_beta      0.880
```