Time series and volatility analysis on Nasdaq100 with ARIMA-GARCH approach and Markov-switching approach

Introduction

Volatility plays an important role both for those who operate in financial markets and for those who study their characteristics and trends. The GARCH model describing the conditional variance of a financial process has been particularly popular among those who operate in the markets and need to attribute a value to volatility. Its theoretical approach, combined with an easy implementation, is at the basis of its frequent use in operational realities. Nevertheless, research activity has highlighted the limits of GARCH in modeling structural breaks often present in financial time series. The literature has ended up suggesting several solutions to the problem of excessive persistence of GARCH, both proposing variations and specifications, and new more complex solutions. Markov Switching GARCHs, with the introduction of a latent variable for the process describing volatility but preserving the classical structure of GARCHs within states, in this sense are models on the border between these two classes. The theoretical development and applications proposed in the literature have confirmed the better performance of MS-GARCH over GARCH in terms of volatility forecasting capability and persistence in parametric terms (source). The empirical analysis proposed in the second part of the thesis proposes, instead, a verification of the ability to specify shocks present in the time series of MS-GARCH models. The hypothesis is that structural breaks occur in the phases in which the model jumps from a regime of standard volatility to one with higher volatility. Further generalization of these models is also proposed by considering the assumption of specifying different density distributions for each volatility regime.

(Source for introduction) Modelli MS-GARCH e Break strutturali: sviluppi teorici ed evidenze empiriche Dott. Guido Caporilli Razza

Comparisons will be made on the various classical methods of volatility analysis, and we will try to specify the best model for both the mean and the variance conditional.

The main goal of this paper is to explain the behaviour of financial time series and the relationship between returns and conditional volatility.

Do ARIMA and GARCH processes provide parsimonious approximations to mean and volatility dynamics?

Index

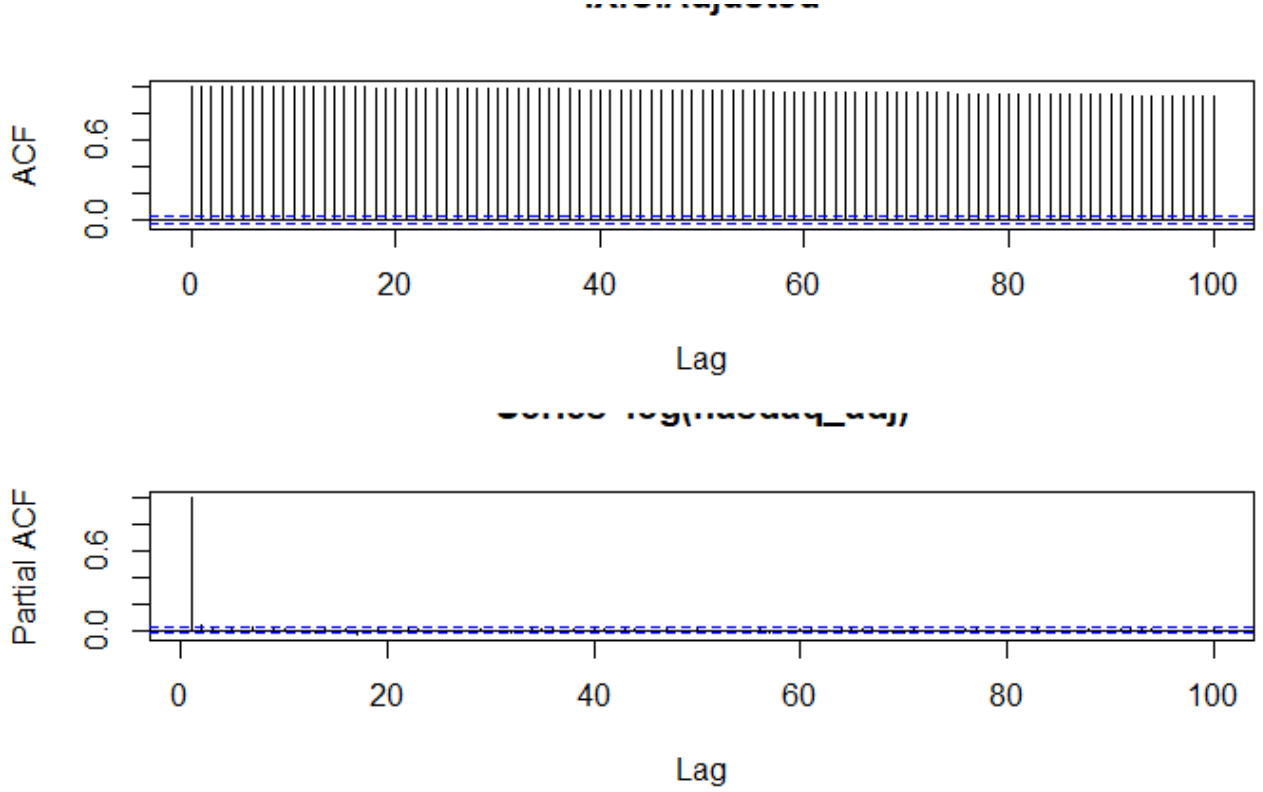

```
suppressMessages(library(quantmod))
suppressMessages(library(PerformanceAnalytics))
suppressMessages(library(xts))
suppressMessages(library(zoo))
suppressMessages(library(fBasics))
suppressMessages(library(mvtnorm))
suppressMessages(library(rugarch))
suppressMessages(library(SBAGM))
suppressMessages(library(forecast))
suppressMessages(library(RcppRoll))
suppressMessages(library(fGarch))
suppressMessages(library(stochvol))
suppressMessages(library(rmgarch))
suppressMessages(library(MSGARCH))
suppressMessages(library(GAS))
#
```

exploratory analysis

```
nasdaq <- suppressMessages(Ad(getSymbols("^IXIC",from= "2002-01-01",to= "2022-03-2
7", auto.assign = FALSE)))
chartSeries(nasdaq)
```

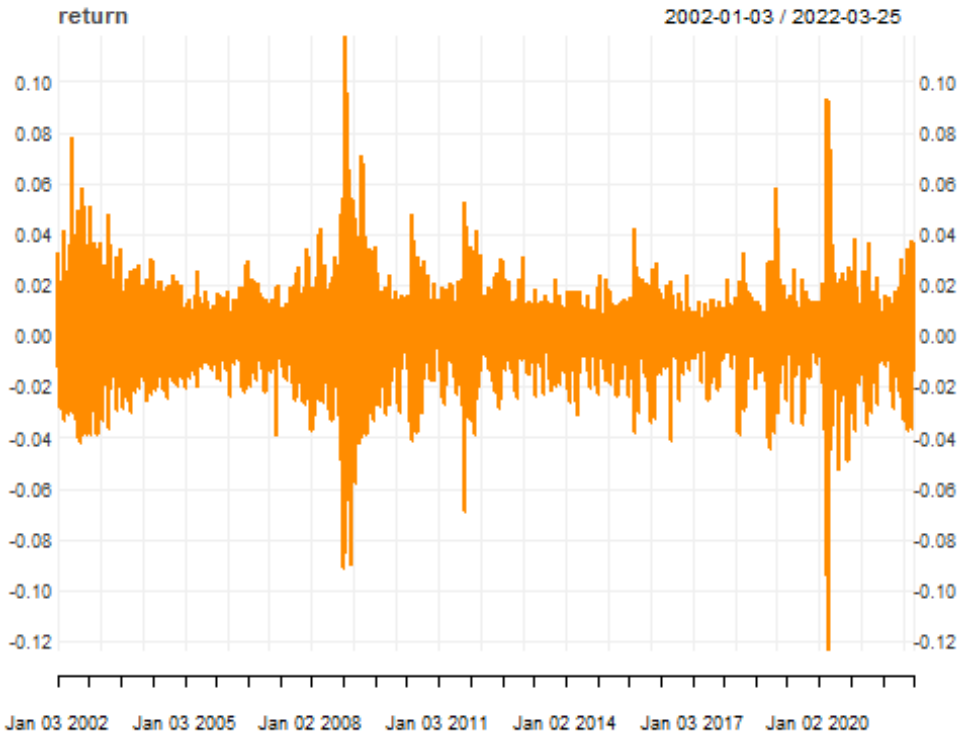
source: Yahoo Finance from 2002-01-01 to 2022-03-27

```
nasdaq_adj <- ts(nasdaq$IXIC.Adjusted)

par(mfrow=c(2,1))
acf(log(nasdaq_adj),lag.max = 100)
pacf(log(nasdaq_adj),lag.max = 100)
```



From the ACF plot, we observe that the plot decays to zero slowly, meaning the shock affects the process permanently. We can conclude that we need to perform time series analysis on the daily return (log return) of the stock prices.

```
return <- CalculateReturns(nasdaq)
return <- return[-c(1),]
chart_Series(return)
```

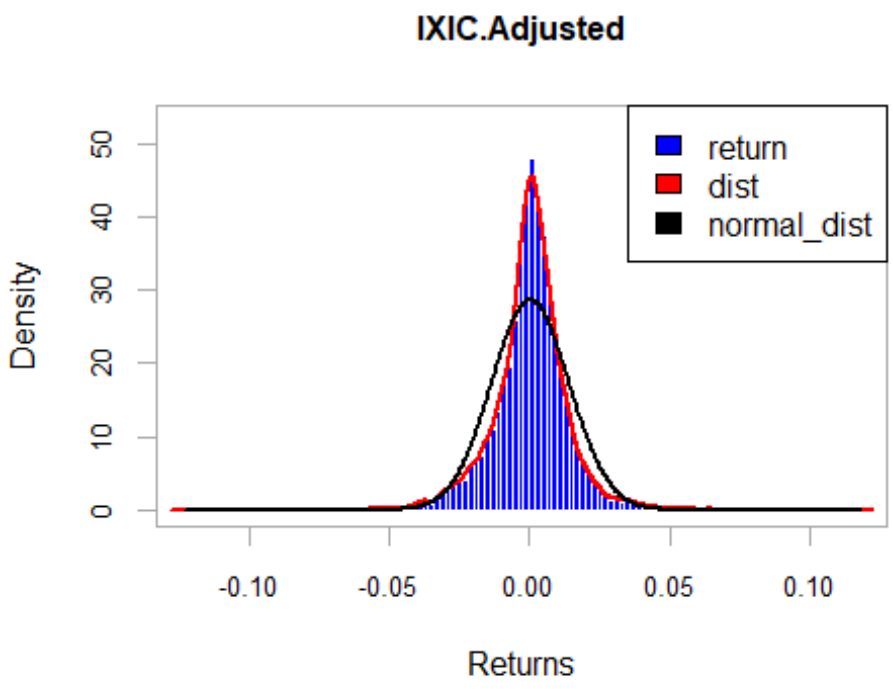


The log returns of the stock prices have zero mean but high volatility, standard stationarity won't work.

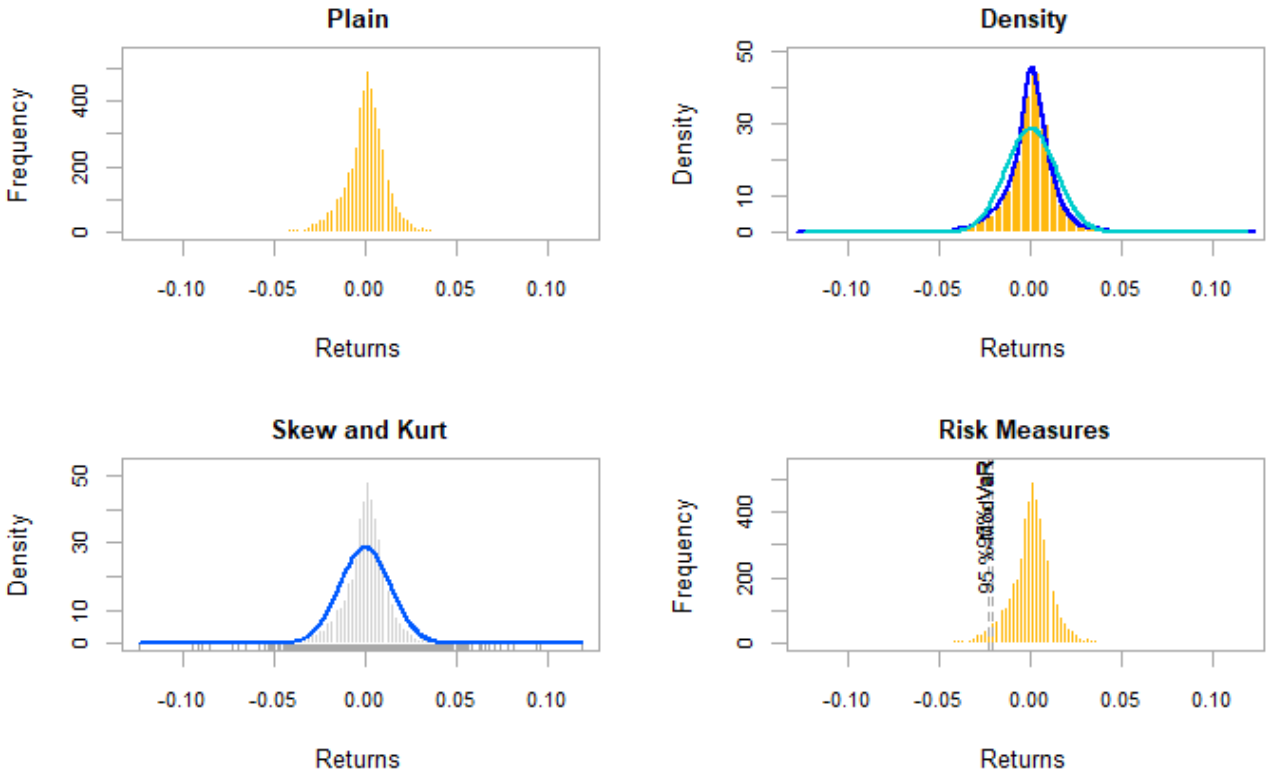
```
print(basicStats(return))

##             IXIC.Adjusted
## nobs          5093.000000
## NAs              0.000000
## Minimum         -0.123213
## Maximum          0.118059
## 1. Quartile     -0.005530
## 3. Quartile      0.007297
## Mean             0.000483
## Median           0.000987
## Sum              2.461913
## SE Mean          0.000195
## LCL Mean         0.000101
## UCL Mean         0.000865
## Variance         0.000193
## Stdev            0.013906
## Skewness        -0.133247
## Kurtosis         6.808118
```

From the basic statistics of the log return of the stock prices we observe that the mean is 0 and the distribution of log returns has large kurtosis (fat tails).

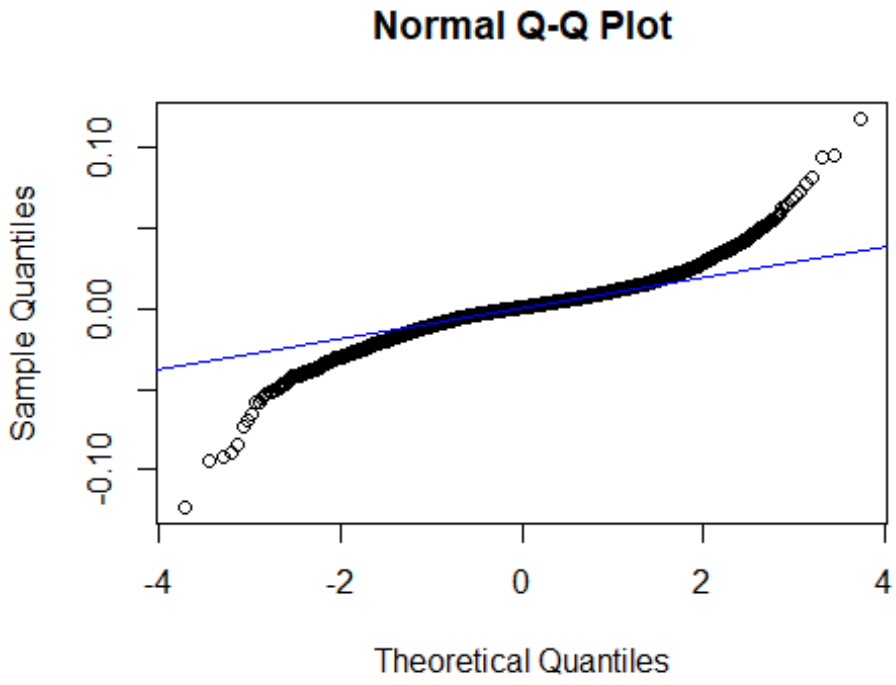
```
chart.Histogram(return, method= c('add.density',"add.normal"),
                colorset = c("blue","red","black"))
legend("topright",legend = c("return","dist","normal_dist")
       ,fill = c("blue","red","black"))
```




```
layout(rbind(c(1,2),c(3,4)))
chart.Histogram(return, main = "Plain", methods = NULL, colorset = c("darkgoldenro
d1"))
chart.Histogram(return, main = "Density", breaks=40,
                methods = c("add.density", "add.normal"), colorset = c("darkgolden
rod1","blue","cyan3"))
chart.Histogram(return, main = "Skew and Kurt", methods = c("add.centered", "add.r
ug"))
chart.Histogram(return, main = "Risk Measures", methods = c("add.risk"),colorset =
"darkgoldenrod1")
```



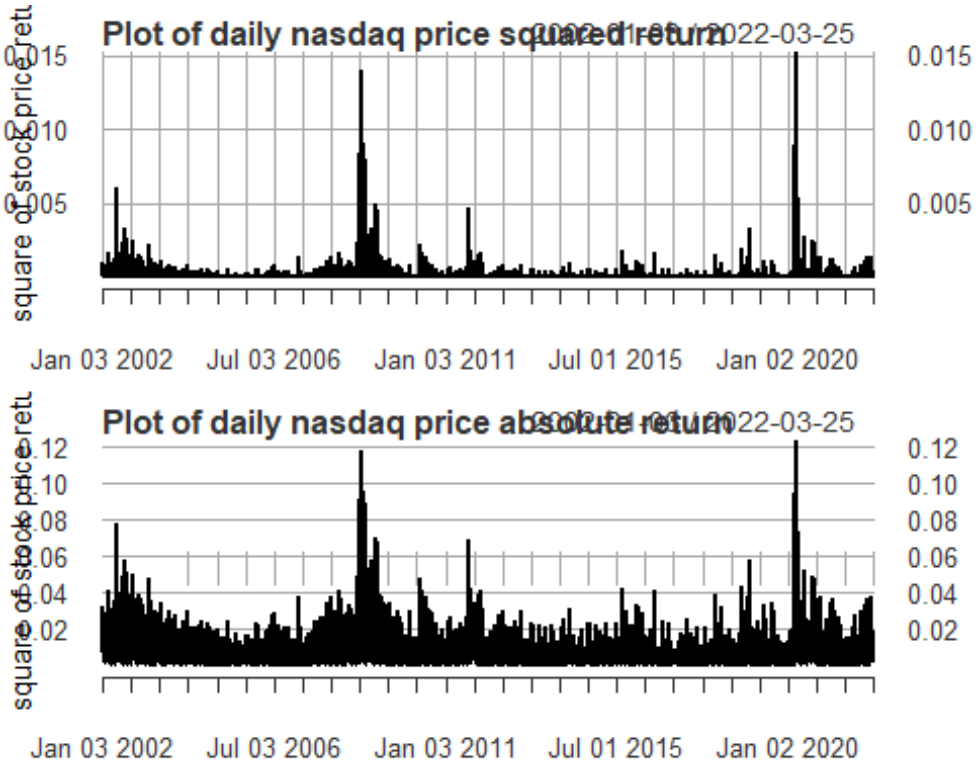

```
qqnorm(return)
qqline(return, col = "blue")
```



The graphs confirm that seems to be more skewed than the normal distribution

the series has a somewhat normal distribution with fat tails at both ends

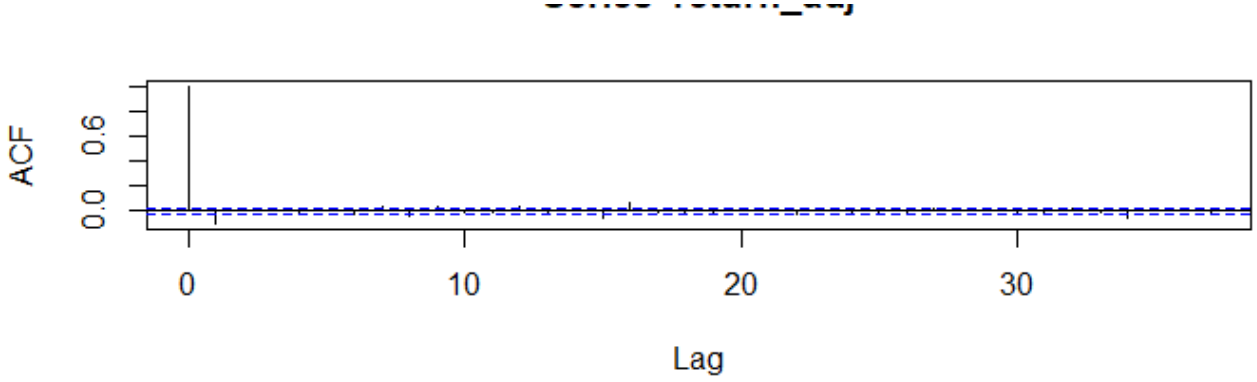
```
par(mfrow=c(2,1))
plot(return^2,type='l', ylab = "square of stock price return", main="Plot of daily nasdaq price squared return")
plot(abs(return),type='l', ylab = "square of stock price return", main="Plot of daily nasdaq price absolute return")
```

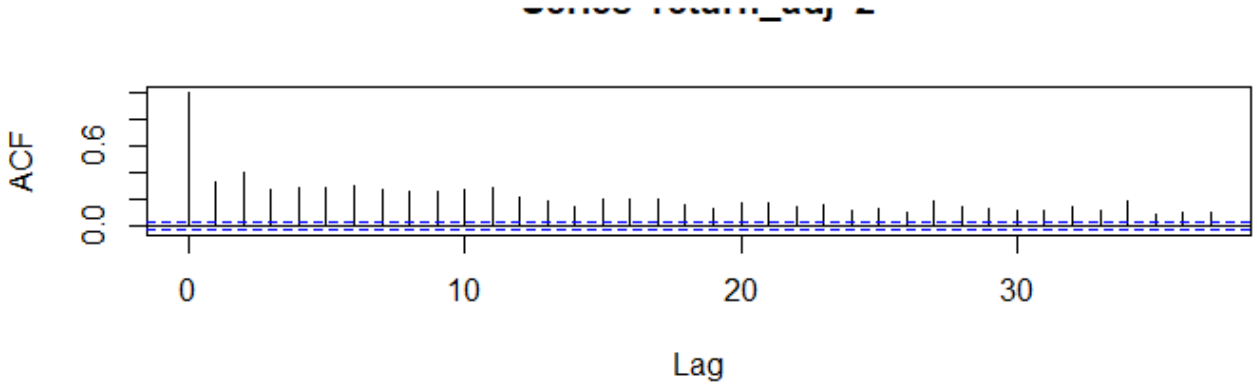


During the years 2002, 2008-2009, 2013 and 2020, there is spike in volatility indicating non-constant conditional volatility. The high volatility doesn't decrease fast because of the negative shocks have an effect on the process.

ACF plot of log return of prices

```
return_adj <- CalculateReturns(nasdaq_adj)
return_adj <-return_adj[-1]
par(mfrow=c(2,1))
acf(return_adj)
acf(return_adj^2)
```





The statistics showed that the mean was constant and nearly 0. This is further confirmed by the time series plot. The ACF plot further shows that since, the log stock price returns are not correlated, the mean is constant for the time series. However, the squared stock price return values have high correlation. Thus, we may conclude that the log returns process has a strong non-linear dependence.

theoretical background means models

synthetic i.i.d. data

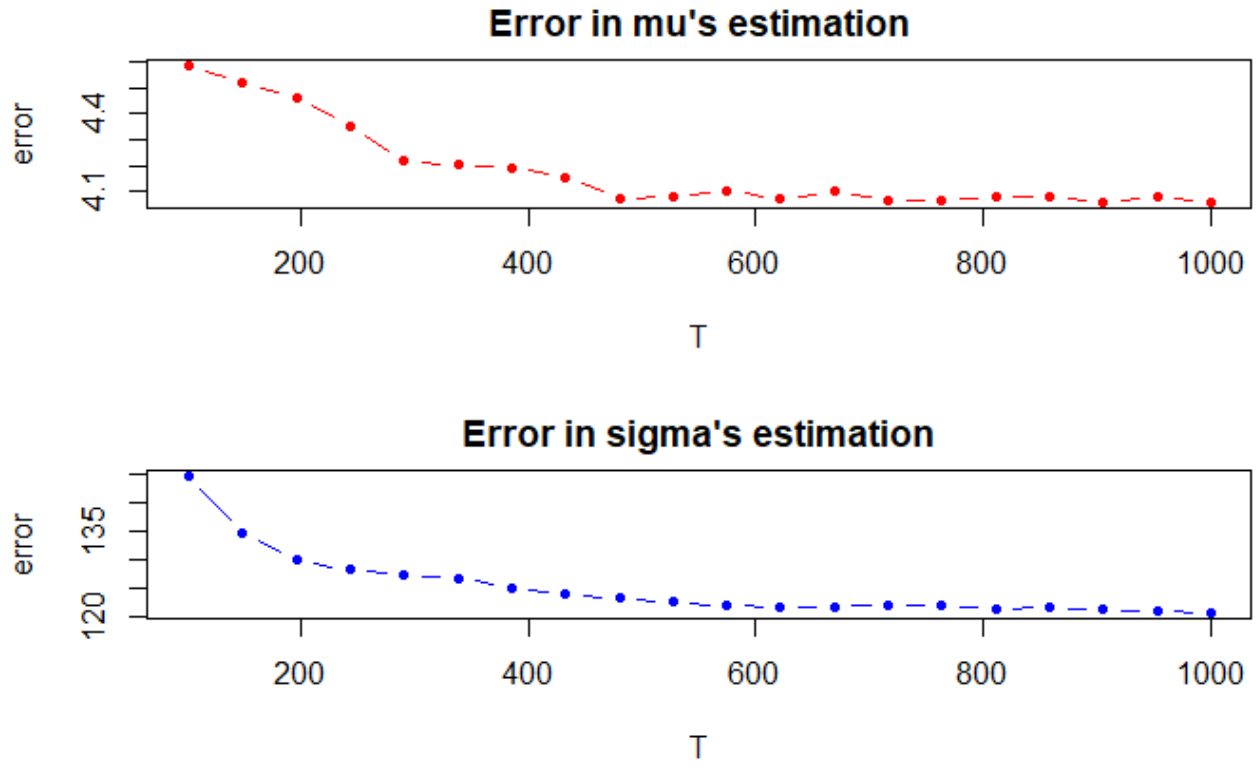
generate Gaussian synthetic return data

```
N <- 100
T_max <- 1000
mu <- runif(N)
U <- t(rmvnorm(n = round(0.7*N), sigma = 0.1*diag(N)))
Sigma <- U %*% t(U) + diag(N)
X <- rmvnorm(n = T_max, mean = mu, sigma = Sigma)

# now loop over subsets of the samples
error_mu_vs_T <- error_Sigma_vs_T <- NULL
T_sweep <- ceiling(seq(1.01*N, T_max, length.out = 20))
for (T_ in T_sweep) {
  X_ <- X[1:T_, ]
  # sample estimates
  mu_sm <- colMeans(X_)
```

```
  Sigma_scm <- cov(X_)
  # compute errors
  error_mu_vs_T    <- c(error_mu_vs_T,    norm(mu_sm     - mu, "2"))
  error_Sigma_vs_T <- c(error_Sigma_vs_T, norm(Sigma_scm - Sigma, "F"))
}
names(error_mu_vs_T) <- names(error_Sigma_vs_T) <- paste("T =", T_sweep)
```

I started by generating synthetic data, looped over subsets of the samples, estimate them, and compute the errors, then make sure the estimation process gives the correct results (sanity check)

```
par(mfrow=c(2,1))
plot(T_sweep, error_mu_vs_T, type = "b", pch = 20, col = "red",
     main = "Error in mu's estimation", xlab = "T", ylab = "error")

plot(T_sweep, error_Sigma_vs_T, type = "b", pch = 20, col = "blue",
     main = "Error in sigma's estimation", xlab = "T", ylab = "error")
```

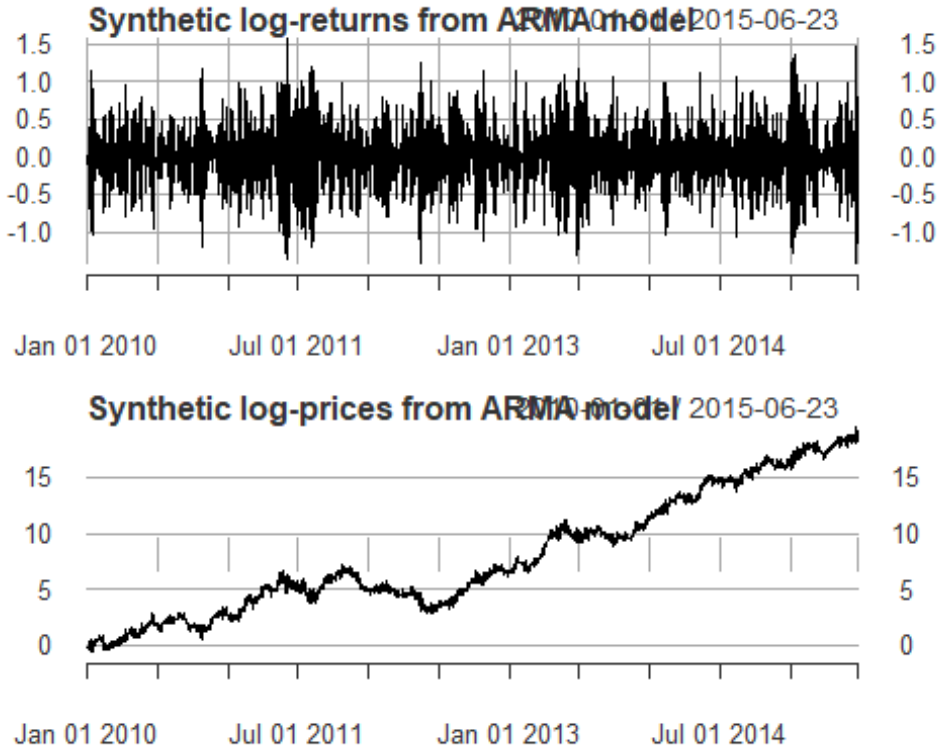


The sanity check passed; the plot shows a decreasing error for increasing T

Univariate ARMA model

```
arma_fixed_spec <- arfimaspec(mean.model = list(armaOrder = c(1,0), include.mean =
TRUE),
                              fixed.pars = list(mu = 0.01, ar1 = -0.9, sigma = 0.2
))
true_params <- unlist(arma_fixed_spec@model$fixed.pars)
# simulate one path
```

```
T <- 2000
path_arma <- arfimapath(arma_fixed_spec, n.sim = T)
# convert to xts and plot
synth_log_returns <- xts(path_arma@path$seriesSim, order.by = as.Date("2010-01-01"
) + 0:(T-1))
synth_log_prices <- xts(diffinv(synth_log_returns)[-1], order.by = index(synth_log
_returns))
par(mfrow=c(2,1))
plot(synth_log_returns, main = "Synthetic log-returns from ARMA model", lwd = 1.5)
plot(synth_log_prices, main = "Synthetic log-prices from ARMA model", lwd = 1.5)
```

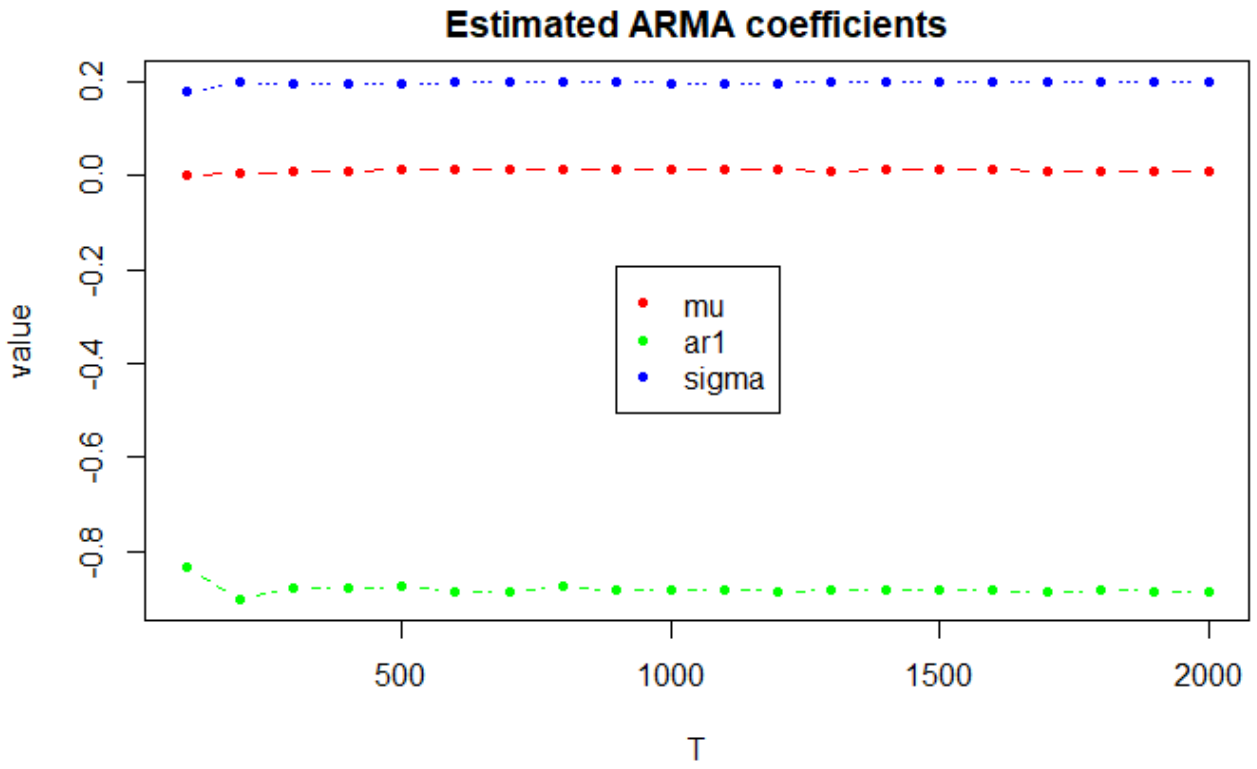


specify an AR(1) model with given coefficients and parameters (where mu, ar1, sigma = 0.01, -0.90, 0.20) are the true parameters and the plot above is the simulated path

After that I estimated a model

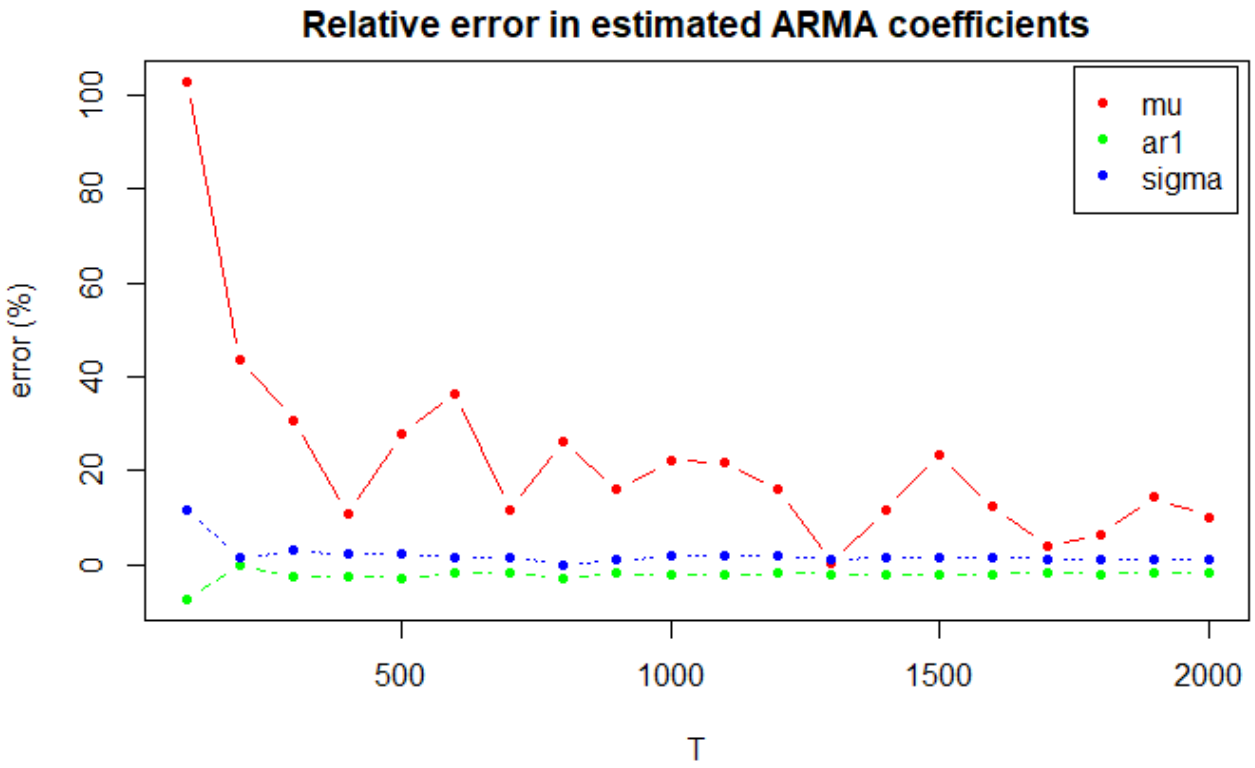
```
arma_spec = arfimaspec(mean.model = list(armaOrder = c(1,0), include.mean = TRUE))
arma_fit <- arfimafit(spec = arma_spec, data = synth_log_returns)
abs(coef(arma_fit) - true_params)

##           mu          ar1        sigma
## 0.0006186942 0.0059595185 0.0025511270

# loop
estim_coeffs_vs_T <- error_coeffs_vs_T <- NULL
T_sweep <- ceiling(seq(100, T, length.out = 20))
for (T_ in T_sweep) {
  arma_fit <- arfimafit(spec = arma_spec, data = synth_log_returns[1:T_])
  estim_coeffs_vs_T <- rbind(estim_coeffs_vs_T, coef(arma_fit))
```

```
  error_coeffs_vs_T <- rbind(error_coeffs_vs_T, abs(coef(arma_fit) - true_params)/
true_params)
}
rownames(error_coeffs_vs_T) <- rownames(estim_coeffs_vs_T) <- paste("T =", T_sweep
)

matplot(T_sweep, estim_coeffs_vs_T,
        main = "Estimated ARMA coefficients", xlab = "T", ylab = "value",
        type = "b", pch = 20, col = rainbow(3))
legend("center", inset = 0.01, legend = colnames(estim_coeffs_vs_T), pch = 20, col
= rainbow(3))
```



they're aligned with the true parameters

```
matplot(T_sweep, 100*error_coeffs_vs_T,
        main = "Relative error in estimated ARMA coefficients", xlab = "T", ylab =
"error (%)",
        type = "b", pch = 20, col = rainbow(3))
legend("topright", inset = 0.01, legend = colnames(error_coeffs_vs_T), pch = 20, c
ol = rainbow(3))
```



The estimation of mu is very noisy, however the coefficients seem to be well estimated after 800 samples.

In practice, the order is unknown, and one must try different combinations of orders. The higher the order, the better the fit, but this will inevitably produce overfitting. Many methods have been developed to penalize the increase of the order complexity to avoid overfitting (AIC, BIC).

```
arma_fit <- autoarfima(data = synth_log_returns,
                       ar.max = 3, ma.max = 3, include.mean = TRUE,
                       criterion = "BIC", method = "partial")
#see the ranking of the combinations
arma_fit$rank.matrix

##    AR MA Mean ARFIMA         BIC converged
## 1   1  0    1      0 -0.34424952         1
## 2   2  0    1      0 -0.34212182         1
## 3   1  1    1      0 -0.34193858         1
## 4   2  1    1      0 -0.33859351         1
## 5   3  0    1      0 -0.33842746         1
## 6   1  2    1      0 -0.33824313         1
## 7   3  1    1      0 -0.33594955         1
## 8   1  3    1      0 -0.33484541         1
## 9   2  2    1      0 -0.33466143         1
## 10  3  2    1      0 -0.33251617         1
## 11  2  3    1      0 -0.33132812         1
## 12  3  3    1      0 -0.32981539         1
## 13  0  3    1      0 -0.02446239         1
## 14  0  2    1      0  0.17446465         1
```

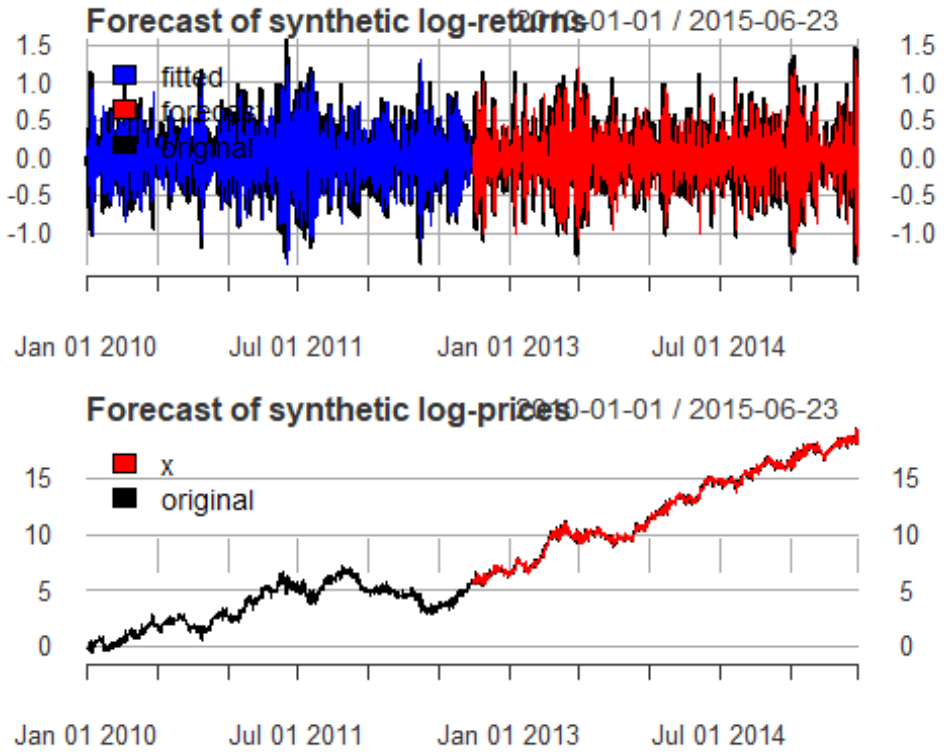
```
## 15  0  1    1      0  0.56412197         1
## 16  0  0    1      0  1.36496316         1
```

and choose the best

```
armaOrder <- arma_fit$rank.matrix[1, c("AR","MA")]
```

Then estimate the model excluding the out of sample, once the ARMA model parameters have been estimated ϕ^i and θ^j, can we use the model to forecast the values ahead.

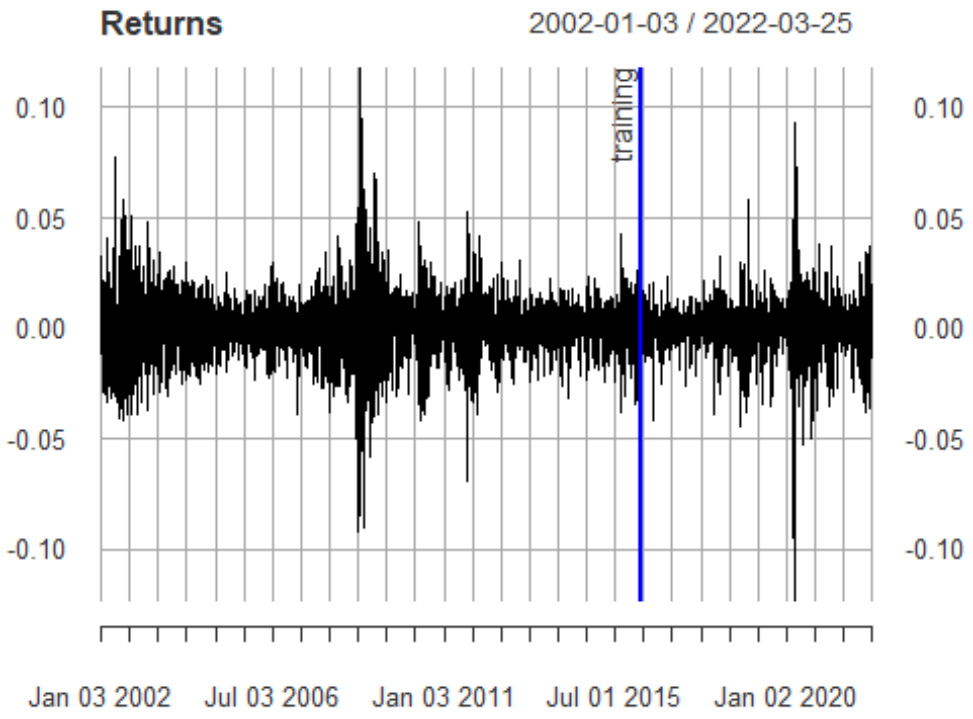
```
out_of_sample <- round(T/2)
dates_out_of_sample <- tail(index(synth_log_returns), out_of_sample)
arma_spec = arfimaspec(mean.model = list(armaOrder = c(1,0), include.mean = TRUE))
arma_fit <- arfimafit(spec = arma_spec, data = synth_log_returns, out.sample = out
_of_sample)

# forecast log-returns along the whole out-of-sample
arma_fore <- arfimaforecast(arma_fit, n.ahead = 1, n.roll = out_of_sample-1)
forecast_log_returns <- xts(arma_fore@forecast$seriesFor[1, ], dates_out_of_sample
)

# recover log-prices
prev_log_price <- head(tail(synth_log_prices, out_of_sample+1), out_of_sample)
forecast_log_prices <- xts(prev_log_price + arma_fore@forecast$seriesFor[1, ], dat
es_out_of_sample)

par(mfrow=c(2,1))
# plot of log-returns
plot(cbind("fitted"  = fitted(arma_fit),
           "forecast" = forecast_log_returns,
           "original" = synth_log_returns),
     col = c("blue", "red", "black"), lwd = c(0.5, 0.5, 2),
     main = "Forecast of synthetic log-returns", legend.loc = "topleft")
# plot of log-prices
plot(cbind("forecast" = forecast_log_prices,
           "original" = synth_log_prices),
     col = c("red", "black"), lwd = c(0.5, 0.5, 2),
     main = "Forecast of synthetic log-prices", legend.loc = "topleft")
```



prepare training and test data

I divided the data in training and test set, and used the training data (for t=1,...,Ttrn) to fit different models. Fit AR(1), ARMA(2,2), ARMA(2,1), MA(1) and use the different models to forecast the log-returns. Then compute the forecast errors

```
T <- nrow(return)
T_trn <- round(0.7*T)
T_tst <- T - T_trn
return_trn <- return[1:T_trn]
return_tst <- return[-c(1:T_trn)]
{ plot(return, main = "Returns", lwd = 1.5)
  addEventLines(xts("training", index(return[T_trn])), srt=90, pos=2, lwd = 2, col
= "blue") }
```



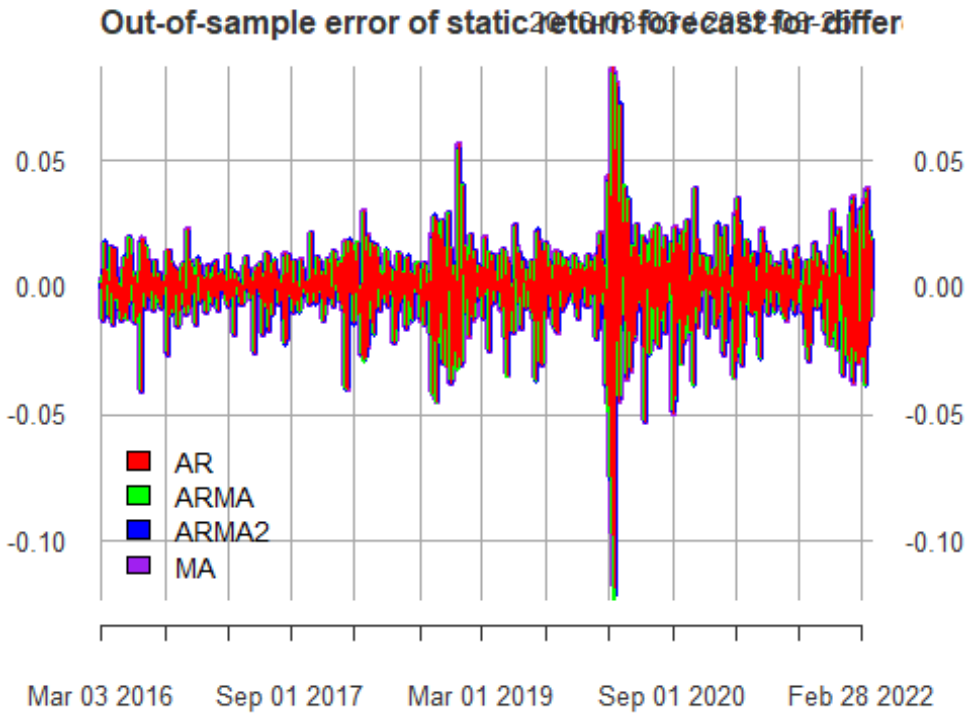
Static -vs- rolling window comparison

```
# fit AR(1) model
ar_spec <- arfimaspec(mean.model = list(armaOrder = c(1,0), include.mean = TRUE))
ar_fit <- arfimafit(spec = ar_spec, data = return, out.sample = T_tst)
coef(ar_fit)

##            mu           ar1         sigma
##  0.0003419991 -0.0613223846  0.0141055726

# fit ARMA(2,2) model
arma_spec <- arfimaspec(mean.model = list(armaOrder = c(2,2), include.mean = TRUE)
)
arma_fit <- arfimafit(spec = arma_spec, data = return, out.sample = T_tst)
coef(arma_fit)

##            mu           ar1           ar2           ma1           ma2
##  0.0003397845 -0.0850509253  0.3231308897  0.0243257672 -0.3718292844
##         sigma
##  0.0140858178

# fit ARMA(2,1) model
arma_spec_2 <- arfimaspec(mean.model = list(armaOrder = c(2,1), include.mean = TRU
E))
arma_fit_2 <- arfimafit(spec = arma_spec_2, data = return, out.sample = T_tst)
coef(arma_fit_2)
```

```
##            mu           ar1           ar2           ma1         sigma
##  0.0003401456 -0.0329897056 -0.0399882929 -0.0314474320  0.0140871516

# fit MA(1) model
ma_spec <- arfimaspec(mean.model = list(armaOrder = c(0,1), include.mean = TRUE))
ma_fit <- arfimafit(spec = ma_spec, data = return, out.sample = T_tst)
coef(ma_fit)

##            mu           ma1         sigma
##  0.0003422499 -0.0666717047  0.0141035541

#use the different models to forecast the log-returns:
dates_out_of_sample <- tail(index(return), T_tst)

# forecast with AR(1) model
ar_fore_return <- xts(arfimaforecast(ar_fit, n.ahead = 1, n.roll = T_tst - 1)@fore
cast$seriesFor[1, ],
                      dates_out_of_sample)

# forecast with ARMA(2,2) model
arma_fore_return <- xts(arfimaforecast(arma_fit, n.ahead = 1, n.roll = T_tst - 1)@
forecast$seriesFor[1, ],
                        dates_out_of_sample)

# forecast with ARMA(2,2) model
arma_2_fore_return <- xts(arfimaforecast(arma_fit_2, n.ahead = 1, n.roll = T_tst -
1)@forecast$seriesFor[1, ],
                      dates_out_of_sample)

# forecast with MA(1) model
ma_fore_return <- xts(arfimaforecast(ma_fit, n.ahead = 1, n.roll = T_tst - 1)@fore
cast$seriesFor[1, ],
                    dates_out_of_sample)

#compute the forecast errors
error_var <- rbind( "AR(1)"                 = c(var(return - fitted(ar_fit)),
                                                var(return - ar_fore_return)),
                    "ARMA(2,2)"             = c(var(return - fitted(arma_fit)),
                                                var(return - arma_fore_return)),
                    "ARMA(2,1)"             = c(var(return - fitted(arma_fit_2)),
                                                var(return - arma_2_fore_return)),
                    "MA(1,0)"             = c(var(return - fitted(ma_fit)),
                                                var(return - ma_fore_return)))
colnames(error_var) <- c("in-sample", "out-of-sample")
print(error_var)

##              in-sample out-of-sample
## AR(1)     0.0001989092  0.0001752712
## ARMA(2,2) 0.0001984664  0.0001767856
## ARMA(2,1) 0.0001985510  0.0001767177
## MA(1,0)   0.0001988444  0.0001751717
```

The important quantity is the out-of-sample error: i can see that increasing the model complexity may give few results. It seems that the simplest AR or MA model is good enough in terms of error of forecast returns.

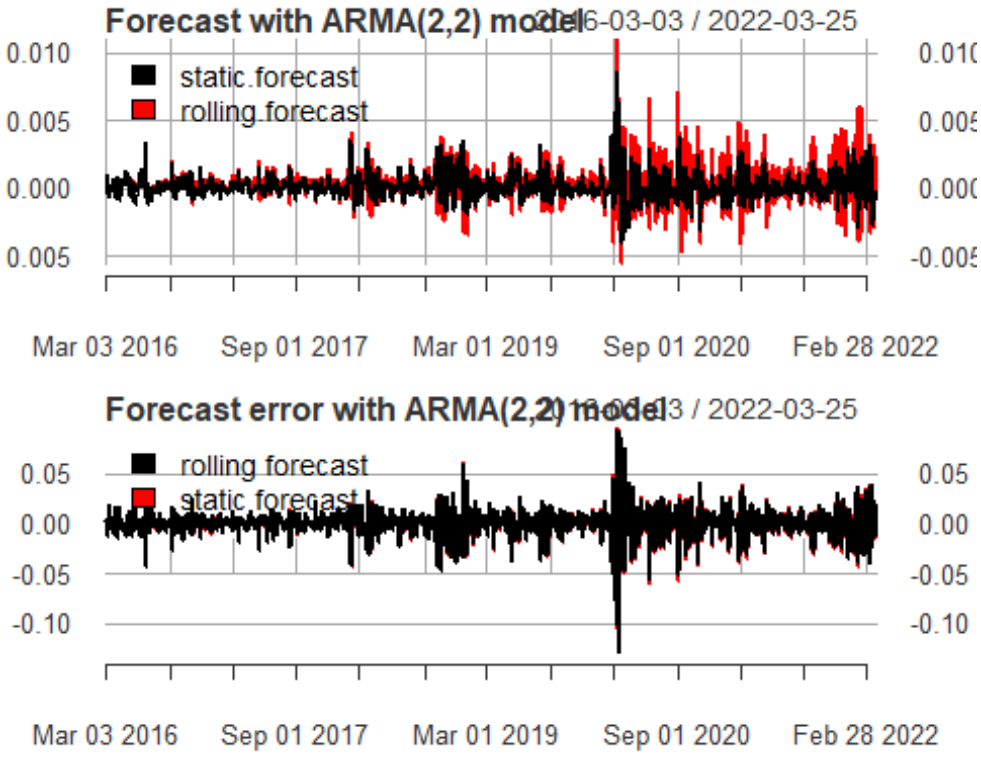
```
error_return <- cbind(return - ar_fore_return,
                          return - arma_fore_return,
                          return - arma_2_fore_return,
                          return - ma_fore_return)

names(error_return) <- c("AR", "ARMA", "ARMA2", 'MA')
plot(error_return, col = c("red", "green", "blue",'purple'), lwd = c(0.2, 2, 3, 4)
,
     main = "Out-of-sample error of static return forecast for different models",
     legend.loc = "bottomleft")
```



it's possible to obtain better results by refitting the models, doing that the error should tends to become better as the time advances.

```
nasdaq_for_roll <- read.csv("^IXIC(TRUE).csv",row.names = NULL)
specx <- arfimaspec(mean.model = list(armaOrder = c(2,2), include.mean = TRUE))

# static fit and forecast
ar_static_fitx <- arfimafit(spec = specx, data = diff(log(nasdaq_for_roll$Adj.Clos
e))[-1], out.sample = T_tst)
ar_static_fore_logreturnsx <- xts(arfimaforecast(ar_static_fitx,
                                                 n.ahead = 1, n.roll = T_tst - 1)@
forecast$seriesFor[1, ],
                                  dates_out_of_sample)
```

```
# rolling fit and forecast
modelrollx <- arfimaroll(spec = specx, data = diff(log(nasdaq_for_roll$Adj.Close))
[-1], n.ahead = 1,
                         forecast.length = T_tst, refit.every = 50, refit.window =
"moving")

ar_rolling_fore_logreturnsx <- xts(modelrollx@forecast$density$Mu, dates_out_of_sa
mple)
error_logreturnsx <- cbind(return - ar_static_fore_logreturnsx,
                           return - ar_rolling_fore_logreturnsx)
names(error_logreturnsx) <- c("rolling forecast", "static forecast")

par(mfrow=c(2,1))
# plot of forecast
plot(cbind("static forecast"  = ar_static_fore_logreturnsx,
           "rolling forecast" = ar_rolling_fore_logreturnsx),
     col = c("black", "red"), lwd = 2,
     main = "Forecast with ARMA(2,2) model", legend.loc = "topleft")
plot(error_logreturnsx, col = c("black", "red"), lwd = 2,
     main = "Forecast error with ARMA(2,2) model", legend.loc = "topleft")
```

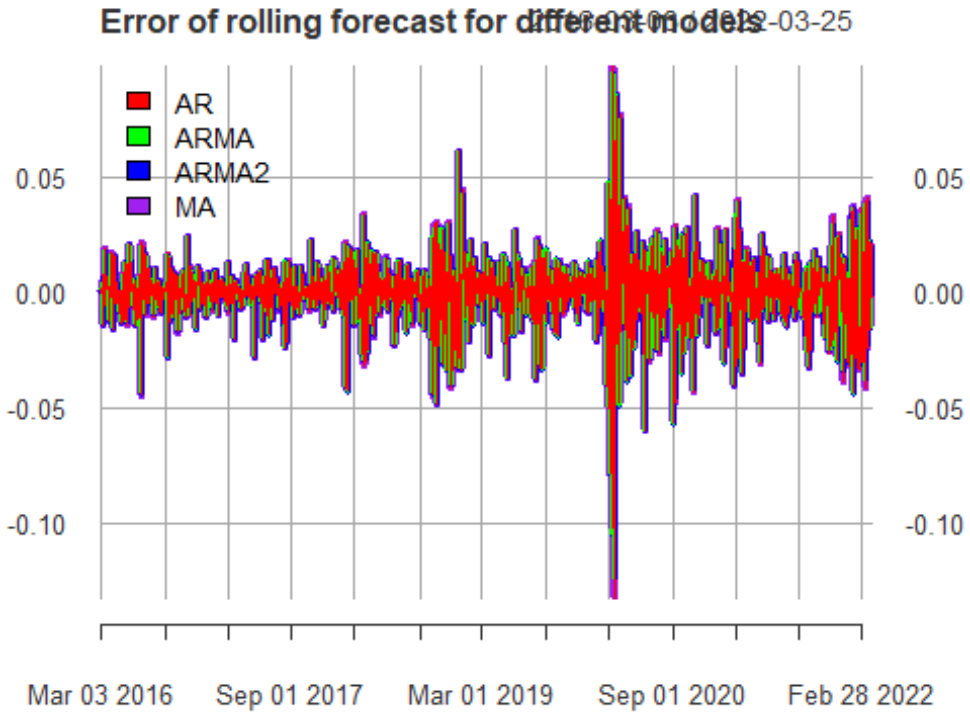


We can observe the effect of the rolling-window process in keeping track with the time series is (with surprise) very smooth.

```
# rolling forecast with AR(1) model
ar_rolling_fore_return <- xts(arfimaroll(ar_spec, data = diff(log(nasdaq_for_roll$
Adj.Close))[-1], n.ahead = 1, forecast.length = T_tst,
```

```
                                                refit.every = 50, refit.window = "mov
ing")@forecast$density$Mu,
                                    dates_out_of_sample)

# rolling forecast with ARMA(2,2) model
arma_rolling_fore_return <- xts(arfimaroll(arma_spec, data = diff(log(nasdaq_for_r
oll$Adj.Close))[-1], n.ahead = 1, forecast.length = T_tst,
                                                  refit.every = 50, refit.window = "m
oving")@forecast$density$Mu,
                                      dates_out_of_sample)

# rolling forecast with ARMA(2,1) model
arma_2_rolling_fore_return <- xts(arfimaroll(arma_spec, data = diff(log(nasdaq_for
_roll$Adj.Close))[-1], n.ahead = 1, forecast.length = T_tst,
                                               refit.every = 50, refit.window = "movin
g")@forecast$density$Mu,
                                    dates_out_of_sample)

# rolling forecast with MA(1) model
ma_rolling_fore_return <- xts(arfimaroll(ma_spec, data = diff(log(nasdaq_for_roll$
Adj.Close))[-1], n.ahead = 1, forecast.length = T_tst,
                                             refit.every = 50, refit.window = "moving"
)@forecast$density$Mu,
                                dates_out_of_sample)

rolling_error_var <- rbind("AR(1)"                  = c(var(return - fitted(ar_fit
)),
                                                        var(return - ar_rolling_fo
re_return)),
                           "ARMA(2,2)"              = c(var(return - fitted(arma_f
it)),
                                                        var(return - arma_rolling_
fore_return)),
                           "ARMA(2,1)"              = c(var(return - fitted(arma_f
it)),
                                                        var(return - arma_2_rollin
g_fore_return)),
                           "MA(1,0)"              = c(var(return - fitted(ma_fit))
,
                                                        var(return - ma_rolling_fo
re_return)))

colnames(rolling_error_var) <- c("in-sample", "out-of-sample")
print(rolling_error_var)

##              in-sample out-of-sample
## AR(1)     0.0001989092  0.0002099782
## ARMA(2,2) 0.0001984664  0.0002041272
## ARMA(2,1) 0.0001984664  0.0002041272
## MA(1,0)   0.0001988444  0.0002111230
```

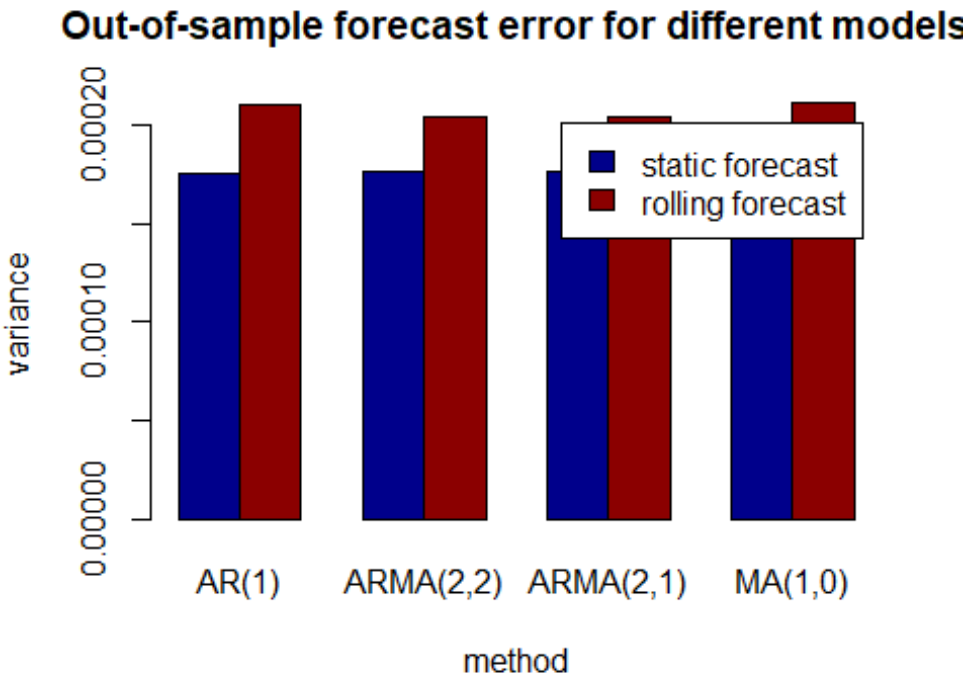
```
error_return <- cbind(return - ar_rolling_fore_return,
                          return - arma_rolling_fore_return,
                          return - arma_2_rolling_fore_return,
                      return - ma_rolling_fore_return)
names(error_return) <- c("AR", "ARMA", "ARMA2", 'MA')
plot(error_return, col = c("red", "green", "blue",'purple'), lwd = c(0.5, 2, 3,4),
     main = "Error of rolling forecast for different models", legend.loc = "toplef
t")
```



all the models should track better the time series. However, we do not observe any significant difference among the models and the methods

compare the static vs rolling basis errors:

```
barplot(rbind(error_var[, "out-of-sample"], rolling_error_var[, "out-of-sample"]),
        col = c("darkblue", "darkred"),
        legend = c("static forecast", "rolling forecast"),
        main = "Out-of-sample forecast error for different models",
        xlab = "method", ylab = "variance", beside = TRUE)
```



it confirms the previous results.

Mean model fit

#Another way to find the best model:

```
#library(SBAGM) #Lower AIC
ARIMAAIC(data = diff(log(nasdaq_adj))[-1], p=4, q=4, d=0, season=list(order=c(0,0,
0)))
```

```
##           q=0       q=1       q=2       q=3
## p=0 -29078.46 -29122.69 -29120.73 -29118.80
## p=1 -29122.56 -29120.72 -29118.73 -29116.76
## p=2 -29120.73 -29118.73 -29116.73 -29114.76
## p=3 -29118.74 -29116.73 -29114.73 -29115.88
```

```
#library(forecast)
auto.arima(y = diff(log(nasdaq_adj))[-1])
```

```
## Series: diff(log(nasdaq_adj))[-1]
## ARIMA(0,0,1) with non-zero mean
##
## Coefficients:
##           ma1   mean
##       -0.0952  4e-04
## s.e.   0.0139  2e-04
##
```

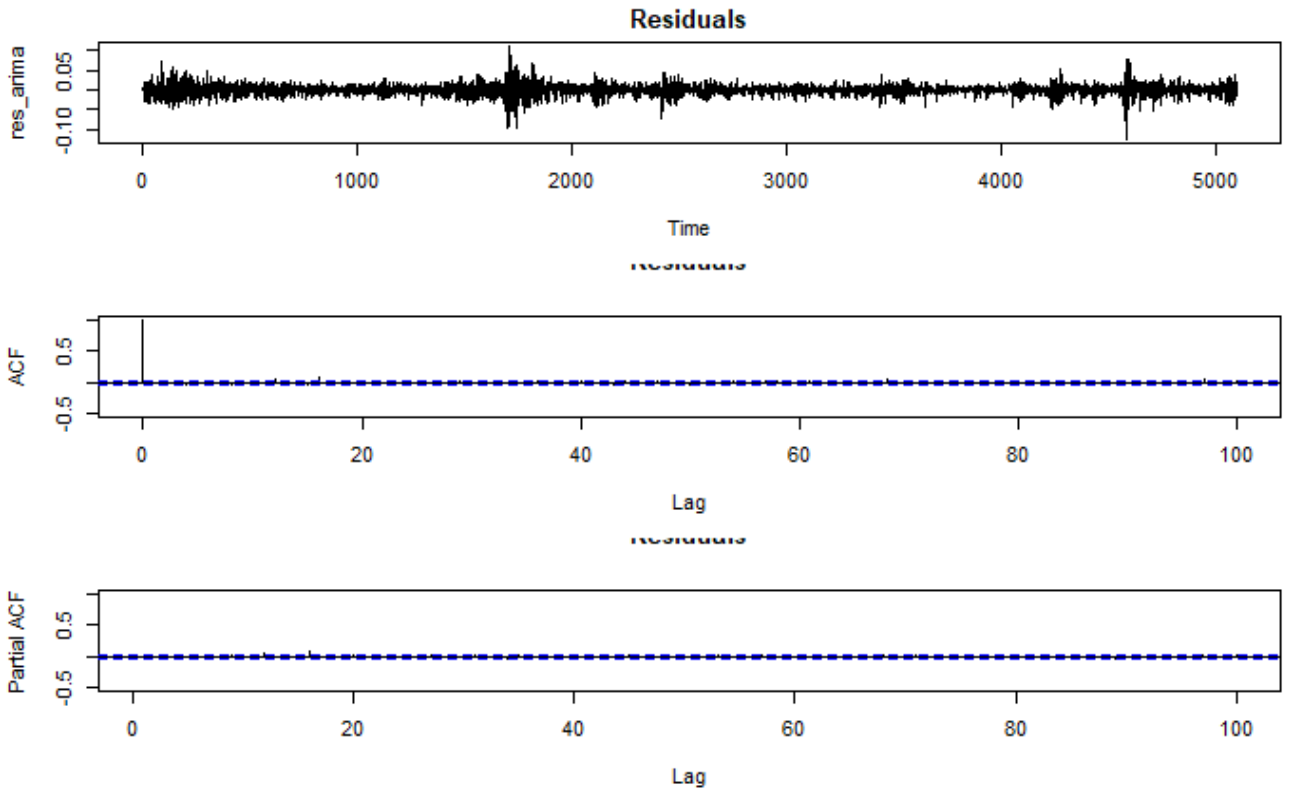
```
## sigma^2 = 0.000192:  log likelihood = 14564.34
## AIC=-29122.69   AICc=-29122.68   BIC=-29103.08
```

As the previous approach, AR(1) and MA(1) give us good fits, but the auto.arima and SBAGM functions have selected MA(1).

```
arima_mod <- arima(diff(log(nasdaq_adj))[-1],order=c(0,0,1))
```

residuals analysis

```
res_arima=arima_mod$res
par(mfcol=c(3,1))
plot(res_arima,main='Residuals')
acf_res=acf(res_arima,main='ACF
Residuals',lag.max=100,ylim=c(-0.5,1))
pacfres=pacf(res_arima,main='PACF
Residuals',lag.max=100,ylim=c(-0.5,1))
```



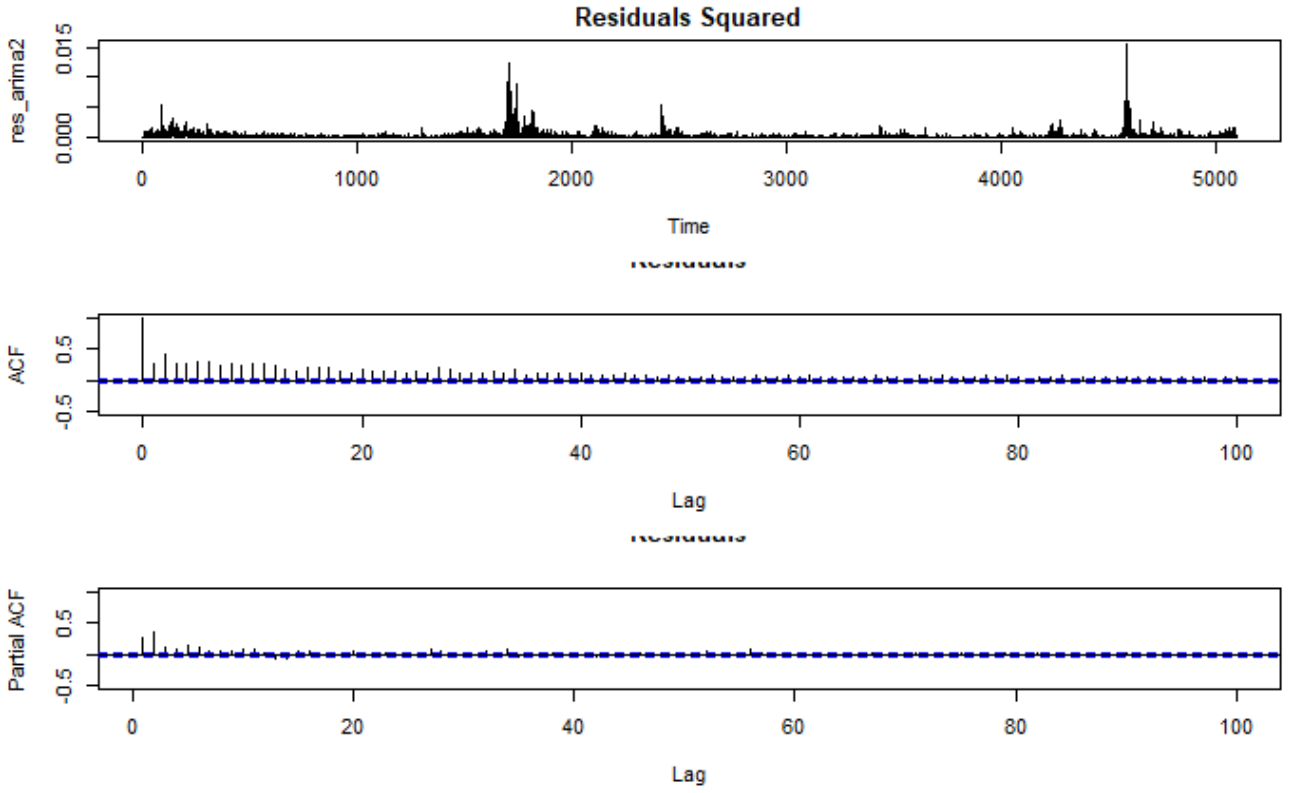
The residual plot, ACF and PACF do not have any significant lag, indicating ARIMA(1,1,0) is a good model to represent the series.

```
Box.test(res_arima, lag = c(2), type = c( "Ljung-Box"), fitdf = 1)

##
##  Box-Ljung test
##
## data:  res_arima
## X-squared = 0.051853, df = 1, p-value = 0.8199
```

```
Box.test(res_arima, lag = c(5), type = c( "Ljung-Box"), fitdf = 1)

##
##  Box-Ljung test
##
## data:  res_arima
## X-squared = 3.8253, df = 4, p-value = 0.4302

Box.test(res_arima, lag = c(10), type = c( "Ljung-Box"), fitdf = 1)

##
##  Box-Ljung test
##
## data:  res_arima
## X-squared = 22.842, df = 9, p-value = 0.006561
```

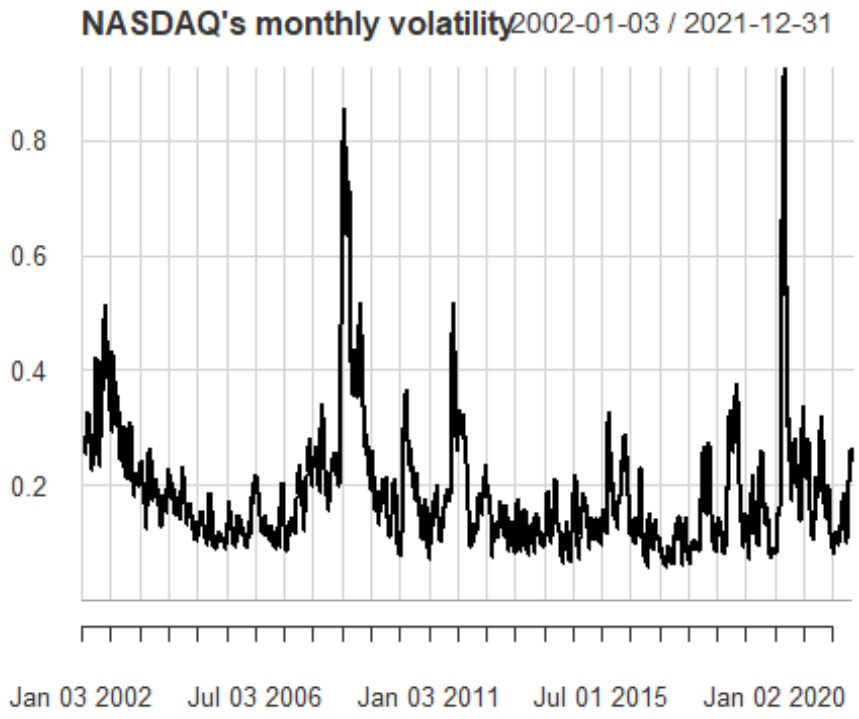
Furthermore, the Box test for the second, fifth and tenth(slightly) lag confirm the hypothesis of independently distributed data.

```
res_arima2=res_arima^2
par(mfcol=c(3,1))
plot(res_arima2,main='Residuals Squared')
acf_res2=acf(res_arima2,main='ACF Squared
Residuals',lag.max=100,ylim=c(-0.5,1))
pacfres2=pacf(res_arima2,main='PACF Squared
Residuals',lag.max=100,ylim=c(-0.5,1))
```



Squared residuals plot shows cluster of volatility PACF and ACF cuts off after lag 10 and 20 even though some remaining lags are significant Thus, there is presence of ARCH effect.

```
chart.RollingPerformance(R= return["2002::2021"],width = 22
                         ,FUN = "sd.annualized",scale=252,
                         main = "NASDAQ's monthly volatility")
```



We may need a stochastic model for conditional volatility. In the next session I will try to fit the best model to deal with the conditional volatility.

#conditional volatility modelling

Theoretical background variance models

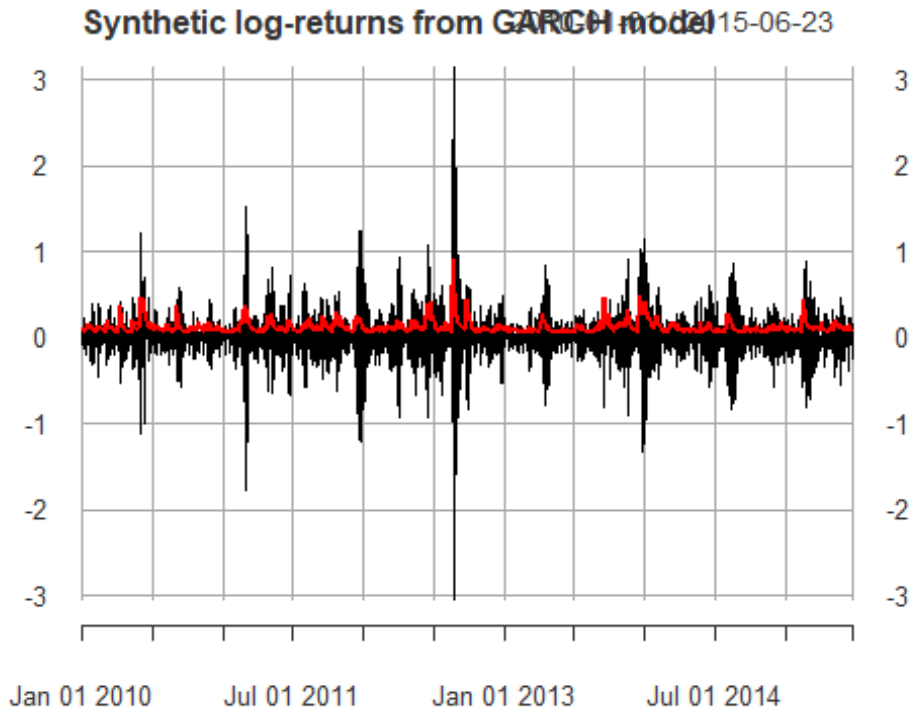
First i specified a GARCH(1,1) model with given coefficients and parameters

```
garch_fixed_spec <- ugarchspec(mean.model = list(armaOrder = c(1,0), include.mean
= TRUE),
                               variance.model = list(model = "sGARCH", garchOrder
= c(1,1)),
                               fixed.pars = list(mu = 0.005, ar1 = -0.9,
                                                 omega = 0.001, alpha1 = 0.3, beta
1 = 0.65))
true_params <- unlist(garch_fixed_spec@model$fixed.pars)
```

And I simulated a path and plotted it with synthetic log returns

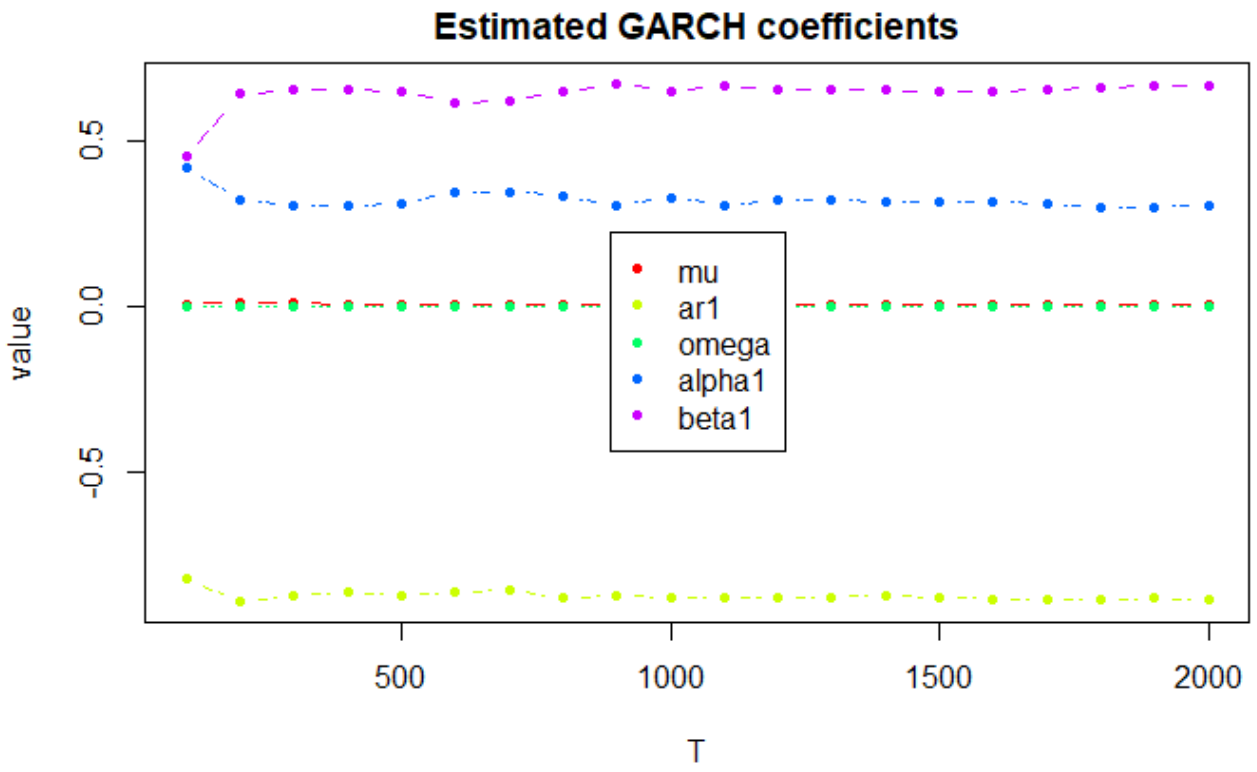
```
T <- 2000
path_garch <- ugarchpath(garch_fixed_spec, n.sim = T)

synth_log_returns <- xts(path_garch@path$seriesSim, order.by = as.Date("2010-01-01
") + 0:(T-1))
synth_volatility <- xts(path_garch@path$sigmaSim, order.by = as.Date("2010-01-01")
+ 0:(T-1))
{ plot(synth_log_returns, main = "Synthetic log-returns from GARCH model", lwd = 1
```

```
.5)
  lines(synth_volatility, col = "red", lwd = 2) }
```

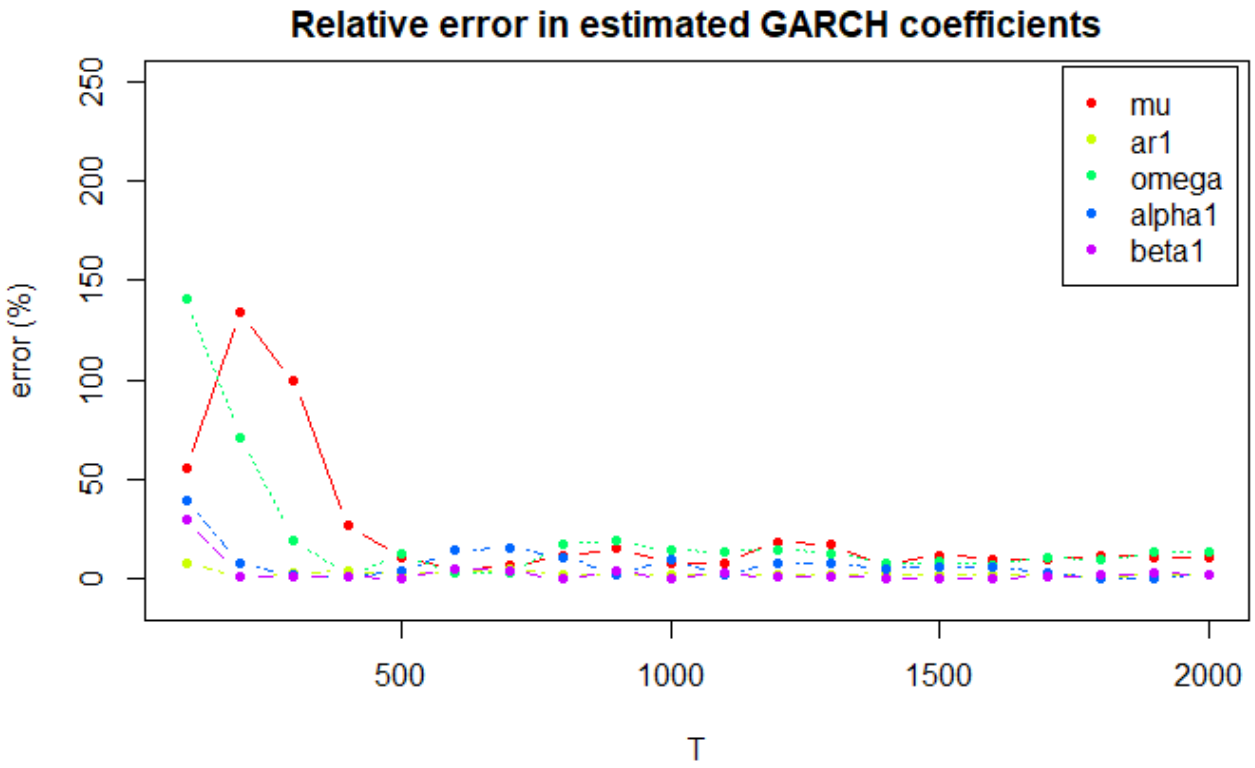


now specify a GARCH model

```
garch_spec <- ugarchspec(mean.model = list(armaOrder = c(1,0), include.mean = TRUE
),
                         variance.model = list(model = "sGARCH", garchOrder = c(1,
1)))
estim_coeffs_vs_T <- error_coeffs_vs_T <- NULL
T_sweep <- ceiling(seq(100, T, length.out = 20))
for (T_ in T_sweep) {
  garch_fit <- ugarchfit(spec = garch_spec, data = synth_log_returns[1:T_])
  error_coeffs_vs_T <- rbind(error_coeffs_vs_T, abs((coef(garch_fit) - true_params
)/true_params))
  estim_coeffs_vs_T <- rbind(estim_coeffs_vs_T, coef(garch_fit))
}
rownames(error_coeffs_vs_T) <- rownames(estim_coeffs_vs_T) <- paste("T =", T_sweep
)

matplot(T_sweep, estim_coeffs_vs_T,
        main = "Estimated GARCH coefficients", xlab = "T", ylab = "value",
        type = "b", pch = 20, col = rainbow(5))
legend("center", inset = 0.01, legend = colnames(estim_coeffs_vs_T), pch = 20, col
= rainbow(5))
```




```
matplot(T_sweep, 100*error_coeffs_vs_T,
        main = "Relative error in estimated GARCH coefficients", xlab = "T", ylab
= "error (%)",
        type = "b", pch = 20, col = rainbow(5), ylim=c(-10,250))
legend("topright", inset = 0.01, legend = colnames(error_coeffs_vs_T), pch = 20, c
ol = rainbow(5))
```



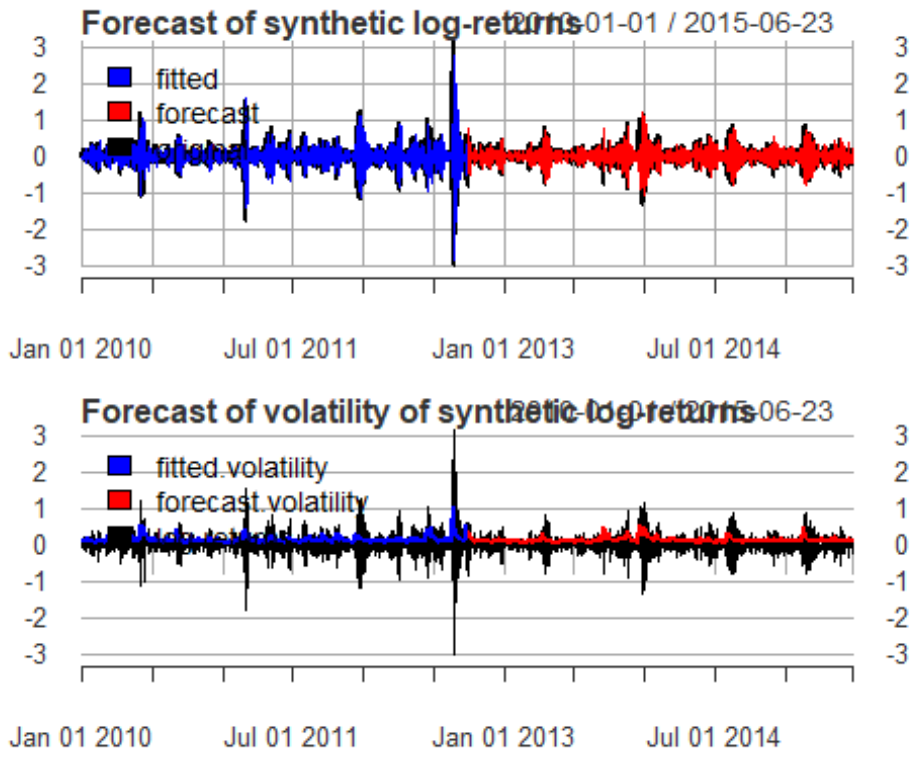
the coefficients seem to be well estimated after T samples

Once the parameters of the GARCH model have been estimated, one can use the model to forecast the values ahead.

Forecast:

```
# estimate model excluding the out-of-sample
out_of_sample <- round(T/2)
dates_out_of_sample <- tail(index(synth_log_returns), out_of_sample)
garch_spec <- ugarchspec(mean.model = list(armaOrder = c(1,0), include.mean = TRUE
),
                         variance.model = list(model = "sGARCH", garchOrder = c(1,
1)))
garch_fit <- ugarchfit(spec = garch_spec, data = synth_log_returns, out.sample = o
ut_of_sample)

# forecast log-returns along the whole out-of-sample
garch_fore <- ugarchforecast(garch_fit, n.ahead = 1, n.roll = out_of_sample-1)
forecast_log_returns <- xts(garch_fore@forecast$seriesFor[1, ], dates_out_of_sampl
e)
forecast_volatility <- xts(garch_fore@forecast$sigmaFor[1, ], dates_out_of_sample)

par(mfrow=c(2,1))
# plot of log-returns
plot(cbind("fitted"   = fitted(garch_fit),
           "forecast" = forecast_log_returns,
           "original" = synth_log_returns),
```

```
     col = c("blue", "red", "black"), lwd = c(0.5, 0.5, 2),
     main = "Forecast of synthetic log-returns", legend.loc = "topleft")
# plot of volatility log-returns
plot(cbind("fitted volatility"    = sigma(garch_fit),
           "forecast volatility" = forecast_volatility,
           "log-returns"          = synth_log_returns),
     col = c("blue", "red", "black"), lwd = c(2, 2, 1),
     main = "Forecast of volatility of synthetic log-returns", legend.loc = "tople
ft")
```



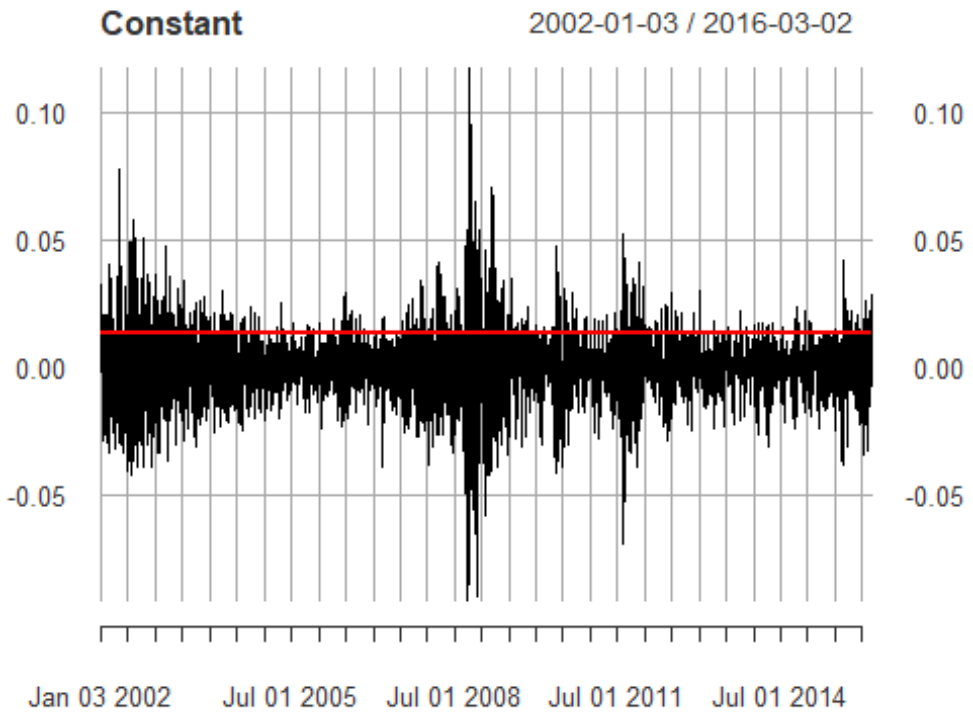
I estimated the model excluding the out-of-sample, then forecasted log-returns along the whole out-of-sample.

Different methods for conditional volatility

##modelling the Nasdaq's volatility different methods:

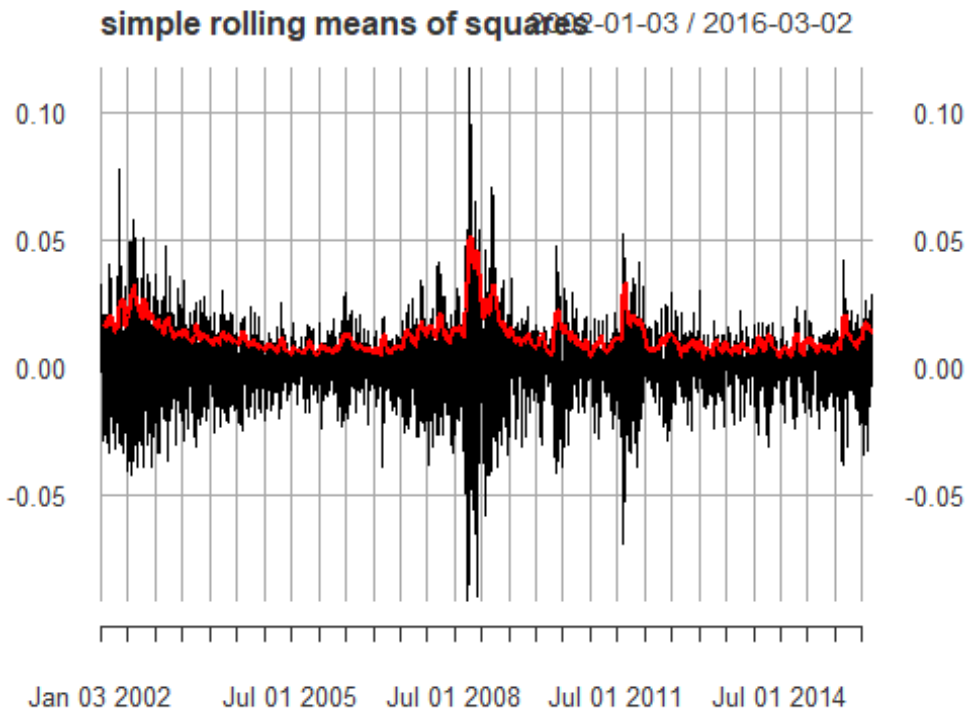
1. costant

```
var_constant <- var(return_trn)
plot(cbind(sqrt(var_constant), return_trn), col = c("red", "black"), lwd = c(2.5,
1.5),
     main = "Constant")
```



MA

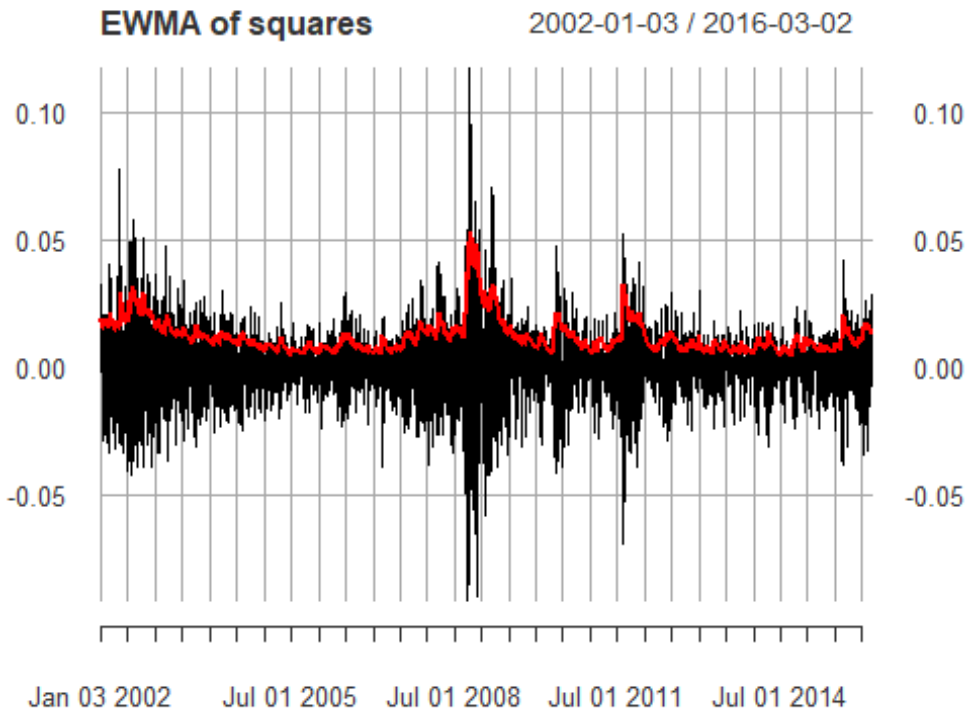
```
lookback_var <- 20
var_t <- roll_meanr(return_trn^2, n = lookback_var, fill = NA)
plot(cbind(sqrt(var_t), return_trn), col = c("red", "black"), lwd = c(2.5, 1.5),
     main = "simple rolling means of squares")
```




```
return_trn_std <- return_trn/sqrt(var_t)
var_ma <- var(return_trn_std, na.rm = TRUE) * tail(var_t, 1)
```

EWMA

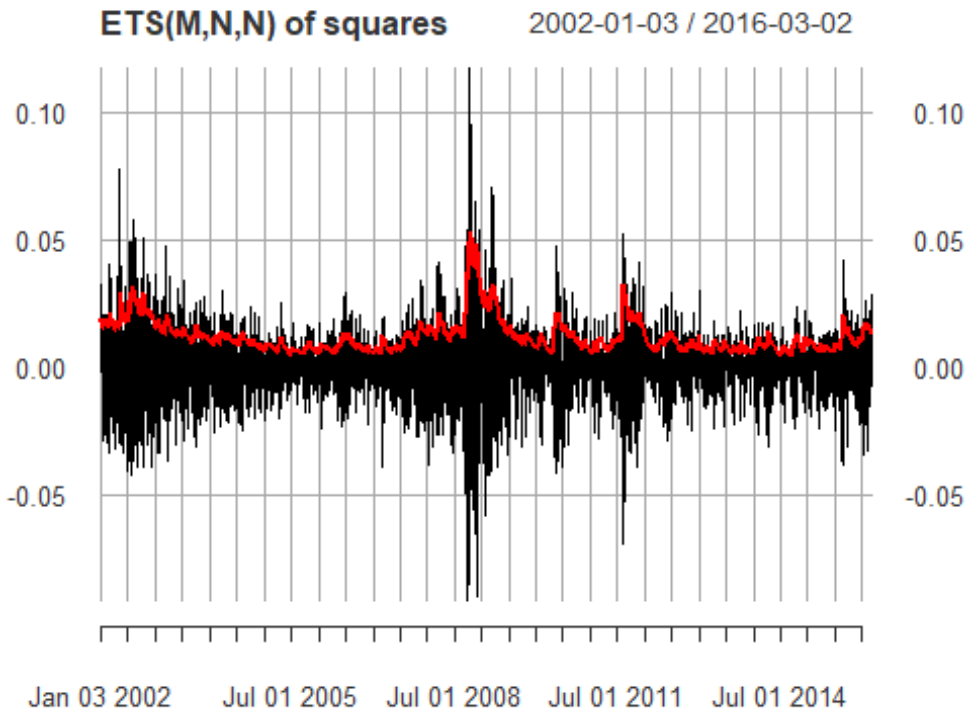
```
fit_ets <- ets(return_trn^2, model = "ANN")
std_t <- as.numeric(sqrt(fit_ets$fitted))
plot(cbind(std_t, return_trn), col = c("red", "black"), lwd = c(2.5, 1.5),
     main = "EWMA of squares")
```




```
return_trn_std_2 <- return_trn/std_t
var_ewma <- var(return_trn_std_2, na.rm = TRUE) * tail(std_t, 1)^2
```

Multiplicative ETS

```
fit_ets2 <- ets(1e-6 + return_trn^2, model = "MNN")
std_t2 <- as.numeric(sqrt(fit_ets2$fitted))
plot(cbind(std_t, return_trn), col = c("red", "black"), lwd = c(2.5, 1.5),
     main = "ETS(M,N,N) of squares")
```




```
return_trn_std_3 <- return_trn/std_t2
var_ets_mnn <- var(return_trn_std_3, na.rm = TRUE) * tail(std_t2, 1)^2
```

ETS (Error, Trend, Seasonal)method is an approach method for forecasting time series univariate. ETS model focuses on trend and seasonal components. The flexibility of the ETS model lies in its ability to trend and seasonal components of different traits. In this paper it's used only for illustration purposes, it's not further covered by this work

here i use the "appgarch" algorithm to find the best GARCH(,) + dist combination

```
appgarch_1 <- appgarch(data = return, methods = c('sGARCH', 'eGARCH', 'gjrGARCH'),
distributions =  c("norm", "std","sstd", "snorm",'ged'),
         aorder = c(0,1), gorder = c(1, 1), stepahead = 2 )
#eGARCH(1,1) with snorm distribution
appgarch_1

## $rmse_mean
##                norm        std       sstd      snorm        ged
## sGARCH   0.01280471 0.01266372 0.01264385 0.01263505 0.01267901
## eGARCH   0.01314183 0.01291799 0.01301078 0.01306634 0.01291860
## gjrGARCH 0.01310694 0.01291799 0.01297889 0.01301380 0.01291667
##
## $mae_mean
##                norm        std       sstd      snorm        ged
## sGARCH   0.01018926 0.01018908 0.01006424 0.01003220 0.01020496
## eGARCH   0.01023014 0.01023245 0.01015853 0.01012516 0.01024431
## gjrGARCH 0.01023609 0.01024236 0.01015405 0.01012270 0.01025683
```

```
##
## $forecast_mean
##      sGARCH-norm  sGARCH-std  sGARCH-sstd sGARCH-snorm  sGARCH-ged  eGARCH-norm
## T+1 0.0013941347 0.001629151 0.0016204842 0.0016250429 0.001609034 0.0008588628
## T+2 0.0008460608 0.001080712 0.0008223621 0.0007628482 0.001092364 0.0003925498
##       eGARCH-std  eGARCH-sstd eGARCH-snorm   eGARCH-ged gjrGARCH-norm
## T+1 0.0012210235 0.0010505833 0.0009542401 0.0012235817  0.0009160329
## T+2 0.0007593347 0.0004410439 0.0002779707 0.0007856046  0.0004616043
##     gjrGARCH-std gjrGARCH-sstd gjrGARCH-snorm gjrGARCH-ged
## T+1 0.0012239911  0.0011005645   0.0010370364 0.0012305476
## T+2 0.0007821032  0.0004820636   0.0003558419 0.0008176138
##
## $forecast_sigma
##     sGARCH-norm sGARCH-std sGARCH-sstd sGARCH-snorm sGARCH-ged eGARCH-norm
## T+1  0.01988816 0.02061097  0.02062876   0.02008425 0.02021335  0.01776043
## T+2  0.01978605 0.02058334  0.02058690   0.01998632 0.02014814  0.01757892
##     eGARCH-std eGARCH-sstd eGARCH-snorm eGARCH-ged gjrGARCH-norm gjrGARCH-std
## T+1 0.01822851  0.01812509   0.01787575 0.01803635    0.01691592   0.01691819
## T+2 0.01800036  0.01793108   0.01771687 0.01779125    0.01681810   0.01683897
##     gjrGARCH-sstd gjrGARCH-snorm gjrGARCH-ged
## T+1    0.01696756     0.01703881   0.01690764
## T+2    0.01686481     0.01692407   0.01680765
```

```
appgarch_2 <- appgarch(data = return, methods = c('sGARCH', 'eGARCH', 'gjrGARCH'),
distributions =  c("norm", "std","sstd", "snorm",'ged'),
         aorder = c(0,1), gorder = c(2, 1), stepahead = 2 )
appgarch_2
```

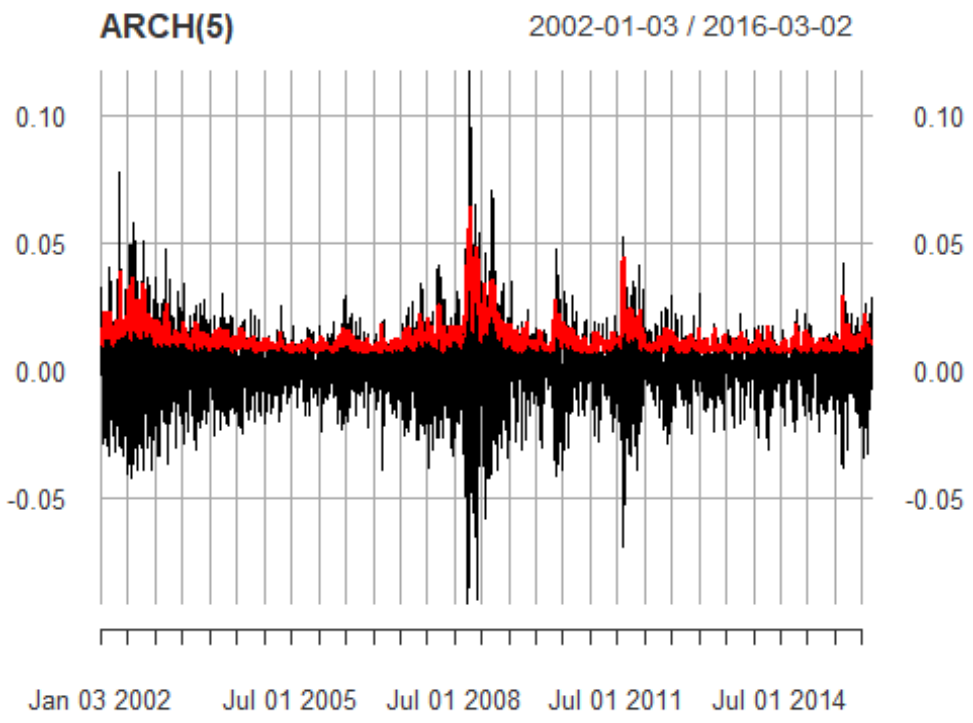
```
## $rmse_mean
##                norm        std       sstd      snorm        ged
## sGARCH   0.01282226 0.01267170 0.01264584 0.01264653 0.01269115
## eGARCH   0.01303188 0.01279397 0.01300566 0.01306762 0.01279891
## gjrGARCH 0.01307939 0.01289029 0.01293950 0.01298031 0.01289348
##
## $mae_mean
##                norm        std       sstd      snorm        ged
## sGARCH   0.01019558 0.01018952 0.01005931 0.01003401 0.01020772
## eGARCH   0.01016173 0.01015717 0.01016290 0.01012935 0.01016948
## gjrGARCH 0.01022309 0.01022777 0.01013247 0.01010509 0.01024497
##
## $forecast_mean
##      sGARCH-norm  sGARCH-std  sGARCH-sstd sGARCH-snorm  sGARCH-ged  eGARCH-norm
## T+1 0.0013671500 0.001615860 0.0016156509 0.0016067254 0.001589563 0.0010176413
## T+2 0.0008317025 0.001068294 0.0008076689 0.0007481441 0.001078409 0.0004145016
##       eGARCH-std  eGARCH-sstd eGARCH-snorm   eGARCH-ged gjrGARCH-norm
## T+1 0.0014018813 0.0010598725 0.0009531664 0.0013975377  0.0009569060
## T+2 0.0007896317 0.0004590677 0.0002852579 0.0008098946  0.0004764823
##     gjrGARCH-std gjrGARCH-sstd gjrGARCH-snorm gjrGARCH-ged
## T+1  0.001265070  0.0011582949   0.0010861489 0.0012650873
## T+2  0.000794015  0.0004966341   0.0003697235 0.0008284343
##
## $forecast_sigma
```

```
##      sGARCH-norm sGARCH-std sGARCH-sstd sGARCH-snorm sGARCH-ged eGARCH-norm
## T+1  0.01999185 0.02077562  0.02087408   0.02025395 0.02032970  0.01849086
## T+2  0.01944968 0.02021708  0.02026550   0.01965940 0.01978522  0.01761937
##      eGARCH-std eGARCH-sstd eGARCH-snorm eGARCH-ged gjrGARCH-norm gjrGARCH-std
## T+1 0.01919160  0.01898025   0.01857086 0.01897107    0.01705878   0.01722670
## T+2 0.01801074  0.01791341   0.01771680 0.01788651    0.01685871   0.01708461
##      gjrGARCH-sstd gjrGARCH-snorm gjrGARCH-ged
## T+1    0.01721300     0.01712078   0.01716232
## T+2    0.01705167     0.01691100   0.01697966
```

the appgarch() function suggests eGARCH(1,1) model with skew normal distribution (snorm)

ARCH(5)

```
arch_fit <- fGarch::garchFit(formula = ~ garch(5,0), return_trn, trace = FALSE)

## Warning: Using formula(x) is deprecated when x is a character vector of length
> 1.
##   Consider formula(paste(x, collapse = " ")) instead.

std_t4 <- arch_fit@sigma.t
plot(cbind(std_t4, return_trn), col = c("red", "black"), lwd = c(2.5, 1.5),
     main = "ARCH(5)")
```

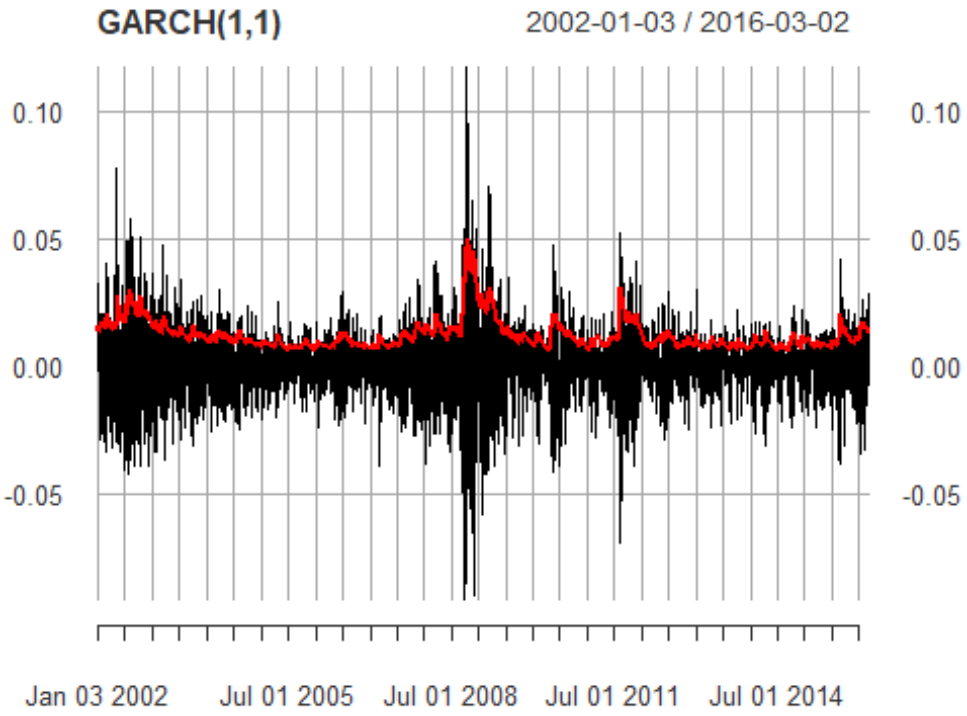



```
var_arch <- tail(std_t4, 1)^2
```

GARCH(1,1)

```
garch_fit <- fGarch::garchFit(formula = ~ garch(1,1), return_trn, trace = FALSE)
```

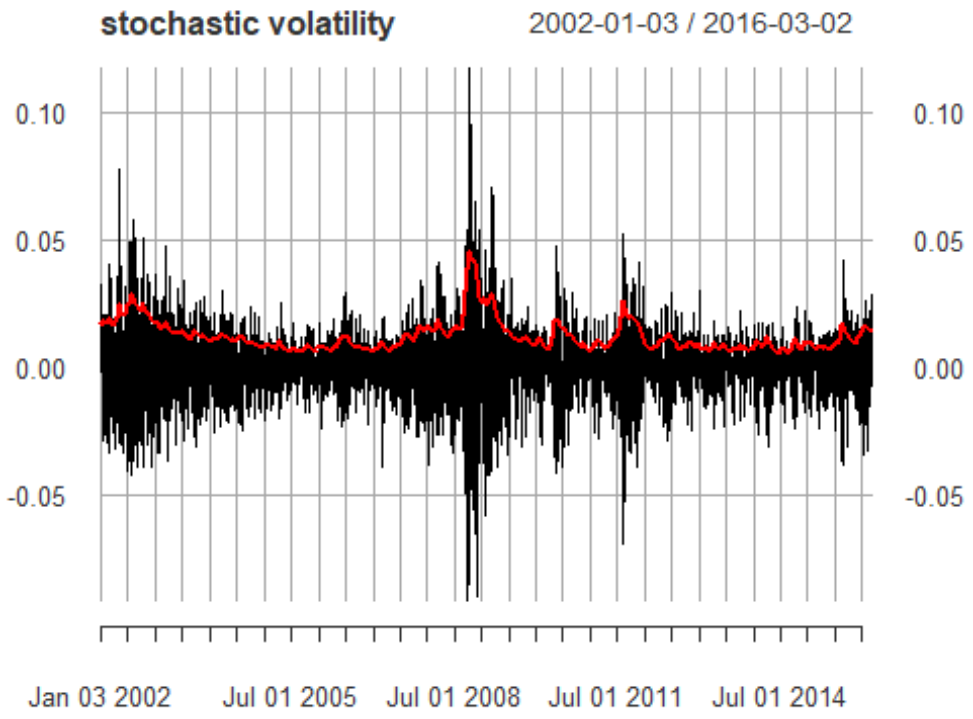
```
std_t5 <- garch_fit@sigma.t
plot(cbind(std_t5, return_trn), col = c("red", "black"), lwd = c(2.5, 1.5),
     main = "GARCH(1,1)")
```




```
var_garch <- tail(std_t5, 1)^2
```

SV

```
res_sv <- svsample(return_trn - mean(return_trn))

## Done!

## Summarizing posterior draws...

std_t_sv <- res_sv$summary$sd[, 1]
plot(cbind(std_t_sv, return_trn), col = c("red", "black"), lwd = c(2.5, 1.5),
     main = "stochastic volatility")
```




```
var_sv <- tail(std_t_sv, 1)^2
```

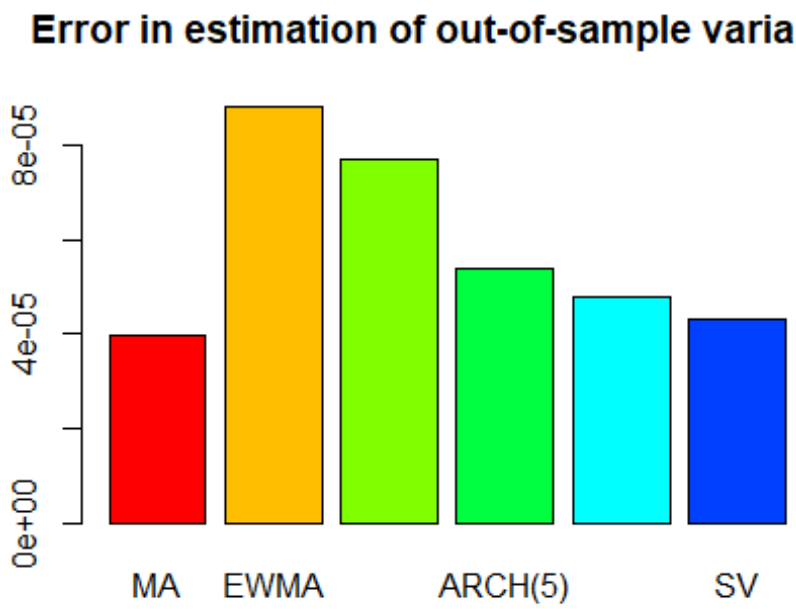
stochastic volatility is not covered by this work, however it has been included for informational purposes

I compared the error in the estimation of the variance by each method for the out-of-sample period

```
error_all <- c("MA"          = abs(var_ma      - var(return_tst)),
               "EWMA"        = abs(var_ewma    - var(return_tst)),
               "ETS(M,N,N)" = abs(var_ets_mnn - var(return_tst)),
               "ARCH(5)"     = abs(var_arch    - var(return_tst)),
               "GARCH(1,1)" = abs(var_garch   - var(return_tst)),
               "SV"          = abs(var_sv      - var(return_tst)))

print(error_all)
```

```
##           MA         EWMA   ETS(M,N,N)      ARCH(5)   GARCH(1,1)           SV
## 3.960304e-05 8.798757e-05 7.679852e-05 5.383680e-05 4.777268e-05 4.319431e-05
```

```
barplot(error_all, main = "Error in estimation of out-of-sample variance", col = r
ainbow(8))
```



as i expected the SV seems to be the best in describing the conditional variance.

Rolling-Window comparison volatility models

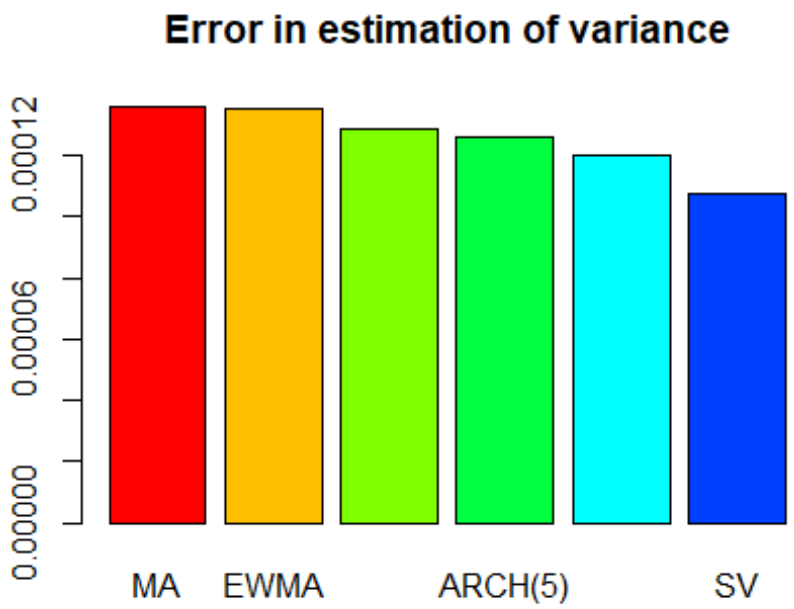
Then, i used a rolling-window comparison of six methods: MA, EWMA, ETS(MNN), ARCH(5), GARCH(1,1), and SV

```
error_sv <- error_garch <- error_garch2 <- error_arch <- error_ets_mnn <- error_ew
ma <- error_ma <- NULL

lookback_ <- 200
len_tst <- 40
for (i in seq(lookback_, T-len_tst, by = len_tst)) {
  return_trn <- return[(i-lookback_+1):i]
  return_tst <- return[(i+1):(i+len_tst)]
  var_tst <- var(return_tst)

  # MA
  var_t <- roll_meanr(return_trn^2,n =20,fill = NA )
  var_fore <- var(return_trn/sqrt(var_t), na.rm = TRUE) * tail(var_t, 1)
  error_ma <- c(error_ma, abs(var_fore - var_tst))

  # EWMA
  fit_ets <- ets(return_trn^2, model = "ANN")
  std_t <- as.numeric(sqrt(fit_ets$fitted))
  var_fore <- var(return_trn/std_t, na.rm = TRUE) * tail(std_t, 1)^2
```

```r
  error_ewma <- c(error_ewma, abs(var_fore - var_tst))

  # ETS(M,N,N)
  fit_ets <- ets(1e-6 + return_trn^2, model = "MNN")
  std_t <- as.numeric(sqrt(fit_ets$fitted))
  var_fore <- var(return_trn/std_t, na.rm = TRUE) * tail(std_t, 1)^2
  error_ets_mnn <- c(error_ets_mnn, abs(var_fore - var_tst))

  # ARCH
  arch_fit <- fGarch::garchFit(formula = ~ garch(5,0), return_trn, trace = FALSE)
  std_t <- as.numeric(arch_fit@sigma.t)
  var_fore <- var(return_trn/std_t, na.rm = TRUE) * tail(std_t, 1)^2
  error_arch <- c(error_arch, abs(var_fore - var_tst))

  # GARCH
  garch_fit <- fGarch::garchFit(formula = ~ garch(1,1), return_trn, trace = FALSE)
  std_t <- as.numeric(garch_fit@sigma.t)
  var_fore <- var(return_trn/std_t, na.rm = TRUE) * tail(std_t, 1)^2
  error_garch <- c(error_garch, abs(var_fore - var_tst))

  # SV
  res <- svsample(return_trn - mean(return_trn))
  std_t <- res$summary$sd[, 1]
  var_fore <- var(return_trn/std_t, na.rm = TRUE) * tail(std_t, 1)^2
  error_sv <- c(error_sv, abs(var_fore - var_tst))
}

error_all <- c("MA"         = mean(error_ma),
               "EWMA"       = mean(error_ewma),
               "ETS(M,N,N)" = mean(error_ets_mnn),
               "ARCH(5)"    = mean(error_arch),
               "GARCH(1,1)" = mean(error_garch),
               "SV"         = mean(error_sv))
print(error_all)
```

```
##           MA         EWMA   ETS(M,N,N)      ARCH(5)   GARCH(1,1)           SV
## 0.0001357935 0.0001352626 0.0001288716 0.0001259760 0.0001201972 0.0001075741
```

```r
barplot(error_all, main = "Error in estimation of variance", col = rainbow(8))
```



the SV keeps the first position in being the best method for describing conditional variance. in this paper i will continue with GARCH modelling which gives a good solution in dealing with conditional variance

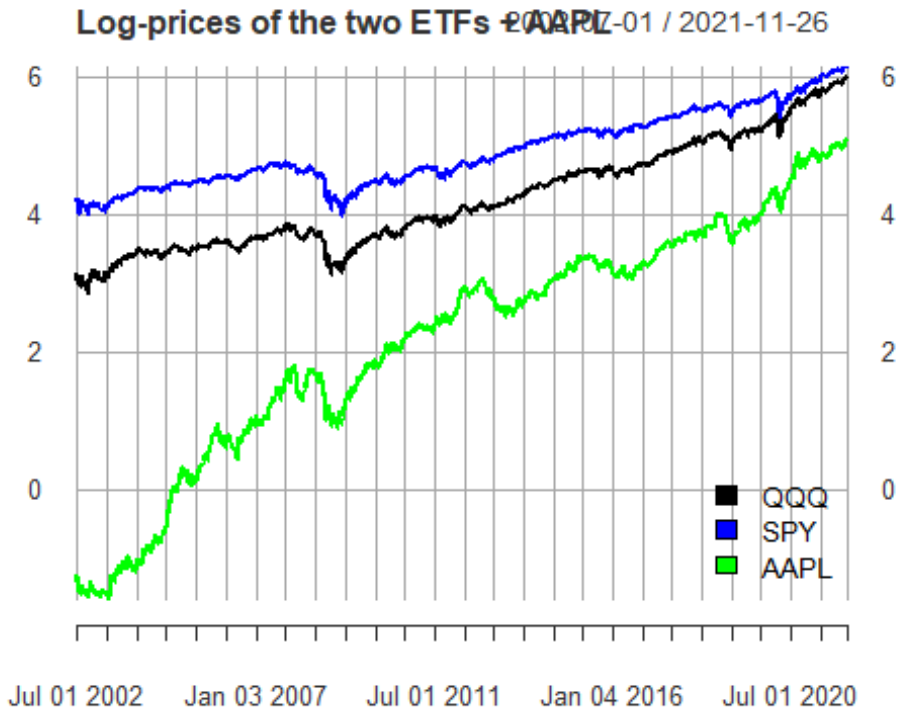
Multivariate GARCH model

#Multivariate GARCH (just a look at the correlations)

I loaded some multivariate ETF data [QQQ, SPY and AAPL stock] from Yahoo Finance.

```
stock_namelist <- c("QQQ", "SPY", "AAPL")

# download data from YahooFinance
prices <- xts()
for (stock_index in 1:length(stock_namelist)){
  prices <- cbind(prices, Ad(getSymbols(stock_namelist[stock_index],
                                        from= "2002-07-01",to= "2021-11-28", auto.
assign = FALSE)))}
colnames(prices) <- stock_namelist
indexClass(prices) <- "Date"

## Warning: 'indexClass<-' is deprecated.
## Use 'tclass<-' instead.
## See help("Deprecated") and help("xts-deprecated").

logreturns <- diff(log(prices))[-1]
```

```
# plot the four series of log-prices
plot(log(prices), col = c("black", "blue", "green",'pink'),
     main = "Log-prices of the two ETFs + AAPL", legend.loc = "bottomright")
```



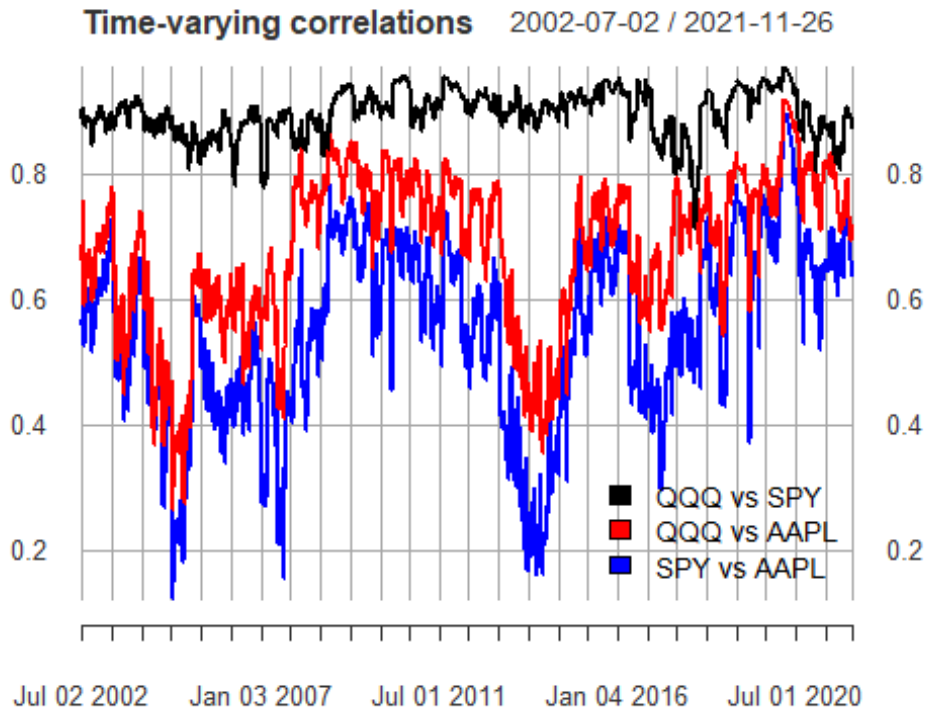
First i specify iid model for the univariate time series, then I specify the DCC model and after that I fit the model

```
# specify i.i.d. model for the univariate time series
ugarch_spec <- ugarchspec(mean.model = list(armaOrder = c(0,0),
               include.mean = FALSE), variance.model = list(model = "sGARCH",
                                          garchOrder = c(1,1)))

# specify DCC model
dcc_spec <- dccspec(uspec = multispec(replicate(ugarch_spec, n = 3)),
                    VAR = TRUE, lag = 3,
                    model = "DCC", dccOrder = c(1,1))
# estimate model
garchdcc_fit <- suppressWarnings(dccfit(dcc_spec, data = logreturns, solver = "nlm
inb"))
```

I extract the time-varying covariance and correlation matrix

```
dcc_cor <- rcor(garchdcc_fit)
dim(dcc_cor)

## [1]    3    3 4887

corr_t <- xts(cbind(dcc_cor[1, 2, ], dcc_cor[1, 3, ], dcc_cor[2, 3, ]), order.by =
index(logreturns))
```

```
colnames(corr_t) <- c("QQQ vs SPY", "QQQ vs AAPL", "SPY vs AAPL")
plot(corr_t, col = c("black", "red", "blue"),
     main = "Time-varying correlations", legend.loc = "bottomright")
```



We see the correlation between the two ETFs is extremely high and quite stable. The correlation between AAPL and the ETFs is cyclical, shifting between 0.8 and 0.4

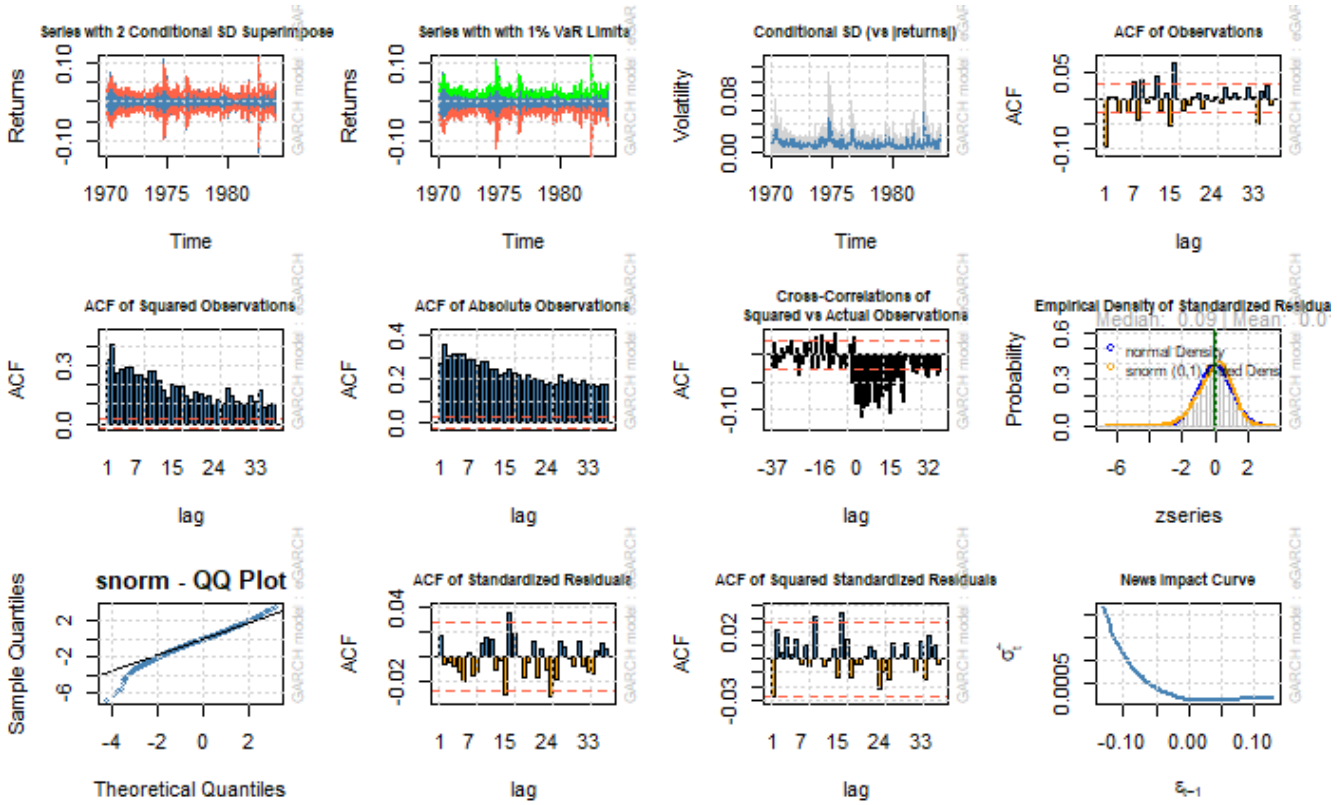
Model specification and fit

```
model_specify <- ugarchspec(mean.model = list(armaOrder=c(0,1), include.mean = TRU
E),
                            variance.model = list(model="eGARCH",
                                                  garchOrder=c(1,1)),
                            distribution.model = "snorm")
model_fit <- ugarchfit(data = diff(log(nasdaq_for_roll$Adj.Close))[-1],spec = mode
l_specify)
print(model_fit)

##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : eGARCH(1,1)
## Mean Model   : ARFIMA(0,0,1)
## Distribution : snorm
```

```
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.000198    0.000136    1.4631  0.14344
## ma1    -0.055044    0.015953   -3.4505  0.00056
## omega  -0.231854    0.002219 -104.5015  0.00000
## alpha1 -0.122486    0.005683  -21.5521  0.00000
## beta1   0.973492    0.000364 2673.7185  0.00000
## gamma1  0.141123    0.001885   74.8504  0.00000
## skew    0.800409    0.014777   54.1656  0.00000
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.000198    0.000175    1.1303 0.258351
## ma1    -0.055044    0.015321   -3.5928 0.000327
## omega  -0.231854    0.007207  -32.1695 0.000000
## alpha1 -0.122486    0.008573  -14.2871 0.000000
## beta1   0.973492    0.000591 1648.2570 0.000000
## gamma1  0.141123    0.010403   13.5652 0.000000
## skew    0.800409    0.018352   43.6151 0.000000
##
## LogLikelihood : 15729.09
##
## Information Criteria
## ------------------------------------
##
## Akaike       -6.1740
## Bayes        -6.1650
## Shibata      -6.1740
## Hannan-Quinn -6.1709
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      1.619  0.2032
## Lag[2*(p+q)+(p+q)-1][2]     1.712  0.3290
## Lag[4*(p+q)+(p+q)-1][5]     2.079  0.6885
## d.o.f=1
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      3.720 0.05377
## Lag[2*(p+q)+(p+q)-1][5]     6.238 0.07900
## Lag[4*(p+q)+(p+q)-1][9]     7.485 0.16182
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
```

```
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.1154 0.500 2.000  0.7341
## ARCH Lag[5]    1.1551 1.440 1.667  0.6875
## ARCH Lag[7]    1.9140 2.315 1.543  0.7355
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  5.4873
## Individual Statistics:
## mu     0.16639
## ma1    0.07691
## omega  0.36506
## alpha1 0.18691
## beta1  0.30513
## gamma1 0.22965
## skew   2.57418
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.69 1.9 2.35
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value     prob sig
## Sign Bias            1.903 0.057156   *
## Negative Sign Bias   1.801 0.071717   *
## Positive Sign Bias   1.833 0.066867   *
## Joint Effect        13.215 0.004194 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     82.91    5.834e-10
## 2    30     96.34    3.735e-09
## 3    40    108.28    1.951e-08
## 4    50    126.49    8.973e-09
##
##
## Elapsed time : 0.7504511
```

```
par(mfrow = c(3, 4))
for (i in 1:12) {
  plot(model_fit, which = i)
}
```

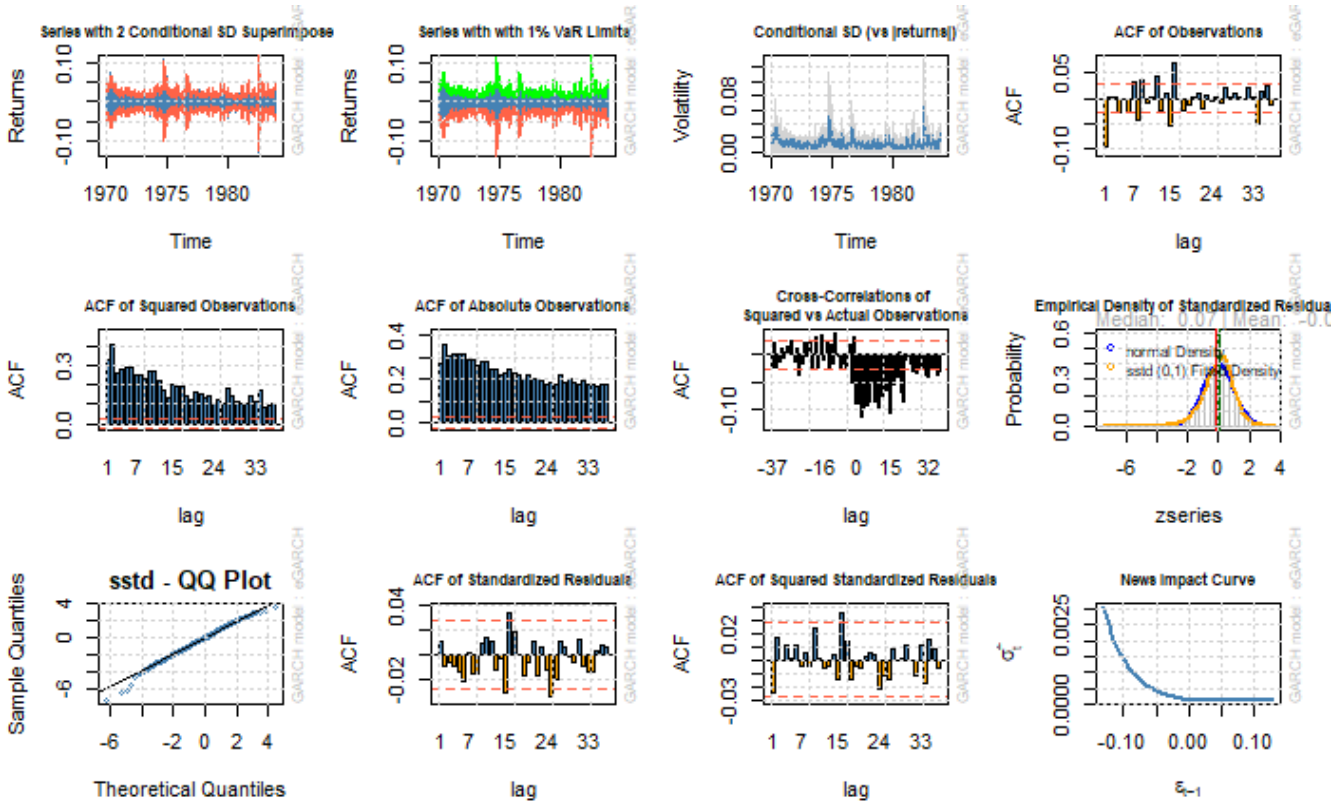



```
model_specify3 <- ugarchspec(mean.model = list(armaOrder=c(0,1), include.mean = TR
UE),
                             variance.model = list(model="eGARCH",
                                                   garchOrder=c(1,1)),
                             distribution.model = "sstd")
model_fit3 <- ugarchfit(data = diff(log(nasdaq_for_roll$Adj.Close))[-1],spec = mod
el_specify3)
print(model_fit3)

##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : eGARCH(1,1)
## Mean Model   : ARFIMA(0,0,1)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error    t value Pr(>|t|)
## mu      0.000392    0.000122     3.2166 0.001297
## ma1    -0.048334    0.014064    -3.4367 0.000589
## omega  -0.210633    0.002761   -76.2810 0.000000
## alpha1 -0.140512    0.007253   -19.3722 0.000000
```

```
## beta1   0.976552    0.000025 39355.7694 0.000000
## gamma1  0.144602    0.007111    20.3358 0.000000
## skew    0.822723    0.016302    50.4686 0.000000
## shape   8.595820    0.995034     8.6387 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error    t value Pr(>|t|)
## mu      0.000392    0.000121     3.2376 0.001205
## ma1    -0.048334    0.011972    -4.0373 0.000054
## omega  -0.210633    0.002855   -73.7875 0.000000
## alpha1 -0.140512    0.008355   -16.8187 0.000000
## beta1   0.976552    0.000026 37642.8096 0.000000
## gamma1  0.144602    0.008143    17.7579 0.000000
## skew    0.822723    0.016172    50.8728 0.000000
## shape   8.595820    1.058131     8.1236 0.000000
##
## LogLikelihood : 15787.07
##
## Information Criteria
## ------------------------------------
##
## Akaike       -6.1964
## Bayes        -6.1861
## Shibata      -6.1964
## Hannan-Quinn -6.1928
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.5803  0.4462
## Lag[2*(p+q)+(p+q)-1][2]    0.7903  0.8562
## Lag[4*(p+q)+(p+q)-1][5]    1.3396  0.8838
## d.o.f=1
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      3.122 0.07723
## Lag[2*(p+q)+(p+q)-1][5]     4.620 0.18621
## Lag[4*(p+q)+(p+q)-1][9]     5.299 0.38687
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3] 0.0007376 0.500 2.000  0.9783
## ARCH Lag[5] 0.5412769 1.440 1.667  0.8714
## ARCH Lag[7] 0.9966121 2.315 1.543  0.9141
##
## Nyblom stability test
```

```
## ------------------------------------
## Joint Statistic:  6.8131
## Individual Statistics:
## mu     0.1984
## ma1    0.1556
## omega  0.8033
## alpha1 0.9552
## beta1  0.6915
## gamma1 0.3379
## skew   2.3596
## shape  1.1097
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.89 2.11 2.59
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value     prob sig
## Sign Bias            1.364 0.172554
## Negative Sign Bias   2.289 0.022120  **
## Positive Sign Bias   2.248 0.024591  **
## Joint Effect        13.476 0.003713 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     29.53      0.05816
## 2    30     36.92      0.14835
## 3    40     49.53      0.12029
## 4    50     55.59      0.24061
##
##
## Elapsed time : 1.306241
```

```r
par(mfrow = c(3, 4))
for (i in 1:12) {
  plot(model_fit3, which = i)
}
```



Graphs: The sstd gives a better distribution based on QQ plot, both models are good in describing the news impact curve (the asymmetrical behavior of the negative news). There's absence of autocorrelation in the Squared Standardized residuals' ACF.

the arch LM test (in both models) shows that the series of residuals exhibits no conditional heteroscedasticity the Ljung-Box Test on Standardized (and Squared) Residuals shows that there is No serial correlation in the models, the eGARCH with sstd distribution shows better pvalues.

the Adjusted Pearson Goodness-of-Fit Test: it tests the null hypothesis that the data comes from a specified distribution: the model with skew Student-t distribution dosen't refuse the null hypothesis while the skew normal does.

In addition to that, the eGARCHwith sstd dist has the higher the value of the log-likelihood and Lower AIC score.

Markov-Switching model

The appmsgarch function computes the root mean square error (RMSE) and mean absolute error (MAE) of the different possible combinations of methods and distributions of the MS-GARCH model

```
appmsgarch_1 <- appmsgarch(data =return_for_msgarch, methods = c( "sGARCH", "eGARCH", "tGARCH"),
           distributions = c("snorm", "sstd",'ged'),
           stepahead = 2 )  #package SBAGM
appmsgarch_1
```

```
## $rmse_mat
##                 snorm-snorm   snorm-sstd    snorm-ged    sstd-sstd     sstd-ged
## sGARCH-sGARCH 0.0002599888 0.0002903842 0.0002992829 0.0003166518 0.0003171690
## sGARCH-eGARCH 0.0001704455 0.0001771039 0.0001915731 0.0001843464 0.0001945981
## sGARCH-tGARCH 0.0002030667 0.0001958892 0.0001973595 0.0001910793 0.0001995379
## eGARCH-eGARCH 0.0002421234 0.0002430714 0.0002510836 0.0002359714 0.0002643738
## eGARCH-tGARCH 0.0002439953 0.0002659066 0.0002436313 0.0002371769 0.0002617593
## tGARCH-tGARCH 0.0001984178 0.0001973887 0.0002076179 0.0002091543 0.0002074019
##                     ged-ged
## sGARCH-sGARCH 0.0003186718
## sGARCH-eGARCH 0.0001907644
## sGARCH-tGARCH 0.0001949679
## eGARCH-eGARCH 0.0002517277
## eGARCH-tGARCH 0.0002416700
## tGARCH-tGARCH 0.0002241362
##
## $mae_mat
##                 snorm-snorm   snorm-sstd    snorm-ged    sstd-sstd     sstd-ged
## sGARCH-sGARCH 0.0002177921 0.0002511812 0.0002586679 0.0002780012 0.0002780529
## sGARCH-eGARCH 0.0001581886 0.0001546021 0.0001522445 0.0001551695 0.0001523699
## sGARCH-tGARCH 0.0001535519 0.0001533443 0.0001535570 0.0001527730 0.0001531841
## eGARCH-eGARCH 0.0001870319 0.0001894876 0.0002009011 0.0001793010 0.0002159552
## eGARCH-tGARCH 0.0001838432 0.0002184637 0.0001900387 0.0001814588 0.0002130655
## tGARCH-tGARCH 0.0001534996 0.0001534850 0.0001471818 0.0001528541 0.0001530061
##                     ged-ged
## sGARCH-sGARCH 0.0002799721
## sGARCH-eGARCH 0.0001524465
## sGARCH-tGARCH 0.0001517972
## eGARCH-eGARCH 0.0002003714
## eGARCH-tGARCH 0.0001875279
## tGARCH-tGARCH 0.0001654591
```

the appmsgarch function suggests a eGARCH-eGARCH wuth sstd-ged distribution.

```
mrs_spec <-CreateSpec(variance.spec = list(model=c("eGARCH","eGARCH")),distribution.spec = list(distribution=c("sstd","ged")),
                      switch.spec = list(do.mix = FALSE) )
summary(mrs_spec)
```

```
## Specification type: Markov-switching
## Specification name: eGARCH_sstd eGARCH_ged
## Number of parameters in each variance model: 4 4
## Number of parameters in each distribution: 2 1
## ----------------------------------------
## Fixed parameters:
## None
## ----------------------------------------
## Across regime constrained parameters:
## None
## ----------------------------------------
```

Maximum Likelihood estimation.

```
fitml <- FitML(spec = mrs_spec,data = return)
print(fitml)
```

```
## Specification type: Markov-switching
## Specification name: eGARCH_sstd eGARCH_ged
## Number of parameters in each variance model: 4 4
## Number of parameters in each distribution: 2 1
## ----------------------------------------
## Fixed parameters:
## None
## ----------------------------------------
## Across regime constrained parameters:
## None
## ----------------------------------------
## Fitted parameters:
##          Estimate Std. Error  t value  Pr(>|t|)
## alpha0_1  -0.6853     0.1143  -5.9959 1.012e-09
## alpha1_1   0.0967     0.0195   4.9479 3.751e-07
## alpha2_1  -0.2353     0.0200 -11.7659    <1e-16
## beta_1     0.9258     0.0122  76.1360    <1e-16
## nu_1       8.6408     1.4233   6.0710 6.355e-10
## xi_1       0.7938     0.0207  38.3908    <1e-16
## alpha0_2  -0.2636     0.0553  -4.7672 9.339e-07
## alpha1_2   0.1318     0.0226   5.8297 2.777e-09
## alpha2_2  -0.1275     0.0177  -7.2153 2.690e-13
## beta_2     0.9676     0.0068 141.5611    <1e-16
## nu_2       1.7214     0.0898  19.1716    <1e-16
## P_1_1      0.9978     0.0016 620.7749    <1e-16
## P_2_1      0.0035     0.0011   3.2943 4.933e-04
## ----------------------------------------
## Transition matrix:
##       t+1|k=1 t+1|k=2
## t|k=1  0.9978  0.0022
## t|k=2  0.0035  0.9965
## ----------------------------------------
## Stable probabilities:
## State 1 State 2
##  0.6115  0.3885
```

```
## ----------------------------------------
## LL: 15804.1234
## AIC: -31582.2467
## BIC: -31497.2836
## ----------------------------------------
```

the parameters indicate that the evolution of the volatility process is quite heterogeneous across the 2 regimes.

alpha2_1 = -0.2323, alpha2_2 = -0.1280 indicates different reactions to past returns.

looking at State 1 0.6613, State 2 0.3387 i understand that markov chain evolve persistently over time

persistence of volatility process: the first regime reports alpha1_1 + alpha2_1 + beta_1 = 0.7908 the second regime reports alpha1_2 + alpha2_2 + beta_2 = 0.975

the first regime has lower unconditional volatility, weaker volatility reaction to past negative returns and lower persistence of volatility process. the second instead is characterized by higher unconditional volatility, stronger volatility reaction to past negative returns and higher persistence of volatility process.

Unconditional vol

```
Unconditional_vol <- sqrt(250) * sapply(ExtractStateFit(fitml), UncVol)
print(Unconditional_vol)
```

```
## [1]        NA 0.2958819
```

Plot of estimated smoothed probability superimposed on the Nasdaq returns

```
plot(as.vector(return), axes = FALSE, ann = FALSE)
par(new = TRUE)
plot(zoo::zoo(State(fitml)$SmoothProb[,1,2,drop=TRUE], order.by = zoo::index(retur
n)), plot.type = "single")
title(main = "Smoothed probabilities")
```

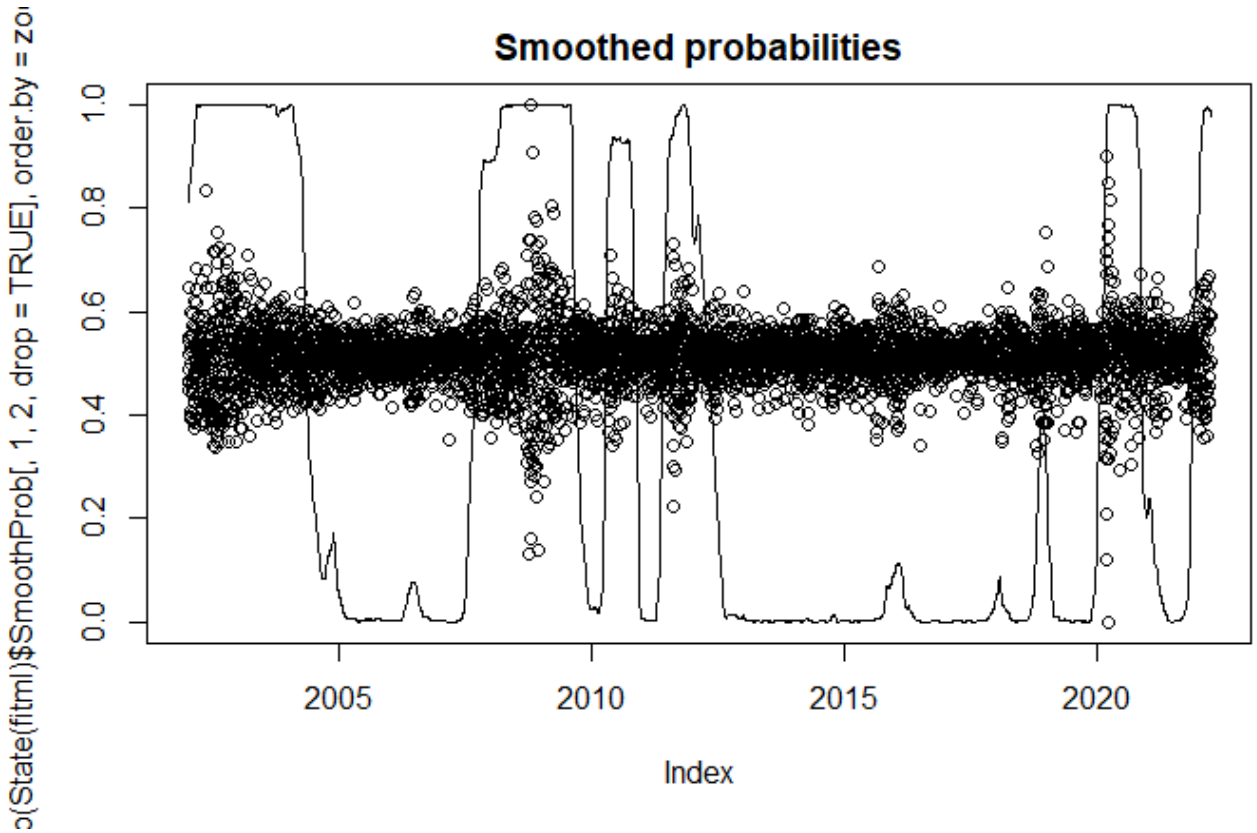


image and code from:Markov-Switching GARCH Models in R:The MSGARCH Package Journal of Statistical Software

State() function reports a list of 4 elements: FiltProb, PredProb and SmoothProb are three array of T x 1 x K dimension containing the filtered, predicted and smoothed probabilies at each time t. The last element, Viterbi is a matrix of dim T x 1 representing decoded states accordi to the Viterbi algo. [Viterbi is not a topic of this work]

when the smoothed probabilities are near one the filtered volatility of the process (graph below) increases. I further note that the MArkov chain evolves persisntently over time. [State 1 0.6613, State 2 0.3387]

#Bayesian approach ML estimation can be difficult for MSGARCH-type models. MCMC procedures can be used to explore the joint posterior distribution of the models parameters, avoiding convergence to local maxima which are possible in ML estimation.

The Bayesian approach also can obtain at low cost, by simulating from the joint posterior distribution, the exact distributions of nonlinear function of the model parameters. In addition, parameter uncertainty can be integrated in the forecasts through the predictive distribution.

The posterior distribution and Markov-switching models often exhibits non elliptical shapes which lead to non-reliable estimation of the uncertainty of model parameters.This invalidates the use of the Gaussian asymptotic distribution for inferential purposes in the finite samples.

##MCMC estimation Markov Chain Monte Carlo / Bayesian estimation

```
fit.mcmc <- FitMCMC(mrs_spec, data = return)
summary(fit.mcmc)
```

```
## Specification type: Markov-switching
## Specification name: eGARCH_sstd eGARCH_ged
## Number of parameters in each variance model: 4 4
## Number of parameters in each distribution: 2 1
## ----------------------------------------
## Fixed parameters:
## None
## ----------------------------------------
## Across regime constrained parameters:
## None
## ----------------------------------------
## Posterior sample (size: 1000)
##             Mean     SD     SE   TSSE    RNE
## alpha0_1 -0.7306 0.0863 0.0027 0.0197 0.0191
## alpha1_1  0.1081 0.0234 0.0007 0.0013 0.3442
## alpha2_1 -0.2672 0.0200 0.0006 0.0013 0.2532
## beta_1    0.9204 0.0088 0.0003 0.0020 0.0195
## nu_1     11.2814 1.2271 0.0388 0.2405 0.0260
## xi_1      0.7590 0.0285 0.0009 0.0067 0.0180
## alpha0_2 -0.1646 0.0264 0.0008 0.0048 0.0297
## alpha1_2  0.1258 0.0224 0.0007 0.0024 0.0889
## alpha2_2 -0.1020 0.0160 0.0005 0.0016 0.0970
## beta_2    0.9804 0.0034 0.0001 0.0006 0.0311
## nu_2      1.5994 0.0829 0.0026 0.0085 0.0956
## P_1_1     0.9921 0.0011 0.0000 0.0003 0.0150
## P_2_1     0.0142 0.0016 0.0001 0.0002 0.0839
## ----------------------------------------
## Posterior mean transition matrix:
##       t+1|k=1 t+1|k=2
## t|k=1  0.9921  0.0079
## t|k=2  0.0142  0.9858
## ----------------------------------------
## Posterior mean stable probabilities:
## State 1 State 2
##  0.6428  0.3572
## ----------------------------------------
## Acceptance rate MCMC sampler: 27.7%
## nmcmc: 10000
## nburn: 5000
## nthin: 10
## ----------------------------------------
## DIC: -31571.7852
## ----------------------------------------

draws <- as.matrix(fit.mcmc$par)

sel <- c("alpha1_1", "alpha2_1")
tmp <- draws[, sel]
par.mle <- fitml$par[sel]
par.bay <- apply(tmp, 2, mean)
xlim <- range(c(tmp[,1], par.mle[1]))
```

```
ylim <- range(c(tmp[,2], par.mle[2]))
par(mfrow = c(1, 1))
plot(tmp, xlim = xlim, ylim = ylim,)
par(new = TRUE)
points(par.bay[1], par.bay[2],col = 'green', lwd = 7,
       pch = 15, xlim = xlim, ylim = ylim)
points(par.mle[1], par.mle[2], col = "red", lwd = 7,
       pch = 15, xlim = xlim, ylim = ylim)
```

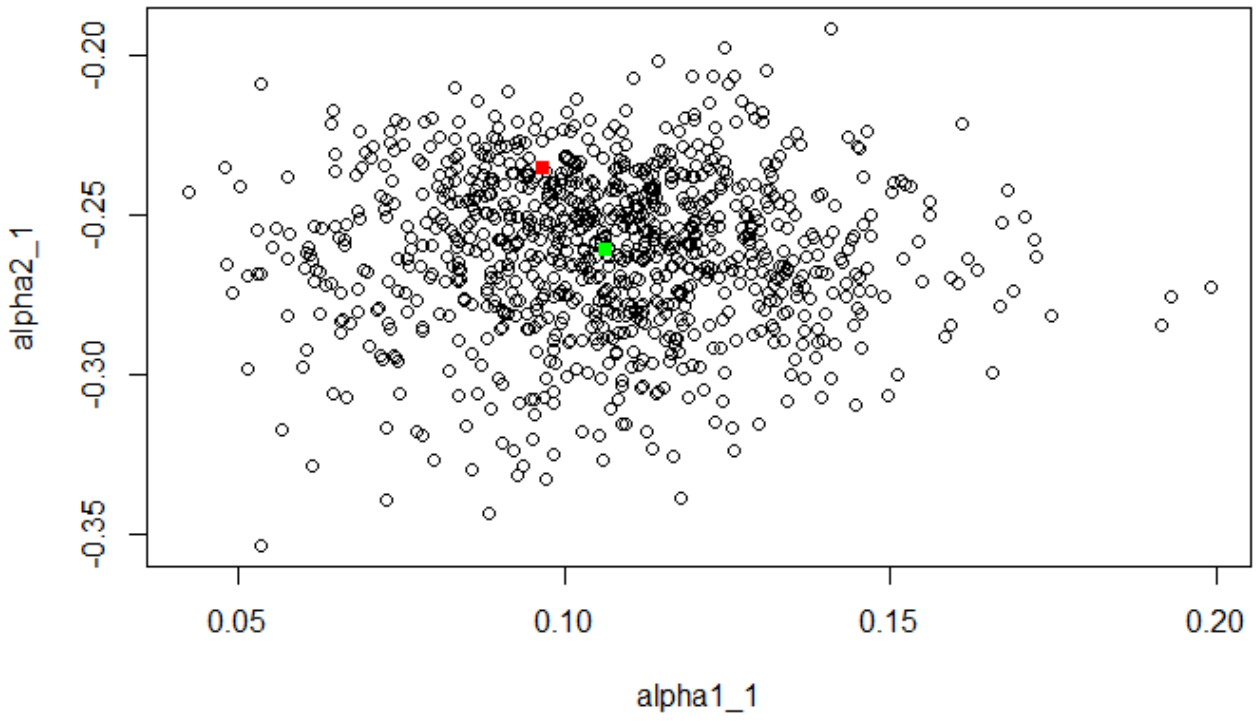


image and code from:Markov-Switching GARCH Models in R:The MSGARCH Package Journal of Statistical Software

i plot 2000 draws of the posterior sample for parameters alpha1_1, alpha1_2. The green square reports the posterior mean, the red square reports the ML estimate.

With the Bayesian estimation we can make distributional statements on any function of model parameters, achieved by simulation. [for each draw in the posterior sample we can compute the unconditional volatility in each regime to get its posterior distribution].

```
dim(tmp)

## [1] 1000    2
```

function that computes the unconditional volatility

```
f_ucvol <- function(par) {
  if (is.vector(par)) {
    par <- matrix(data = par, nrow = 1, dimnames = list(1, names(par)))
  }
```

```
  ucvol_1 <- sqrt(250) * par[,"alpha0_1"] / (1 - (par[,"alpha1_1"] + 0.5 * par[,"a
lpha2_1"] + par[,"beta_1"]))
  ucvol_2 <- sqrt(250) * par[,"alpha0_2"] / (1 - (par[,"alpha1_2"] + 0.5 * par[,"a
lpha2_2"] + par[,"beta_2"]))
  out <- list(ucvol_1 = ucvol_1, ucvol_2 = ucvol_2)
  return(out)
}
```

Compute unconditional volatility

```
ucvol.draws <- f_ucvol(draws)
ucvol.bay   <- lapply(ucvol.draws, mean)
ucvol.mle   <- f_ucvol(fitml$par)
```

posterior distributions of the unconditional annualized volatility in each regime

```
n <- length(ucvol.draws$ucvol_1)

df <- as.data.frame(ucvol.draws)
```

Histograms of posterior distribution for the unconditional volatility in each regime

```
par(mfrow = c(1, 2))
hist(df[,1], breaks = 200,xlim = range(-300, 10),ylim = range(0, 50),
     xlab = "Volatility (%)", main= '')
title(main = "Regime 1")
suppressWarnings(rug(ucvol.draws$ucvol_1)); box()
points(ucvol.bay$ucvol_1, 0, pch = 15, col = "green", lwd = 2,cex=2)
points(ucvol.mle$ucvol_1, 0, pch = 17, col = "red", lwd = 2,cex=2)

hist(df[,2], breaks = 8000,xlim = range(0, 500),ylim = range(0, 110),
     xlab = "Volatility (%)", main= '')
suppressWarnings(rug(ucvol.draws$ucvol_2)); box()
points(ucvol.bay$ucvol_2, 0, pch = 15, col = "green", lwd = 2,cex=2)
points(ucvol.mle$ucvol_2, 0, pch = 17, col = "red", lwd = 2,cex=2)
title(main = "Regime 2")
```

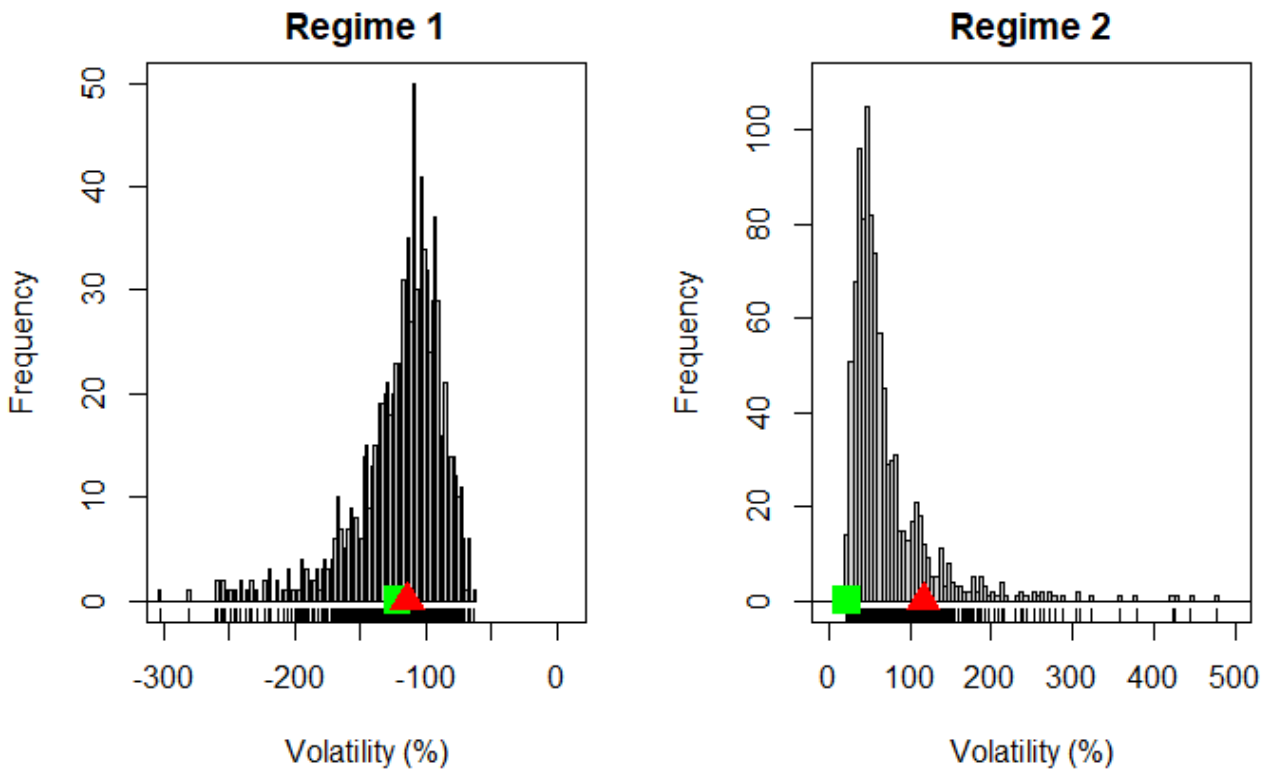


image and code from:Markov-Switching GARCH Models in R:The MSGARCH Package Journal of Statistical Software

based on 2000 draws from the joint posterior sample

Quantiles of unconditional volatility

```
sapply(ucvol.draws, quantile, probs = c(0.025, 0.975))

##          ucvol_1   ucvol_2
## 2.5%  -221.38704  24.69504
## 97.5%  -74.76481 200.36906
```

Now I want to include parameter uncertainty in the one step ahead predictive density of MSGARCH models

```
x_pred <- seq(from = -5, to = 0, length.out = 1000)
pred.mle <- as.vector(PredPdf(fitml,x= x_pred, nahead = 1))
pred.bay <- as.vector(PredPdf(fit.mcmc,x= x_pred,  nahead = 1))
pred.draws <- matrix(data = NA, nrow = nrow(draws), ncol = 1000)

for (i in 1:nrow(draws)) {
  mrs_spec <- CreateSpec(variance.spec = list(model=c("eGARCH","eGARCH")),distribu
tion.spec = list(distribution=c("sstd","ged")),
                       switch.spec = list(do.mix = FALSE) )

  tmp <- PredPdf(mrs_spec, par = draws[i,], x = x_pred, data = return, nahead = 1)
  pred.draws[i,] <- as.vector(tmp)
}
```

evaluate the one-step ahead predictive density in the range of values from −5 to 0

```
xlim <- c(-0.15, 0.0)
ylim <- c(0, 0.35)
par(mfrow = c(1, 1))
matplot(x_pred, t(pred.draws), xlim = xlim, ylim = ylim,
        type = "l",col = "lightsteelblue", xlab = "Return (%)", ylab = "Predictive
",
       lty = 1.5, las = 1, cex.axis = 1.5, cex.lab = 1.5)
title(main = "Left-tail forecast of Nasdaq return", cex.main = 1.5)
lines(x_pred, pred.bay, xlim = xlim, ylim = ylim,
      type = "l", lty = "solid", col = "green", lwd = 3)
lines(x_pred, pred.mle, xlim = xlim, ylim = ylim,
      type = "l", pch = "o", lty = "dashed", col = "red", lwd = 3)
legend("topleft", c("MCMC draws", "Bayesian","ML"),
       col = c("lightsteelblue", "green", "red"), lwd = 3,
       lty = c(1, 1, 2), bty = "n", cex = 2)
box()
```

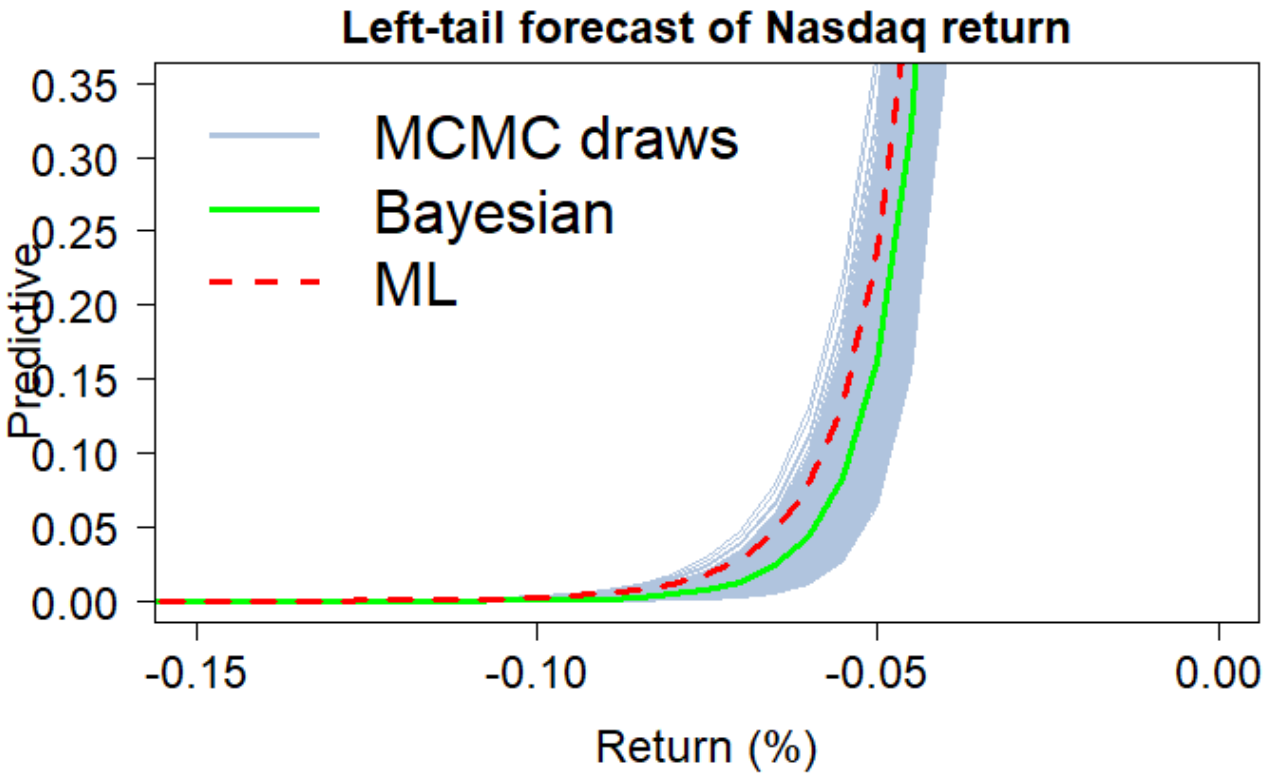


image and code from:Markov-Switching GARCH Models in R:The MSGARCH Package Journal of Statistical Software

Left tail of the one-step ahead predictive distribution for the Nasdaq returns, the green line reports the predictive Bayesian density, the red line reports the ML conditional density and the blue lines report the conditional densities obtained for each 2000 draws in posterior sample.

The Bayesian predictive density is a particular average of the predictive densities that can be formed with individual poster MCMC draws. It's more conservative that the predictive density

with plugged ML estimates and offers additional flexibility by accounting for all likely scenarios within the model structure.

VaR

#Back test analysis of the Markov Switching eGarch-eGarch model against its single-regime counterpart

First i create the single-regime model

```
mrs_spec2 <- CreateSpec(variance.spec = list(model = "eGARCH"),
                        distribution.spec = list(distribution = "sstd"),
                        switch.spec = list(K = 1))
```

and gather both specification in a list

```
models <- list(mrs_spec2, mrs_spec)

return_333 <- return[333:5093]
return_333 <- return_333[-c(1),]
return2_333 <- ts(as.vector(return_333)) *100
```

i define the backtest settings: 1000 out-of-sample observations and focus on the one-step ahead VaR forecasts at 5% risk lvl

Forecasts based on rolling windows of 1500 observations and models are re-estimated every 100 observations.

```
n.ots    <- 1000 # number of out-of-sample evaluation
n.its    <- 1500 # fit sample size
alpha    <- 0.05 # risk Level
k.update <- 100  # estimation frequency
```

2. I initialize the vector of out-of-sample returns and the matrix of forecasts

```
## Initialization
VaR   <- matrix(NA, nrow = n.ots, ncol = length(models))
y.ots <- matrix(NA, nrow = n.ots, ncol = 1)
model.fit <- vector(mode = "list", length = length(models))
```

2. I loop over the observations in the out-of-sample window

the last 1500

```
# iterate over out-of-sample time
for (i in 1: n.ots) {
  cat("Backtest - Iteration: ", i, "\n")
  y.its    <- return2_333[i:(n.its + i - 1)] # in-sample data
  y.ots[i] <- return2_333[n.its + i]         # out-of-sample data

  # iterate over models
```

```r
  for (j in 1: length(models)) {

    # update the model estimation
    if (k.update == 1 || i %% k.update == 1) {
      cat("Model", j, "is reestimated\n")
      model.fit[[j]] <- FitML(spec = models[[j]], data = y.its,
                              ctr = list(do.se = FALSE))
    }
    # VaR 1-step ahead
    VaR[i,j] <- Risk(model.fit[[j]]$spec, par = model.fit[[j]]$par,
                     data = y.its,
                     n.ahead = 1,
                     alpha   = alpha,
                     do.es   = FALSE,
                     do.its  = FALSE)$VaR
  }
}



time.index <- zoo::index(return)[(n.its + 1):(n.ots + n.its)]
y_ots <- zoo::zoo(y.ots, order.by = time.index)
VaR   <- zoo::zoo(VaR, order.by = time.index)

par(mfrow = c(1, 1))
plot(y_ots, type = 'p', las = 1, lwd = 1, xlab = "Date ",
     ylab = "", col = "black", cex.axis = 1.5, cex.lab = 1.5, pch = 19)
lines(VaR[,1], type = 'l', col = "red", lwd = 3, lty = "dashed")
lines(VaR[,2], type = 'l', col = "blue", lwd = 3)
legend("topleft", legend = c("VaR 5% - single", "VaR 5% - MS2"),
       col = c("red", "blue"), lwd = 3, cex = 1.5, lty = c("dashed", "solid"))
abline(h = 0)
title("Backtesing VaR at 5% risk level", cex.main = 1.5)
```

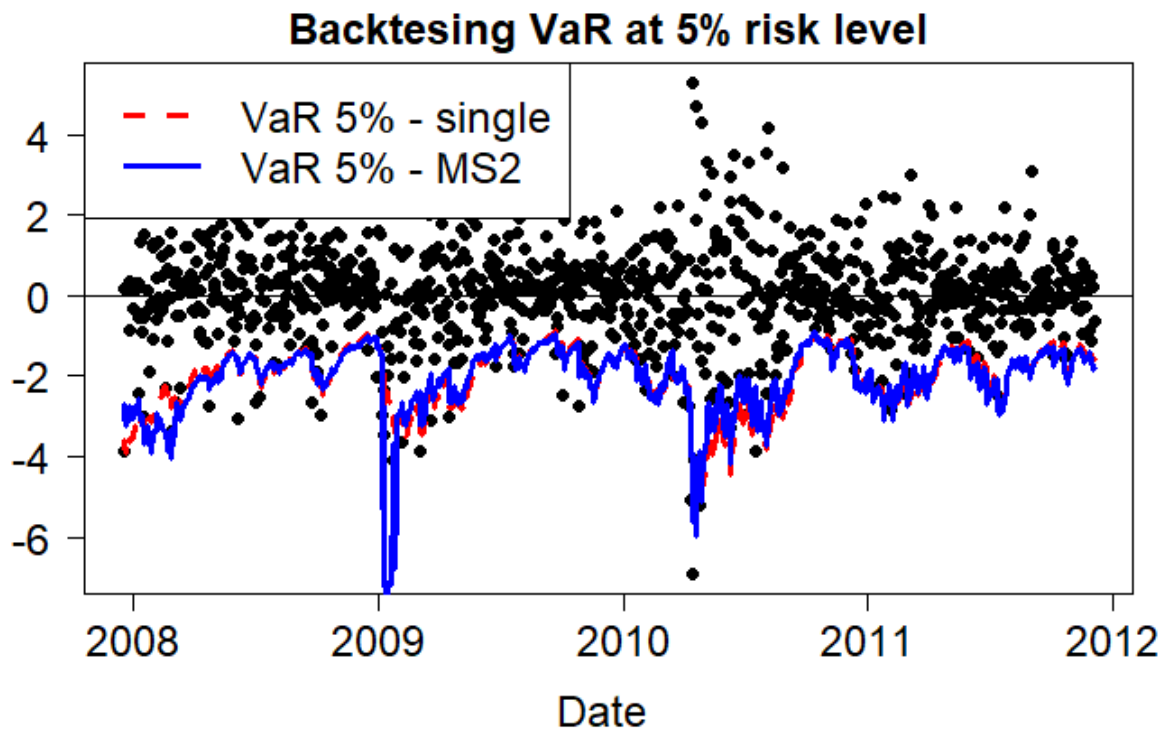


image and code from:Markov-Switching GARCH Models in R:The MSGARCH Package Journal of Statistical Software

1 day ahead VaR forecasts at 5% risk level, the Markov-switching (blue line) and single-regime (red line), with the realized returns.

I notice discrepancy during the stress periods of the stock market, I need to evaluate which VaR forecasts are most accurate in predicting the α-quantile loss (that I expect to have a proportion α of exceedances).

GAS package

compute the p values of two backtesting hypothesis tests of correct conditional coverage of the VaR (CC) and the dynamic quantile (DQ) test

they determine if the VaR forecasts achieve correct unconditional coverage and if the violations of the VaR are independent over time

```
CC.pval <- DQ.pval <- vector("double", length(models))
for (j in 1:length(models)) {
  test <- GAS::BacktestVaR(data  = y.ots,
                           VaR   = VaR[,j],
                           alpha = alpha)

  CC.pval[j] <- test$LRcc[2]
  DQ.pval[j] <- test$DQ$pvalue
}
names(CC.pval) <- names(DQ.pval) <- c("single", "MS2")

print(CC.pval)
```

```
##    single       MS2
## 0.0848907 0.0891077

print(DQ.pval)

##    single       MS2
## 0.1104897 0.1385840
```

For both tests I notice the better performance of Markov-Switching specification

Conclusion

CONCLUSION (MSGARCH)

MRS-GARCH model enjoy popularity in the financial studies with the presence of structural breaks. In this

study, we sought to identify the optimal model to capture the characteristics of Nasdaq Index over the period of

2002-01-02–2022-03-25. We used the MRS-GARCH family models to investigate the volatility of the series. In this paper we used the log-likelihood function and the Monte Carlo simulation for modelling the GARCH and MRS-GARCH family models.

Then their performance was compared with those of ARMA-GARCH family models under the normal, student-t (skewed) and GED distributions. Overall, our findings demonstrated strong evidence of switching behaviour in the Nasdaq's stock market between two states. We found the MRS-GARCH models to provide a more powerful tool in modelling the series' volatility than the traditional volatility models. The findings indicated that MRS-EGARCH model under Student-t skewed distributions has better results than other MRS-GARCH models.

We also investigated the advantages of Bayesian approach, especially while working with posterior distribution and parameters uncertainty for forecasting through the predictive distribution. Then we analysed the resulting VaR forecasts of a single regime and two-regime model together with out-of-sample (realized) returns. Surprisingly the most accurate VaR forecasts in term of predicting the α-quantile loss (such that we expect to have a proportion α of exceedances) discovered by CC and DQ tests is the single-regime.

"I would like to conclude this paper by outlining some points, which deserve further investigation. First, one could investigate the performances of MRS-GARCH models for forecasting. The estimation of the VaR depends on the reasonable assumption of innovations, and is, therefore, beyond the scope of this paper. Second, the univariate MRS-GARCH models could be extended to multivariate ones, which could model the covariance between two different asset returns in order to investigate hedging and asset allocations".

Conclusion written with the help of Mehdi Zolfaghari, Bahram Sahabi paper.

Hybrid ARIMA-GARCH in Trading strategies

For the last part of my thesis, I applied the theory of ARIMA-(MS)GARCH models and used for building a trading strategy.

Even though I tried to build my own strategy (from quantstart) I found it very difficult, so I chose to illustrate the strategy from others work.

Fist I am going to compare the ARIMA's performance with the combination of ARIMA and GARCH family models to forecast log returns (SP500).

Source: Article, *Applying Hybrid ARIMA-SGARCH in Algorithmic Investment Strategies on S&P500 Index Nguyen Vo and Robert Slepaczuk*

By using a rolling window approach, i compared ARIMA with the hybrid models (arima-sGarch, arima-eGarch)

To compare the precision of these models (in forecasting), i compared their equity lines, their forecasting error metrics (MAE, MAPE, RMSE, MAPE), and their performance metrics (annualized return compounded, annualized standard deviation, maximum drawdown, information ratio, and adjusted information ratio).

The aim of this work is to show that the hybrid models outperform ARIMA and the benchmark (Buy&Hold strategy on S&P500) over the long-term period.

The results are not sensitive to varying window sizes, the type of distribution, and the type of the GARCH model.

◊ Firstly, i conducted a rolling forecast based on the ARIMA model (window size s = 1000), while the most common approach was based on simple division of in-sample and one out-of-sample sets.
The optimized combination of p and q which has the lowest AIC is used to predict the return for the next day.

◊ Secondly, i describe and the implementation of ARIMA(p,1,q)-SGARCH(1,1) models with generalized error distribution (GED) and window size =1000, in which optimized ARIMA(p,1,q) is taken from the first step

◊ Thirdly, i evaluate the performance of sGarch with different window sizes as well as various distributions to check the sensitivity of the results.
eGARCH was also applied in the sensitivity analysis in order to check the robustness of the initial assumptions. (checking various types of GARCH models and various assumptions concerning the error distributions).

• forecasts' precision is verified in the 2-step procedure which combines the evaluation of econometric model forecasts with standard error metrics and the evaluation of investment signals constructed on these forecasts with the help of performance metrics calculated based on final equity lines.

Methodology and Input Parameter:

• rolling forecast based on an ARIMA model with window size s= 1000 (optimized combination of p, q with the lowest AIC is used to predict return for the next step)

• implementation of dynamic ARIMA(p,1,q)-sGARCH(1,1) models with GED distribution and window size = 1000 and where optimized ARIMA(p,1,q) is given by the first step

• evaluate the results based on error metrics, performance metrics, and equity curve

•• then hybrid models with different input parameters are built, different window size (500, 1500) and different (skewed) distributions: SNORM, SSTD, SGED

••• replace sGARCH with eGARCH, conduct forecasting ARIMA on different window sizes
Does the hybrid model outperform ARIMA in different input variables?

Sets of input parameters (ARIMA/hybrid ARIMA-xGARCH).

Parameters	Values
Sample sizes	s ∈ {500, **1000**, 1500} (days)
Distribution	**Generalized Error Distribution (GED)** Skewed Normal Distribution (SNORM) Skewed Generalized Error Distribution (SGED) Skewed Student t Distribution (SSTD)
xGARCH MODEL	x ∈ {**SGARCH**, eGARCH} (x represents the type of tested GARCH model. In other words, x is either **symmetric** (s)GARCH or exponential (e)GARCH.

Note: The letters in bold represent the parameters in the main test.

Image from: Applying Hybrid ARIMA-SGARCH in Algorithmic Investment Strategies on S&P500 Index Nguyen Vo and Robert Slepaczuk

implementation of dynamic **ARIMA(p,1,q)**-sGARCH(1,1):

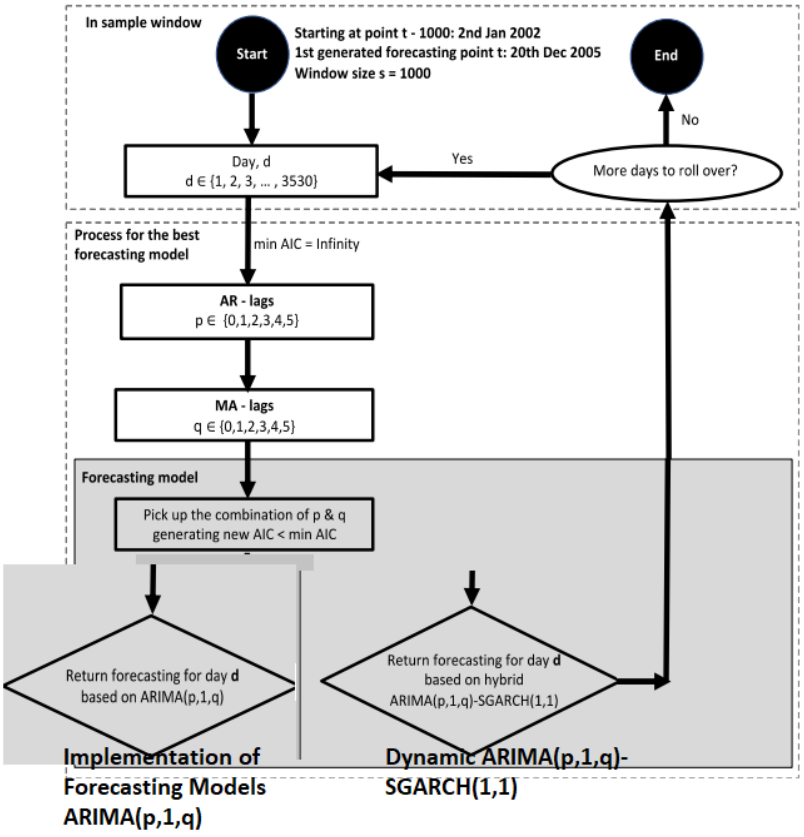


Image from: Applying Hybrid ARIMA-SGARCH in Algorithmic Investment Strategies on S&P500 Index Nguyen Vo and Robert Slepaczuk

Results:

The performance of ARIMA, hybrid model ARIMA(p,1,q)-SGARCH(1,1) with GED distribution and Buy&Hold strategy (window size = 1000)

Forecasting performance of ARIMA(p,1,q) and ARIMA(p,1,q)-SGARCH(1,1).

	Error Metrics				Performance Statistics				
METHOD	MAE	MSE	RMSE	MAPE	ARC	ASD	MD	IR	IR*
BUY&HOLD S&P500					6.931%	**18.826%**	56.775%	0.368	0.045
ARIMA 1000	12.122	310.372	17.617	0.00775	8.084%	18.878%	50.007%	0.428	0.069
SGARCH.GED 1000	**11.831**	**303.044**	**17.408**	**0.00754**	**14.026%**	18.893%	**25.885%**	**0.742**	**0.402**

Note: In order to simplify the structure of the table, SGARCH.GED 1000 is understood as ARIMA(p,1,q)-SGARCH(1,1) with GED distribution and window size equal to 1000 days. MAE: mean absolute error; MSE: mean squared error; RMSE: root mean squared error; MAPE: mean absolute percentage error; ARC: annualized return compounded; ASD: annualized standard deviation; MD: maximum drawdown; IR = ARC/ASD: information ratio; IR* = (ARC^2 * sign(ARC))/(ASD * MD): adjusted information ratio. Figures in bold indicate the best results.

Image from: Applying Hybrid ARIMA-SGARCH in Algorithmic Investment Strategies on S&P500 Index Nguyen Vo and Robert Slepaczuk

SGARCH.GED is more accurate than ARIMA in predicting returns and has the lowest values of MAE, MSE, RMSE, and MAPE.

Looking at performance statistics, the hybrid model generates the highest IR among the 3 methods.
The difference between IR and IR* is that we additionally take into account MD as a measure of risk beside ASD.

cumulative returns of the strategies are plotted to visualize the performance of ARIMA(p,1,q), hybrid model with GED distribution and benchmark.

Image from: Applying Hybrid ARIMA-SGARCH in Algorithmic Investment Strategies on S&P500 Index Nguyen Vo and Robert Slepaczuk

Before the financial crisis the buy and hold strategy remained above the ARIMA-GARCH, then in a short period time of 2 years ARIMA outperforms ARIMA-GARCH, then the hybrid model did proof to be the best model, it captures well all the movements of time series and is much better when compared with the benchmark

Robustness Test:

verify whether the results we obtained above are robust to varying family of GARCH, various distributions, and different window lengths.

◊ first robustness test → substitute sGARCH with eGARCH (keeping GED distribution and same window size).

◊ second → change GED to SNROM, SSTD, and SGED (other conditions remain unchanged).

◊ last → replace window size of 1000 to 500 and 1500 (remaining conditions are kept the same).

Forecasting performance of ARIMA(p,1,q) & ARIMA(p,1,q)-EGARCH(1,1).

	Error Metrics				Performance Statistics				
METHOD	**MAE**	**MSE**	**RMSE**	**MAPE**	**ARC**	**ASD**	**MD**	**IR**	**IR***
BUY & HOLD S&P500					6.931%	**18.826%**	56.775%	0.368	0.045
ARIMA 1000	12.122	310.372	17.617	0.00775	8.084%	18.878%	50.007%	0.428	0.069
SGARCH.GED 1000	11.831	303.044	17.408	0.00754	**14.026%**	18.893%	**25.885%**	**0.742**	**0.402**
EGARCH.GED 1000	**11.828**	**301.745**	**17.371**	**0.00753**	11.010%	18.901%	29.150%	0.582	0.220

Note: In order to simplify the structure of the table, EGARCH.GED 1000 is understood as ARIMA(p,1,q)-EGARCH(1,1) with GED distribution and window size equal to 1000 days. MAE: mean absolute error; MSE: mean squared error; RMSE: root mean squared error; MAPE: mean absolute percentage error; ARC: annualized return compounded; ASD: annualized standard deviation; MD: maximum drawdown; IR = ARC/ASD: information ratio; IR* = (ARC^2 * sign(ARC))/(ASD * MD): adjusted information ratio. The figures in bold indicate the best results.

Image from: Applying Hybrid ARIMA-SGARCH in Algorithmic Investment Strategies on S&P500 Index Nguyen Vo and Robert Slepaczuk

The combination of ARIMA(p,1,q) and EGARCH(1,1) outperforms ARIMA in a similar way as the combination of ARIMA(p,1,q) and SGARCH(1,1), even though error metrics of EGARCH.GED.1000 have the lowest values, it cannot beat the SGARCH.GED 1000 in terms of performance statistics.

EGARCH is introduced as more advanced than SGARCH since it takes the magnitude of volatility into consideration. However, the result based on IR*, which is selected as the most important performance statistic to evaluate the model, does not support this theory. *The best model is not necessarily the same when the selection is based on the best error metrics or the best performance statistics*.

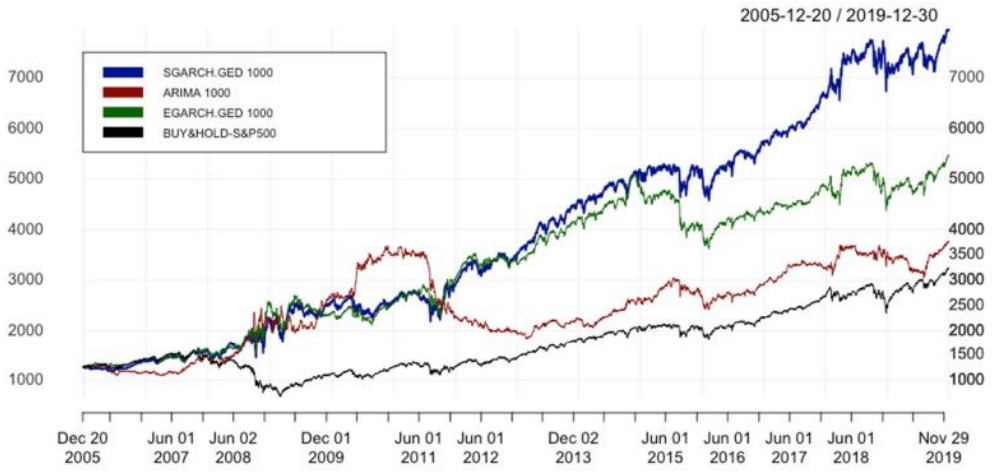


Figure 8. Equity curves of ARIMA(p,1,q) & ARIMA(p,1,q)-EGARCH(1,1). In order to simplify the structure of the legend, SGARCH.GED 1000 is understood as ARIMA(p,1,q)-SGARCH(1,1) with GED distribution and window size equal to 1000 days; EGARCH.GED 1000 is understood as ARIMA(p,1,q)-EGARCH(1,1) with GED distribution and window size equal to 1000; ARIMA 1000 is ARIMA(p,1,q) with window size s = 1000; BUY&HOLD-S&P500 is the benchmark strategy.

Image from: Applying Hybrid ARIMA-SGARCH in Algorithmic Investment Strategies on S&P500 Index Nguyen Vo and Robert Slepaczuk

Varying Window Sizes:

Performance of ARIMA(p,1,q) and hybrid models in different window sizes.

	Error Metrics				Performance Statistics				
METHOD	**MAE**	**MSE**	**RMSE**	**MAPE**	**ARC**	**ASD**	**MD**	**IR**	**IR***
BUY & HOLD S&P500					6.931%	**18.826**%	56.775%	0.368	0.045
ARIMA 500	12.216	318.342	17.842	0.00777	−0.573%	18.830%	46.471%	−0.030	0.000
SGARCH.GED 500	11.91	307.812	17.545	0.00758	5.912%	18.871%	35.666%	0.313	0.052
ARIMA 1000	12.122	310.372	17.617	0.00775	8.084%	18.878%	50.007%	0.428	0.069
SGARCH.GED 1000	11.831	**303.044**	**17.408**	**0.00753**	**14.026%**	18.893%	**25.885%**	**0.742**	**0.402**
ARIMA 1500	12.069	308.983	17.578	0.00771	5.005%	18.852%	50.733%	0.265	0.026
SGARCH.GED 1500	**11.825**	303.298	17.415	0.00753	12.186%	18.896%	25.885%	0.645	0.304

Note: In order to simplify the structure of the table, SGARCH.GED 500 is understood as ARIMA(p,1,q)-SGARCH(1,1) with GED distribution and window size s = 500 days, similar for s = 1000 and = 1500 days. MAE: mean absolute error; MSE: mean squared error; RMSE: root mean squared error; MAPE: mean absolute percentage error; ARC: annualized return compounded; ASD: annualized standard deviation; MD: maximum drawdown; IR = ARC/ASD: information ratio; IR* = (ARC^2 * sign(ARC))/(ASD * MD): adjusted information ratio. The figures in bold indicate the best results.

Image from: Applying Hybrid ARIMA-SGARCH in Algorithmic Investment Strategies on S&P500 Index Nguyen Vo and Robert Slepaczuk

With the window size switched from 500 to 1500, the hybrid models are superior to ARIMA (as I want to demonstrate),

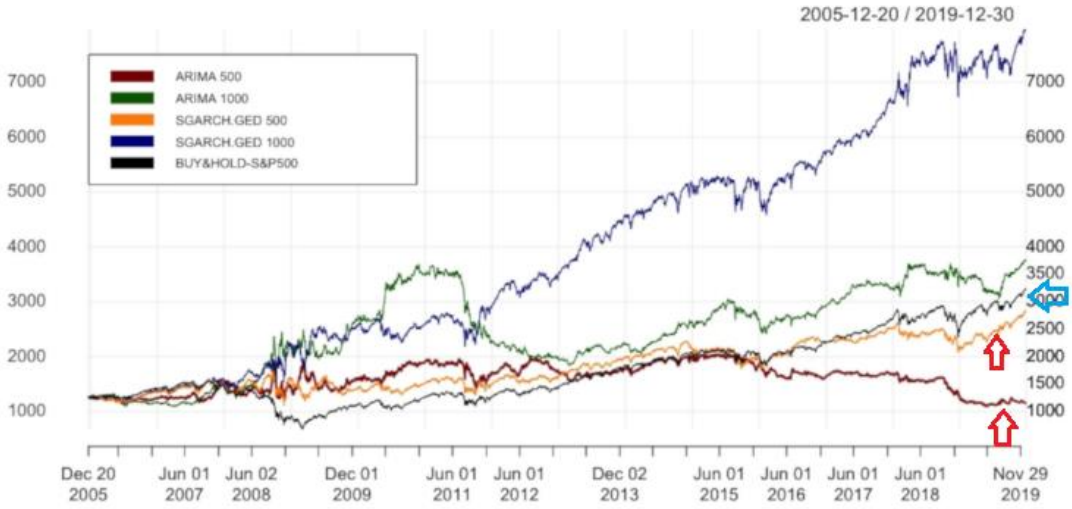


Figure 9. Equity curves of ARIMA(p,1,q) and hybrid models with window sizes = 500 and 1000. In order to simplify the structure of the legend, SGARCH.GED 500/SGARCH.GED 1000 is understood as ARIMA(p,1,q)-SGARCH(1,1) with GED distribution and window size(s) equal to 500/1000 days.

Image from: Applying Hybrid ARIMA-SGARCH in Algorithmic Investment Strategies on S&P500 Index Nguyen Vo and Robert Slepaczuk

ARIMA 500 and SGARCH.GED 500 underperform benchmark at the end. The Buy&Hold strategy is not the worst. In general, hybrid models seem to be sensitive to the values of window size. However, i can still conclude that *hybrid models outperform ARIMA regardless of the values of window size as an input parameter*.

Varying Distributions:

Performance of ARIMA(p,1,q) and hybrid models in different distributions.

	Error Metrics				Performance Statistics				
METHOD	**MAE**	**MSE**	**RMSE**	**MAPE**	**ARC**	**ASD**	**MD**	**IR**	**IR***
BUY & HOLD S&P500					6.931%	**18.826%**	56.775%	0.368	0.045
ARIMA 1000	12.122	310.372	17.617	0.00775	8.084%	18.879%	50.007%	0.428	0.069
SGARCH.GED 1000	**11.831**	**303.044**	**17.408**	**0.00754**	**14.026%**	18.893%	**25.885%**	**0.742**	**0.402**
SGARCH.SNORM 1000	11.880	303.151	17.411	0.00758	8.987%	18.890%	33.079%	0.476	0.129
SGARCH.SSTD 1000	11.928	305.642	17.483	0.00762	8.860%	18.881%	28.373%	0.469	0.147
SGARCH.SGED 1000	11.848	302.362	17.389	0.00755	9.201%	18.859%	37.566%	0.488	0.119

Note: In order to simplify the structure of the table, SGARCH.GED 1000 is understood as ARIMA(p,1,q)-SGARCH(1,1) with GED distribution and window size s = 1000 days, similar for SNORM, SGED, and SSTD. MAE: mean absolute error; MSE: mean squared error; RMSE: root mean squared error; MAPE: mean absolute percentage error; ARC: annualized return compounded; ASD: annualized standard deviation; MD: maximum drawdown; IR = ARC/ASD: information ratio; IR* = (ARC^2 * sign(ARC))/(ASD * MD): adjusted information ratio. Figures in bold indicate the best results.

Image from: Applying Hybrid ARIMA-SGARCH in Algorithmic Investment Strategies on S&P500 Index Nguyen Vo and Robert Slepaczuk

ARIMA has the worst performance with the highest values of MAE, MSE, RMSE, and MAPE if compared to hybrid models.

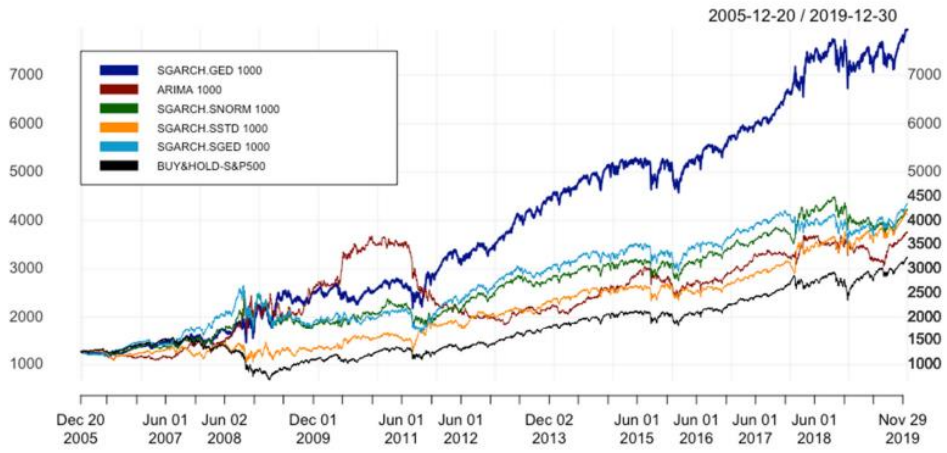


Figure 12. Equity curves of all hybrid models with different distributions. In order to simplify the structure of the legend, SGARCH.GED 1000 is understood as ARIMA(p,1,q)-SGARCH(1,1) with GED distribution and window size(s) equal to 1000 days, similar for SNORM, SSTD, and SGED.

Image from: Applying Hybrid ARIMA-SGARCH in Algorithmic Investment Strategies on S&P500 Index Nguyen Vo and Robert Slepaczuk

Cumulative returns of hybrid models with SNORM, SSTD, and SGED distributions show no significant differences at the end, but all of them surpass ARIMA's.

Conclusion:

The ARIMA(p,1,q)-SGARCH(1,1) (hybrid model) with window size 1000 can generate a trading strategy that outperforms ARIMA(p,1,q) and Buy&Hold.

The *main test* are robust to varying family of GARCH models, varying window sizes, and varying distributions.

The performance of hybrid models in the *main test* change with varying family of GARCH model, varying window sizes and varying distributions.

Conclusion from: Applying Hybrid ARIMA-SGARCH in Algorithmic Investment Strategies on S&P500 Index Nguyen Vo and Robert Slepaczuk

Trading using Garch Volatility Forecast (regime -switching)

Regime Switching System Using Volatility Forecast

This topic has been taken from *systematicinvestor in R bloggers*, both code and comments

Build an algorithm to switch between mean-reversion and trend-following strategies based on the market volatility.

◊ Two models are examined: one using the historical volatility and another using the Garch(1,1) Volatility Forecast

• The mean-reversion strategy is modelled with RSI(2) [long with RSI(2), short otherwise]

• The trend-following strategy is modeled with SMA 50/200 crossover (long if SMA 50 > 200, short if reverse)

Systematic Investor Toolbox (SIT) is used

Compares performance of the Buy and Hold, Mean-Reversion, and Trend-Following strategies using the back testing library in the Systematic Investor Toolbox

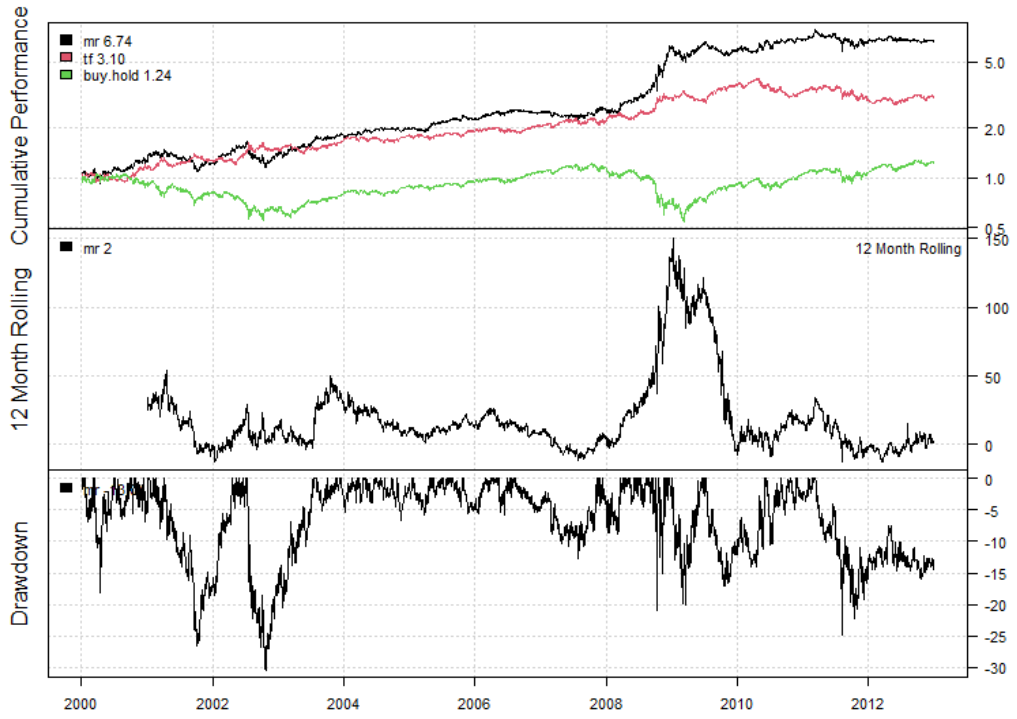


Image from: Trading using Garch Volatility Forecast by systematicinvestor in R bloggers

Now create a strategy that switches between mean-reversion and trend-following strategies based on historical market volatility:

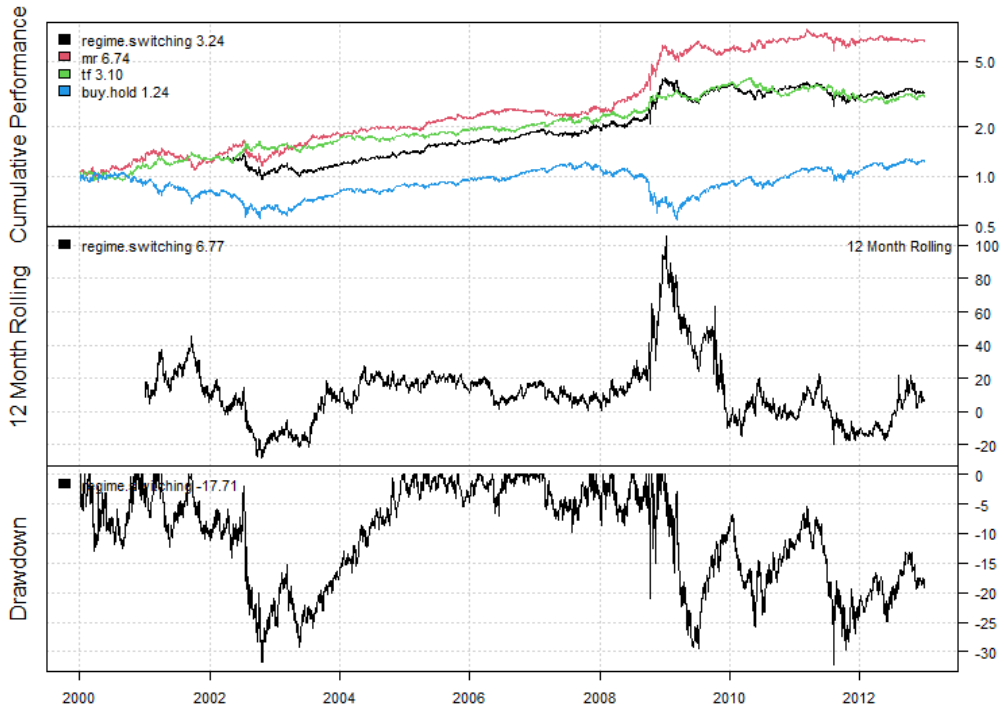


Image from: Trading using Garch Volatility Forecast by systematicinvestor in R bloggers

Next, create a GARCH(1,1) Volatility Forecast (references in the web page).

Used: `packages('tseries,fGarch')`

`garch` function from `tseries` package and `garchFit` function from `fGarch` package.

Garch function from tseries package is faster but doesn't always find solution, while garchFit function from fGarch package is slower but does converge more consistently.

Then create a strategy that switches between mean-reversion and trend-following strategies based on GARCH(1,1) volatility forecast:

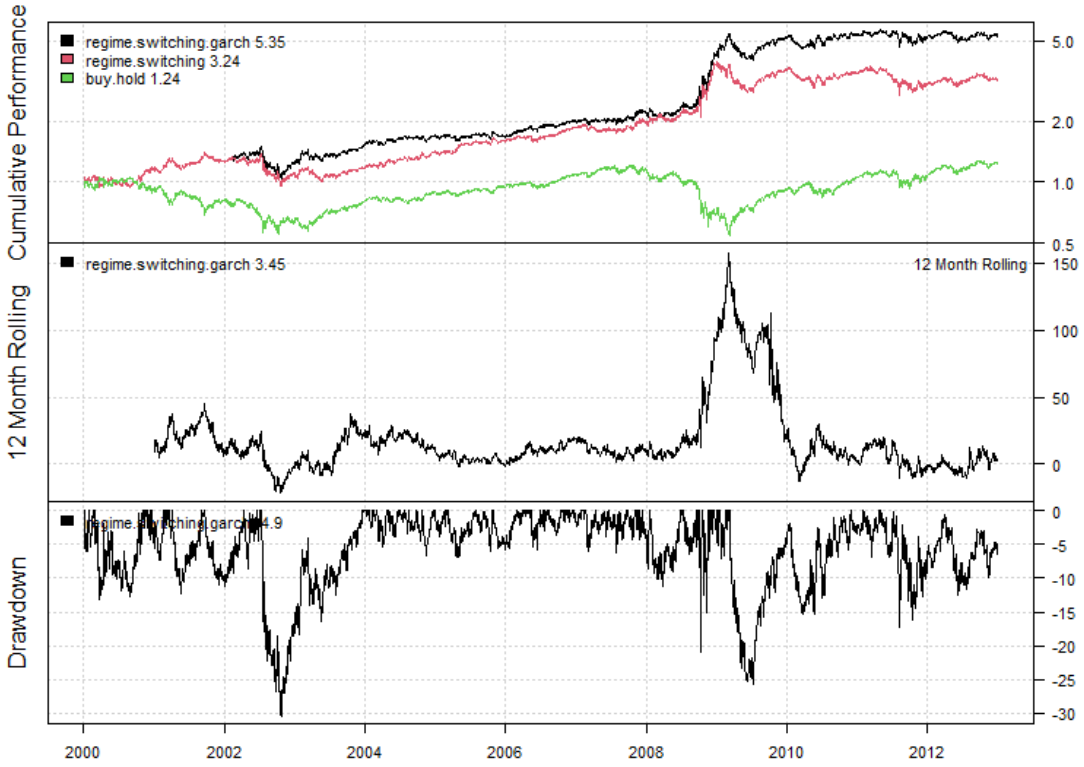


Image from: Trading using Garch Volatility Forecast by systematicinvestor in R bloggers

The switching strategy that uses GARCH(1,1) volatility forecast performs better than the one which uses historical volatility.

Full code: here

Plagiarism checks here.

References

Akaike H (1974). "A New Look at the Statistical Model Identification." IEEE Transactions on Automatic Control, 19(6), 716–723. doi:10.1007/978-1-4612-1694-0_16.

Ardia D (2008). Financial Risk Management with Bayesian Estimation of GARCH Models: Theory and Applications, volume 612 of Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, Berlin Heidelberg. doi:10.1007/978-3-540-78657-3.

Ardia D, Bluteau K, Boudt K, Catania L (2018). "Forecasting Risk with Markov-Switching GARCH Models: A Large-Scale Performance Study." International Journal of Forecasting, 34(4), 733–747. doi:10.1016/j.ijforecast.2018.05.004.

Ardia D, Bluteau K, Boudt K, Catania L, Ghalanos A, Peterson B, Trottier DA (2019a). MSGARCH: Markov-Switching GARCH Models in R. R package version 2.31, URL https://CRAN.R-project.org/package=MSGARCH.

Ardia D, Boudt K, Catania L (2019b). "Generalized Autoregressive Score Models in R: The GAS Package." Journal of Statistical Software, 88(6), 1–28. doi:10.18637/jss.v088.i06.

Ardia D, Hoogerheide LF (2010). "Bayesian Estimation of the GARCH(1, 1) Model with Student-t Innovations." The R Journal, 2(2), 41–47. doi:10.32614/rj-2010-014.

Ardia D, Hoogerheide LF, Van Dijk HK (2009). "Adaptive Mixture of Student-t Distributions as a Flexible Candidate Distribution for Efficient Simulation: The R Package AdMit." Journal of Statistical Software, 29(3), 1–32. doi:10.18637/jss.v029.i03.

Ardia D, Kolly J, Trottier DA (2017). "The Impact of Parameter and Model Uncertainty on Market Risk Predictions from GARCH-Type Models." Journal of Forecasting, 36(7), 808–823. doi:10.1002/for.2472.

Bauwens L, Backer B, Dufays A (2014). "A Bayesian Method of Change-Point Estimation with Recurrent Regimes: Application to GARCH Models." Journal of Empirical Finance, 29, 207–229. doi:10.1016/j.jempfin.2014.06.008.

Black F (1976). "Studies of Stock Price Volatility Changes." In Proceedings of the 1976 Meetings of the Business and Economics Statistics Section, pp. 177–181. American Statistical Association.

Bollerslev T (1986). "Generalized Autoregressive Conditional Heteroskedasticity." Journal of Econometrics, 31(3), 307–327. doi:10.1016/0304-4076(86)90063-1.

Brooks C, Burke SP, Persand G (2001). "Benchmarks and the Accuracy of GARCH Model Estimation." International Journal of Forecasting, 17(1), 45–56. doi:10.1016/

s0169-2070(00)00070-4.

Cai J (1994). "A Markov Model of Switching-Regime ARCH." Journal of Business & Economic Statistics, 12(3), 309–316. doi:10.2307/1392087.

Christie AA (1982). "The Stochastic Behavior of Common Stock Variances: Value, Leverage and Interest Rate Effects." Journal of Financial Economics, 10(4), 407–432. doi:10.1016/0304-405x(82)90018-6.

Christoffersen PF (1998). "Evaluating Interval Forecasts." International Economic Review, 39(4), 841–862. doi:10.2307/2527341.

Dueker MJ (1997). "Markov Switching in GARCH Processes and Mean-Reverting Stock-Market Volatility." Journal of Business & Economic Statistics, 15(1), 26–34. doi:10.

2307/1392070.

Eddelbuettel D (2019). CRAN Task View: Empirical Finance. Version 2019-03-07, URL https://CRAN.R-project.org/view=Finance.

Eddelbuettel D, François R (2011). "Rcpp: Seamless R and C++ Integration." Journal of Statistical Software, 40(8), 1–18. doi:10.18637/jss.v040.i08.

Eddelbuettel D, François R, Allaire JJ, Ushey K, Kou Q, Bates D, Chambers J (2018). Rcpp: Seamless R and C++ Integration. R package version 0.12.16, URL https://CRAN.R-project.org/package=Rcpp.

Eddelbuettel D, François R, Bates D (2017). RcppArmadillo: Rcpp Integration for the Armadillo Templated Linear Algebra Library. R package version 0.8.500.0, URL https://CRAN.R-project.org/package=RcppArmadillo.

Eddelbuettel D, Sanderson C (2014). "RcppArmadillo: Accelerating R with High-Performance C++ Linear Algebra." Computational Statistics & Data Analysis, 71, 1054–1063. doi:10.1016/j.csda.2013.02.005.

Engle RF (1982). "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation." Econometrica, 50(4), 987–1007. doi:10.2307/1912773.

Engle RF, Manganelli S (2004). "CAViaR: Conditional Autoregressive Value at Risk by Regression Quantiles." Journal of Business & Economic Statistics, 22(4), 367–381. doi:

10.1198/073500104000000370.

Fernández C, Steel MFJ (1998). "On Bayesian Modeling of Fat Tails and Skewness." Journal of the American Statistical Association, 93(441), 359–371. doi:10.1080/01621459.1998.10474117.

Francq C, Zakoian JM (2011). GARCH Models: Structure, Statistical Inference and Financial Applications. John Wiley & Sons. doi:10.1002/9780470670057.

Frühwirth-Schnatter S (2006). Finite Mixture and Markov Switching Models. 1st edition. Springer-Verlag, New York. doi:10.1007/978-0-387-35768-3.

Geweke J (2007). "Interpretation and Inference in Mixture Models: Simple MCMC Works." Computational Statistics & Data Analysis, 51(7), 3529–3550. doi:10.1016/j.csda.2006.11.026.

Ghalanos A (2017). rugarch: Univariate GARCH Models. R package version 1.3-8, URL https://CRAN.R-project.org/package=rugarch.

Gilbert P, Varadhan R (2016). numDeriv: Accurate Numerical Derivatives. R package version 2016.8-1, URL https://CRAN.R-project.org/package=numDeriv.

Glosten LR, Jagannathan R, Runkle DE (1993). "On the Relation Between the Expected Value and the Volatility of the Nominal Excess Return on Stocks." Journal of Finance, 48(5), 1779–1801. doi:10.1111/j.1540-6261.1993.tb05128.x.

Gray SF (1996). "Modeling the Conditional Distribution of Interest Rates as a Regime-Switching Process." Journal of Financial Economics, 42(1), 27–62. doi:10.1016/

0304-405x(96)00875-6.

Haas M, Mittnik S, Paolella MS (2004a). "A New Approach to Markov-Switching GARCH Models." Journal of Financial Econometrics, 2(4), 493–530.

Haas M, Mittnik S, Paolella MS (2004b). "Mixed Normal Conditional Heteroskedasticity." Journal of Financial Econometrics, 2(2), 211–250.

Hamilton JD (1989). "A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle." Econometrica, 57(2), 357–384. doi:10.2307/1912559.

Hamilton JD (1994). Time Series Analysis. Princeton University Press, Princeton.

Hamilton JD, Susmel R (1994). "Autoregressive Conditional Heteroskedasticity and Changes in Regime." Journal of Econometrics, 64(1–2), 307–333. doi:10.1016/0304-4076(94)90067-1.

Hoogerheide L, Van Dijk HK (2010). "Bayesian Forecasting of Value at Risk and Expected

Shortfall Using Adaptive Importance Sampling." International Journal of Forecasting, 26(2), 231–247. doi:10.1016/j.ijforecast.2010.01.007.

Hyndman RJ (2019). CRAN Task View: Time Series Analysis. Version 2019-08-25, URL https://CRAN.R-project.org/view=TimeSeries.

Kastner G (2016). "Dealing with Stochastic Volatility in Time Series Using the R Package stochvol." Journal of Statistical Software, 69(5), 1–30. doi:10.18637/jss.v069.i05.

Kim CJ, Nelson CR (1999). "Has the U.S. Economy Become More Stable? A Bayesian Approach Based on a Markov-Switching Model of the Business Cycle." Review of Economics and Statistics, 81(4), 608–616. doi:10.1162/003465399558472.

Klaassen F (2002). "Improving GARCH Volatility Forecasts with Regime-Switching GARCH." In Advances in Markov-Switching Models, pp. 223–254. Springer-Verlag, Berlin Heidelberg. doi:10.1007/978-3-642-51182-0_10.

Lamoureux CG, Lastrapes WD (1990). "Persistence in Variance, Structural Change, and the GARCH Model." Journal of Business & Economic Statistics, 8(2), 225–234. doi:10.2307/1391985.

McNeil AJ, Frey R, Embrechts P (2015). Quantitative Risk Management: Concepts, Techniques and Tools. 2nd edition. Princeton University Press.

Mullen KM, Ardia D, Gil DL, Windover D, Cline J (2011). "DEoptim: An R Package for Global Optimization by Differential Evolution." Journal of Statistical Software, 40(6), 1–26. doi:10.18637/jss.v040.i06.

Nelson DB (1991). "Conditional Heteroskedasticity in Asset Returns: A New Approach." Econometrica, 59(2), 347–370. doi:10.2307/2938260.

Nobel Media (2003). "The Prize in Economic Sciences 2003 – Press Release." URL https://www.nobelprize.org/nobel_prizes/economic-sciences/laureates/2003/press.html.

Papastamoulis P (2016). "label.switching: An R Package for Dealing with the Label Switching Problem in MCMC Outputs." Journal of Statistical Software, 69(1), 1–24. doi:10.18637/jss.v069.c01.

Plummer M, Best N, Cowles K, Vines K (2006). "coda: Convergence Diagnosis and Output Analysis for MCMC." R News, 6(1), 7–11. URL https://www.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

Pretis F, Reade JJ, Sucarrat G (2018). Automated General-to-Specific (GETS) Regression Modeling and Indicator Saturation for Outliers and Structural Breaks. doi:10.18637/jss.v086.i03.

R Core Team (2018). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. R version 3.5.0, URL https://www.R-project.org/.

Schwarz G (1978). "Estimating the Dimension of a Model." The Annals of Statistics, 6(2), 461–464.

Sousa T, Otiniano C, Lopes S, Diethelm W (2015). GEVStableGarch: ARMA-GARCH/APARCH Models with GEV and Stable Distributions. R package version 1.1,

URL https://CRAN.R-project.org/package=GEVStableGarch.

Spiegelhalter DJ, Best NG, Carlin BP, Van der Linde A (2002). "Bayesian Measures of Model Complexity and Fit." Journal of the Royal Statistical Society B, 64(4), 583–639. doi:10.1111/1467-9868.00353.

Sucarrat G (2015). lgarch: Simulation and Estimation of Log-GARCH Models. R package version 0.6-2, URL https://CRAN.R-project.org/package=lgarch.

Teräsvirta T (2009). "An Introduction to Univariate GARCH Models." In Handbook of Financial Time Series, pp. 17–42. Springer-Verlag, Berlin Heidelberg. doi:10.1007/978-3-540-71297-8_1.

Trapletti A, Hornik K (2017). tseries: Time Series Analysis and Computational Finance. R package version 0.10-42, URL https://CRAN.R-project.org/package=tseries.

Trottier DA, Ardia D (2016). "Moments of Standardized Fernandez-Steel Skewed Distributions: Applications to the Estimation of GARCH-Type Models." Finance Research Letters, 18, 311–316. doi:10.1016/j.frl.2016.05.006.

Vihola M (2012). "Robust Adaptive Metropolis Algorithm with Coerced Acceptance Rate." Statistics and Computing, 22(5), 997–1008. doi:10.1007/s11222-011-9269-5.

Viterbi A (1967). "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm." IEEE Transactions on Information Theory, 13(2), 260–269. doi:10.1109/tit.1967.1054010.

Wuertz D, Chalabi Y, Miklovic M, Boudt K, Chausse P (2016). fGarch: Rmetrics – Autoregressive Conditional Heteroskedastic Modelling. R package version 3010.82.1, URL

https://CRAN.R-project.org/package=fGarch.

Zakoian JM (1994). “Threshold Heteroskedastic Models.” Journal of Economic Dynamics & Control, 18(5), 931–955. doi:10.1016/0165-1889(94)90039-6.

Zeileis A, Grothendieck G (2005). “zoo: S3 Infrastructure for Regular and Irregular Time Series.” Journal of Statistical Software, 14(6), 1–27. doi:10.18637/jss.v014.i06.

Zhang Y, Wang J, Wang X (2014). “Review on Probabilistic Forecasting of Wind Power Generation.” Renewable and Sustainable Energy Reviews, 32, 255–270. doi:10.1016/j.rser.2014.01.033.

reference source.