Quantitative models 2 & 3: ARMA-GARCH and Black Box Methods

Sergey Pavlov 141

14 October 2017

library(quantmod)   
library(fBasics)   
library(car)   
library(tseries)   
library(normtest)   
library(moments)   
library(forecast)   
library(lmtest)  
library(TSA)  
library(fracdiff)  
library(fGarch)  
library(MASS)  
library(nnet)  
library(randomForest)  
source('backtest.R')

# ARMA-GARCH

# 1 Stationary AR models. Get a time series for some stock using getSymbols command:

For this assignment I've chosen the following dataset: Microsoft inc. (MSFT) daily stock prices from Yahoo finance.

Initially we have to download it:

MSFT = getSymbols('MSFT', src = 'yahoo', auto.assign = FALSE, from = as.Date("2009-01-01"), to = as.Date("2017-01-01"))

## a) Compute and plot the log price xt and the log return rt. Comment on the two plots (how volatile the data are, volatility clustering, outliers etc).

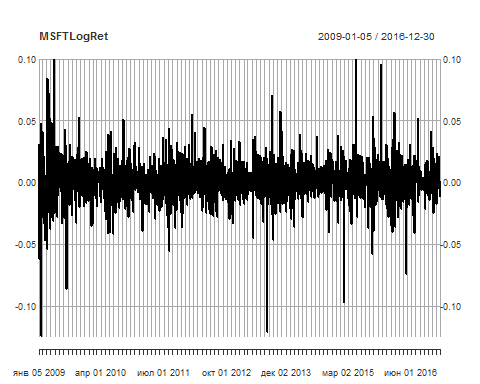
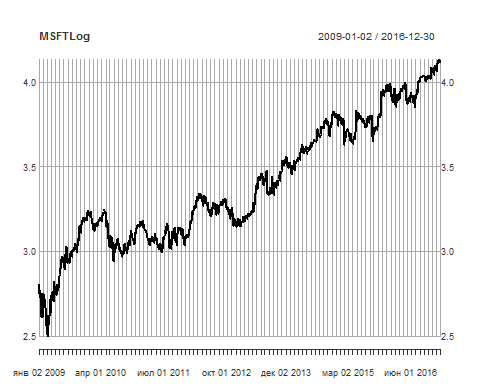
Define a function we'll also use later and count the required values:

#function we'll also need later  
getLogReturns <- function(data)  
{  
 return (na.omit(diff(log(data))))  
}  
  
MSFTLog <- na.omit(log(Ad(MSFT)))  
MSFTLogRet <-getLogReturns(Ad(MSFT))

Now let's plot them:

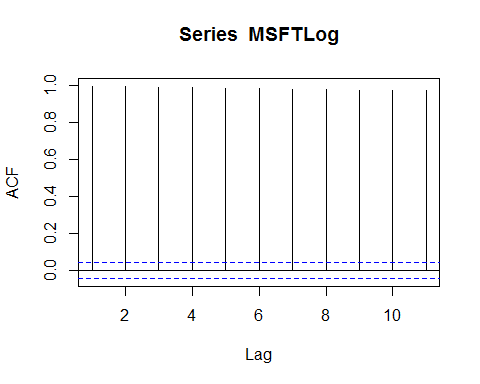
plot(MSFTLog)

plot(MSFTLogRet)



As explained in the previous work (where I've also used this dataset) Microsoft stocks are less volatile than the other companies’ stocks.  
On the Logs graph we can see the end of the 2007-2009 crysis (huge dip), followed by the recovery. Then we can see the steady constant rise.  
On the Log returns graph we can see the pretty same picture - a bigger volatility cluster at the beginning and more steady changes afterwards.

## b) Compute and plot the first 12 lags of ACF of xt. Comment on the plot. Based on the ACF, is there a unit root in xt dataset? Why?

plot(acf(MSFTLog,lag.max = 12)[1:11])

Too slow decay (visualy it stays at the 1.0 level all the time) of ACF graph tells us that the process is non-stationary and the probability of unit root existence is enormously high, therefore we can assume that there is a unit root.

## c) Consider the time series for rt. Perform the Ljung-Box test for m = 12. Draw a conclusion and justify it with the statistical language, i.e., in terms of the critical region or p-value.

Box.test(MSFTLogRet, lag = 12, type = "Ljung-Box")

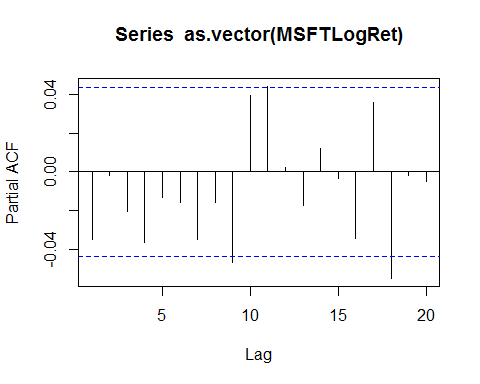
##   
## Box-Ljung test  
##   
## data: MSFTLogRet  
## X-squared = 20.796, df = 12, p-value = 0.05344

In this test we check if all of the 12 lag autocorrelations equal 0 or not. Null hypothesis says that sum of autocorr = 0.  
However, p-value is slightly more than 0.05, which means that in terms of 95% significance level we have no evidence to reject the null hypothesis saying that Microsoft log-returns aren't auto correlated to 12 lags.

## d) Use the command ar(rt,method=”mleвЂ”,order.max=20) to specify the order of an AR model for rt. Use the PACF and AIC criteria (ar() and pacf() commands). Compare both approaches.

first goes the PACF:

PACFMSFTLogRet<-pacf(as.vector(MSFTLogRet),lag.max = 20)



Here we can see that PACF tells us that the 18th element saves significancy, however we want to have a more low-level model hence we should take 11th max significance level.

Next let's evaluate the order with AIC criteria:

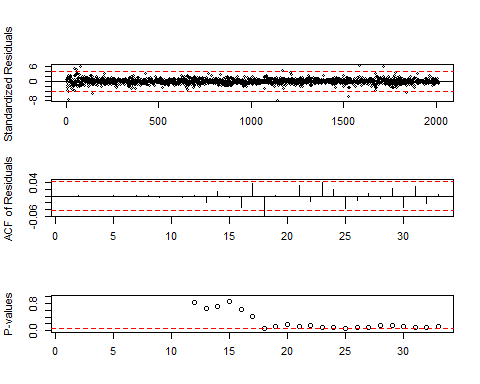
ARMSFTLogRet<-ar(x=as.vector(MSFTLogRet),method='mle',order.max=20)   
ARMSFTLogRet$aic

## 0 1 2 3 4 5 6   
## 0.4304175 0.0000000 1.9946238 3.1667118 2.4879217 4.1299224 5.6160326   
## 7 8 9 10 11 12 13   
## 5.1424768 6.6263647 4.2494453 3.1240577 1.1979650 3.1890552 4.5815367   
## 14 15 16 17 18 19 20   
## 6.1600960 8.1379075 7.7992411 7.1463742 2.8024792 4.8020725 6.7510762

As we can see the best order order chosen is the 1st lag. However we want to see a more ordered model than 1. Then we can see that the most significant lag after the first (and zero) is the 11th. Since it's the same as our pacf value, from now on we'll use the 11st lag for the AR model.

## e) Build an AR model for rt. Plot the time series of the residuals, ACF and p-values of the Ljung-Box test (command tsdiag()). Perform the Ljung-Box test of the residuals by hand adjusting the degrees of freedom for the number of the model parameters (see [2], p.66). Is the model adequate? Why? Refine the model by eliminating all estimates with t-ratio less than 1.645 and check the new model as described above. Is the new model adequate? Why? Write down the final model.

ArModelMSFTLogRet <- arima(MSFTLogRet,order=c(11,0,0))  
tsdiag(ArModelMSFTLogRet)



Box.test(ArModelMSFTLogRet$residuals, lag = 20, type = "Ljung-Box" ,fitdf =11)

##   
## Box-Ljung test  
##   
## data: ArModelMSFTLogRet$residuals  
## X-squared = 12.843, df = 9, p-value = 0.1699

Since the P-value is bigger than 0.05 , we should accept the null hypothesis of the non-correlated residuals therefore the model is adequate.

coeftest(ArModelMSFTLogRet) #lmtest

##   
## z test of coefficients:  
##   
## Estimate Std. Error z value Pr(>|z|)   
## ar1 -0.03804175 0.02226802 -1.7084 0.08757 .  
## ar2 -0.00299111 0.02225897 -0.1344 0.89310   
## ar3 -0.02177426 0.02231632 -0.9757 0.32921   
## ar4 -0.03742662 0.02232351 -1.6766 0.09363 .  
## ar5 -0.01423021 0.02234465 -0.6369 0.52422   
## ar6 -0.01635791 0.02233950 -0.7322 0.46402   
## ar7 -0.03352764 0.02233696 -1.5010 0.13336   
## ar8 -0.01563022 0.02236858 -0.6988 0.48470   
## ar9 -0.04580506 0.02236429 -2.0481 0.04055 \*  
## ar10 0.04127228 0.02239006 1.8433 0.06528 .  
## ar11 0.04470266 0.02249293 1.9874 0.04688 \*  
## intercept 0.00066032 0.00031074 2.1250 0.03359 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Analyzing the significant coefficients we can build a vector of the fixed values.

meaningfulCoefs <- c(NA, 0, 0, NA, 0, 0, 0, 0, NA, NA, NA, NA)  
ArModelRefinedMSFTLogRet<-arima(MSFTLogRet,order=c(11,0,0), fixed=meaningfulCoefs)

Box.test(ArModelRefinedMSFTLogRet$residuals, lag = 20, type = "Ljung-Box" ,fitdf = 5)

##   
## Box-Ljung test  
##   
## data: ArModelRefinedMSFTLogRet$residuals  
## X-squared = 16.949, df = 15, p-value = 0.3219

ArModelRefinedMSFTLogRet

##   
## Call:  
## arima(x = MSFTLogRet, order = c(11, 0, 0), fixed = meaningfulCoefs)  
##   
## Coefficients:  
## ar1 ar2 ar3 ar4 ar5 ar6 ar7 ar8 ar9 ar10  
## -0.0362 0 0 -0.0349 0 0 0 0 -0.0439 0.0438  
## s.e. 0.0222 0 0 0.0223 0 0 0 0 0.0223 0.0224  
## ar11 intercept  
## 0.0472 7e-04  
## s.e. 0.0225 3e-04  
##   
## sigma^2 estimated as 0.0002517: log likelihood = 5484.84, aic = -10957.69

Big p-value (>0.05) shows that we should accept the null hypothesis saying that residuals on the lag = 20 don't have any serial dependence on each other. Also we can see the final model coefficients.

## f) Does the model imply existence of a cycle? Why? If the cycles are present, compute the average length of these cycles

polynom=c(1,-ArModelRefinedMSFTLogRet$coef[1:11])  
res\_poly<-polyroot(polynom)  
res\_poly

## [1] 1.0761557+0.6925340i -0.9449752+0.8762169i -0.2602928-1.2386050i  
## [4] 1.0761557-0.6925340i 0.5130140+1.1489256i -1.5063050+0.2432700i  
## [7] -0.9449752-0.8762169i 1.3183671+0.0000000i -0.2602928+1.2386050i  
## [10] -1.5063050-0.2432700i 0.5130140-1.1489256i

mod\_res\_poly<-Mod(res\_poly)  
mod\_res\_poly

## [1] 1.279732 1.288695 1.265660 1.279732 1.258258 1.525823 1.288695  
## [8] 1.318367 1.265660 1.525823 1.258258

We can see that the cycles are present in the folowing pairs: 1-4, 2-7, 3-9, 5-11, 6-10. 8 is the non-paired value. Let's now compute all the cycle lengths and the average lenth:

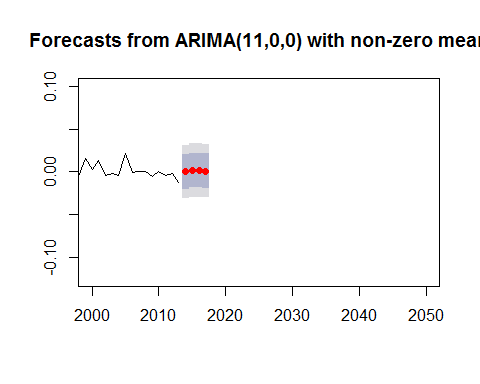
k0 <-2\*pi/acos(1.0761557/1.279732)  
k1 <-2\*pi/acos(-0.9449752/1.288695)  
k2 <-2\*pi/acos(-0.2602928/1.265660)  
k3 <-2\*pi/acos(0.5130140/1.258258)  
k4 <-2\*pi/acos(-1.5063050/1.525823)  
kMean <- (k0+k1+k2+k3+k4)/5  
kMean

## [1] 4.942779

k coeffs correspond to the number of quarters in one stochastic cycle. As we can see, the average length of a cycle is roughly 5 days.

## g) Use the fitted AR model to compute 1-step to 4-step ahead forecasts of rt at the forecast origin corresponding to the last observed date of the time series. Also, compute the corresponding 95% interval. Plot these results.

prediction\_for\_4\_steps<-forecast(model = ArModelRefinedMSFTLogRet, object = MSFTLogRet, h = 4)  
plot(prediction\_for\_4\_steps,shaded=TRUE,xlim=c(2000,2050),fcol=2)

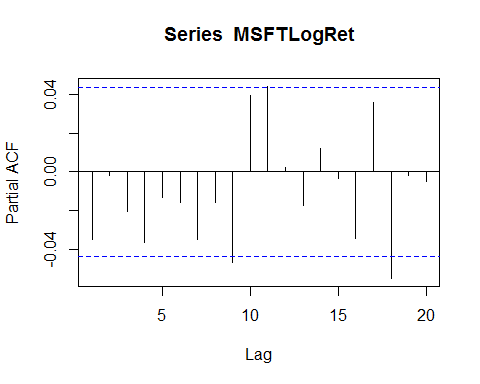


We can see the 95% interval as the gray “window” on the graph.

# 2. Consider a MA model for rt:

## a) Choose the order of such model. Support your choice with the ACF plot

plot(pacf(MSFTLogRet,lag.max = 20)[1:20])



The last not too huge lag which has significant autocorrelation is 11th, so our choice for MA is 11th order, as well as in the case of AR.

## b) Build the model. Refine it by removing coefficients estimates with t-ratio less than 1.645. Write down the fitted model.

MaMSFT<-arima(MSFTLogRet,order=c(0,0,11))  
Box.test(MaMSFT$residuals, type = c("Ljung-Box"), fitdf = 11, lag = 20)

##   
## Box-Ljung test  
##   
## data: MaMSFT$residuals  
## X-squared = 12.274, df = 9, p-value = 0.1983

P-Value on the Box-test is greater than 0.05 which means that there's no serial correlation

coeftest(MaMSFT)

##   
## z test of coefficients:  
##   
## Estimate Std. Error z value Pr(>|z|)   
## ma1 -3.7267e-02 2.2257e-02 -1.6744 0.09406 .  
## ma2 -5.1964e-05 2.2274e-02 -0.0023 0.99814   
## ma3 -2.1639e-02 2.2294e-02 -0.9706 0.33175   
## ma4 -3.8494e-02 2.2218e-02 -1.7325 0.08318 .  
## ma5 -7.6057e-03 2.2392e-02 -0.3397 0.73411   
## ma6 -1.6549e-02 2.2447e-02 -0.7373 0.46096   
## ma7 -3.1981e-02 2.2115e-02 -1.4462 0.14813   
## ma8 -3.1235e-03 2.3306e-02 -0.1340 0.89338   
## ma9 -5.2489e-02 2.3963e-02 -2.1904 0.02849 \*  
## ma10 4.5910e-02 2.2565e-02 2.0345 0.04190 \*  
## ma11 4.0235e-02 2.2847e-02 1.7611 0.07822 .  
## intercept 6.5919e-04 3.1060e-04 2.1223 0.03381 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Significant coefs are 1st, 4th, 9th, 10th, 11th and intercept Let's now create and print the refined model:

fixedMA=c(NA,0,0,NA,0,0,0,0,NA,NA,NA,NA)  
MaMSFTRefined<-arima(MSFTLogRet,order=c(0,0,11),fixed = fixedMA)  
MaMSFTRefined

##   
## Call:  
## arima(x = MSFTLogRet, order = c(0, 0, 11), fixed = fixedMA)  
##   
## Coefficients:  
## ma1 ma2 ma3 ma4 ma5 ma6 ma7 ma8 ma9 ma10  
## -0.0368 0 0 -0.0387 0 0 0 0 -0.0507 0.0454  
## s.e. 0.0223 0 0 0.0223 0 0 0 0 0.0236 0.0224  
## ma11 intercept  
## 0.0415 7e-04  
## s.e. 0.0226 3e-04  
##   
## sigma^2 estimated as 0.0002516: log likelihood = 5485.12, aic = -10958.23

## c) Compute the Ljung-Box statistic of the residuals of the fitted MA model. Is there serial correlation in the residuals? Why?

Box.test(MaMSFTRefined$residuals, type = c("Ljung-Box"), fitdf = 5, lag = 20)

##   
## Box-Ljung test  
##   
## data: MaMSFTRefined$residuals  
## X-squared = 15.834, df = 15, p-value = 0.3932

P-Value is greater than 0.05 which means that there's no serial correlation.

## d)Consider the in-sample fits of the AR model of Problem 1 and the MA model. Which model is preferred? Why?

AIC: more regressors -> bigger criteria value -> worse predictions

AIC(ArModelRefinedMSFTLogRet)

## [1] -10955.69

AIC(MaMSFTRefined)

## [1] -10956.23

We can see that absolute value of MA model is less than AR in the absolute values, therefore we can assume that it predicts better due to the AIC criteria

BIC: more data -> bigger criteria value -> worse predictions

BIC(ArModelRefinedMSFTLogRet)

## [1] -10916.44

BIC(MaMSFTRefined)

## [1] -10916.98

We can see the same picture in the case of BIC criteria. The MA model predicts better.

## e) Use backtest at some forecast origin with horizon h = 1 to compare the two models. Indicate clearly the parameters of such backtesting (the estimation and forecasting subsamples, forecast origin and so on). Which model is preferred? Why?

I’ve chosen to divide the samples on 1800 and 200 subsamples.

ArRefinedModelBackTest <- backtest(ArModelRefinedMSFTLogRet, MSFTLogRet, 1800, 1)

## [1] "RMSE of out-of-sample forecasts"  
## [1] 0.01281129  
## [1] "Mean absolute error of out-of-sample forecasts"  
## [1] 0.008811734

MaRefinedModelBackTest <- backtest(MaMSFTRefined, MSFTLogRet, 1800, 1)

## [1] "RMSE of out-of-sample forecasts"  
## [1] 0.01282625  
## [1] "Mean absolute error of out-of-sample forecasts"  
## [1] 0.008819902

[1] "RMSE of out-of-sample forecasts" [1] 0.01482452 [1] "Mean absolute error of out-of-sample forecasts" [1] 0.01005583

[1] "RMSE of out-of-sample forecasts" [1] 0.01484428 [1] "Mean absolute error of out-of-sample forecasts" [1] 0.0100773

AR model shows a less significant deviation from the sample forecasts therefore it's preffered over the MA model.

# 3. Yet again, focus on the log return series rt of the asset from Problem 1. Build an ARMA model including

## a) Choosing the order of the model

eacfCrit <- eacf(z = MSFTLogRet, ar.max = 20, ma.max = 20)

## AR/MA  
## 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20  
## 0 o o o o o o o o o x o o o o o o o x o o o   
## 1 x o o o o o o o o o o o o o o o o x o o o   
## 2 x x o o o o o o o o o o o o o o o x o o o   
## 3 x x x o o o o o o o o o o o o o o x o o o   
## 4 x x x o o o o o o o o o o o o o o x o o o   
## 5 x x x x x o o o o o o o o o o o o o o o o   
## 6 x x x x o o o o o o o o o o o o o o o o o   
## 7 x x x o x x x o o o o o o o o o o o o o o   
## 8 x x x o x x x x o o o o o o o o o o o o o   
## 9 x x x x x o o x x o o o o o o o o o o o o   
## 10 x x x x x x o o x x o o o o o o o o o o o   
## 11 o x x x x o o o x o o o o o o o o o o o o   
## 12 x x o x o x o o x x x x o o o o o o o o o   
## 13 x x o x o x o x x x x x x o o o o o o o o   
## 14 x x x o x o x x o x o o x o o o o o o o o   
## 15 x x x o o o o x x x o o x o o o o o o o o   
## 16 x x x x x o x x x x o o o o x x o o o o o   
## 17 x x x x x o x x x x o x o o x x o o o o o   
## 18 o x x x x x x x x x x o x o o x x o o o o   
## 19 o o x x x x x x o o x o x o o o x o o o o   
## 20 x o x x x x x x x o x x x x o o x o o o o

2/sqrt(2013) #the 95% critical value

## [1] 0.04457672

#print(eacfCrit$eacf, digits = 2)

we can see that there's a clear triangle pointing to the (0,0). However also we can see that there are non significant X's on the column 9 and 17. Taking into account this information and that (0,0) model is very unlikely for us, we can say that it's somewhere on the line p=q. (1-1, 2-2 and etc.)

Let's now evaluate them by building all the functions and take the best one. First we'll use auto.arima() func in order to find the closest point and then we'll check the closest points by using a custom function, choosing the best model based on the AIC criteria.

#We can use the following function which takes into account the possibility of the white noise and handles it by increasing the I coefficient   
auto.arima(MSFTLogRet, ic = 'aic') # it gives us arima(3,0,2)

## Series: MSFTLogRet   
## ARIMA(3,0,2) with non-zero mean   
##   
## Coefficients:  
## ar1 ar2 ar3 ma1 ma2 mean  
## -0.0330 0.6926 -0.0005 -0.0041 -0.7129 7e-04  
## s.e. 0.2387 0.2113 0.0291 0.2372 0.2217 3e-04  
##   
## sigma^2 estimated as 0.0002538: log likelihood=5479.46  
## AIC=-10944.92 AICc=-10944.86 BIC=-10905.66

getModelAIC <- function (data, modelType, orderP, orderQ, armaP = 0, armaQ = 0)  
{  
 if(modelType == "arima")  
 {  
 modelArima <- arima(MSFTLogRet, order = c(orderP,0,orderQ))  
 return (modelArima$aic)  
 }  
 else if (modelType == "garch")  
 {  
 formula = as.formula(paste(paste("~arma(",armaP,",",armaQ,")",sep=""),"+",paste("garch(",orderP,",",orderQ,")",sep = "")))  
 modelGarch <- garchFit(formula, data = data, trace = FALSE)  
 return ((modelGarch@fit)$ics[1])  
 }  
 else if (modelType == "aparch")  
 {  
 formula = as.formula(paste(paste("~arma(",armaP,",",armaQ,")",sep=""),"+",paste("aparch(",orderP,",",orderQ,")",sep = "")))  
 modelAparch <- garchFit(formula, data = data, trace = FALSE, delta = 2, include.delta = FALSE)  
 return ((modelAparch@fit)$ics[1])  
 }  
}  
  
autoFitArima <- function (data, max.order, modelType, armaP = 0, armaQ = 0)  
{  
 min.aic = 0  
 optimal.p = 0  
 optimal.q = 0  
   
 for (p in 0:max.order)  
 {  
 if (p==0 & modelType != "arima")  
 {  
 p = 1  
 }  
   
 for (q in 0:max.order)  
 {  
 this.aic = getModelAIC(data, modelType, p,q,armaP,armaQ)  
 if(this.aic <= min.aic)  
 {  
 min.aic = this.aic  
 optimal.p = p  
 optimal.q = q  
 }  
 }  
 }  
   
 return (list(aic = min.aic, p = optimal.p, q = optimal.q))  
}  
  
#aics:  
autoFitArima(MSFTLogRet, max.order = 4, modelType = "arima")

## $aic  
## [1] -10965.3  
##   
## $p  
## [1] 3  
##   
## $q  
## [1] 3

The auto.arima func suggests us (3,0,2) model, however I've checked all the nearest points and discovered the (3,0,3) model, which has the smallers 'aic' value (-10965.3). Therefore we'll use p=3 and q=3.

## b) Writing down the model

Let's now write down the model:

eacfArima <- arima(MSFTLogRet, order = c(3,0,3))  
eacfArima

##   
## Call:  
## arima(x = MSFTLogRet, order = c(3, 0, 3))  
##   
## Coefficients:  
## ar1 ar2 ar3 ma1 ma2 ma3 intercept  
## -0.5151 0.4925 0.9889 0.5121 -0.5120 -1.0000 6e-04  
## s.e. 0.0034 0.0049 0.0035 0.0031 0.0031 0.0032 1e-04  
##   
## sigma^2 estimated as 0.0002494: log likelihood = 5489.65, aic = -10965.3

## c) Checking the model for adequacy by analyzing the residuals

Box.test(eacfArima$residuals, type = c("Ljung-Box"), fitdf = 6, lag = 20)

##   
## Box-Ljung test  
##   
## data: eacfArima$residuals  
## X-squared = 36.978, df = 14, p-value = 0.0007434

P-Value is less than 0.05 which means that the serial correlation exists and the model isn't adequate.

## d) Backtesting and comparing the model with those of Problems 1 and 2.

eacfBackTest <- backtest(eacfArima, MSFTLogRet, 1800, 1)

## [1] "RMSE of out-of-sample forecasts"  
## [1] 0.01285353  
## [1] "Mean absolute error of out-of-sample forecasts"  
## [1] 0.008779106

let's also compare the results considering the previously made backtests for AR and MA models

eacfBackTest$rmse

## [1] 0.01285353

ArRefinedModelBackTest$rmse

## [1] 0.01281129

MaRefinedModelBackTest$rmse

## [1] 0.01282625

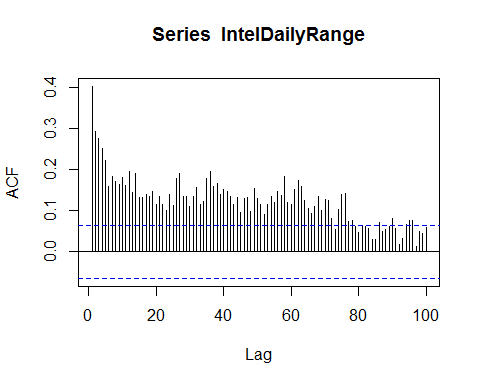
We can see that the best model due to the backtest is the refined AR model.

# 4. Consider the daily range (daily high minus daily low) of a blue chip stock (Apple, CocaCola etc.) for the last 4 years. Compute the first 100 lags of ACF of this series. Is there evidence of long-range dependence? Explain! If the range series has long memory, build an AFRIMA model for the data.

For this task I've chosen the Intel stocks:

Intel = getSymbols('INTC', src = 'yahoo', auto.assign = FALSE, from="2013-09-30", to = "2017-06-01")  
IntelDailyRange <-Hi(Intel)-Lo(Intel)

plot(acf(IntelDailyRange,lag.max = 100)[0:100])



On the graph we can see that the decay is less significant than the exponent therefore we can tell that there's a long memory.

arfimaIntelModel <- arfima(y = as.numeric(IntelDailyRange))  
  
summary(arfimaIntelModel)

##   
## Call:  
## arfima(y = as.numeric(IntelDailyRange))   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## d 2.864e-01 7.676e-07 373070 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## sigma[eps] = 0.2614616   
## [d.tol = 0.0001221, M = 100, h = 7.678e-07]  
## Log likelihood: -71.87 ==> AIC = 147.7307 [2 deg.freedom]

#fracdiff(as.numeric(IntelDailyRange), nar = 1,nma = 1)

Here we have the arfima model with the automatically estimated d-coef. AR and MA are assigned to 0.

# 5 Consider the log return series rt of the asset from Problem 1.

## a) Build an appropriate ARMA model.

the best model achieved previously is arma(3,0,3)

eacfArima

##   
## Call:  
## arima(x = MSFTLogRet, order = c(3, 0, 3))  
##   
## Coefficients:  
## ar1 ar2 ar3 ma1 ma2 ma3 intercept  
## -0.5151 0.4925 0.9889 0.5121 -0.5120 -1.0000 6e-04  
## s.e. 0.0034 0.0049 0.0035 0.0031 0.0031 0.0032 1e-04  
##   
## sigma^2 estimated as 0.0002494: log likelihood = 5489.65, aic = -10965.3

## b) Test the residuals for the ARCH effect.

Box.test((eacfArima$residuals^2), type = c("Ljung"), lag = 20)

##   
## Box-Ljung test  
##   
## data: (eacfArima$residuals^2)  
## X-squared = 79.481, df = 20, p-value = 4.808e-09

Due to the small P-value we reject the null hypothesis and accept the alternative hypothesis telling that the model has significant serial correlation (ARCH effect). This has proved our previous assumption.

## c) Fit an ARMA-GARCH Gaussian model to the data.

Due to an error in the predict function we can't use P >= 2 in the arma, otherwise we'll get an error when predicting. Therefore, we have to take AR value equal to 1. More info on this issue as well as the research is given here: <https://stackoverflow.com/questions/15475869/error-in-predict-for-arma-garch-model>

autoFitArima(data = MSFTLogRet, max.order = 2, modelType = "garch", armaP = 1, armaQ = 3)

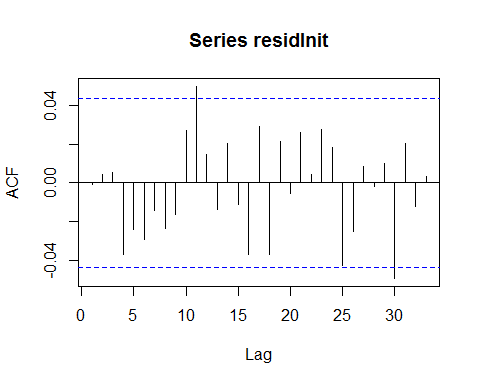
## $aic  
## AIC   
## -5.527448   
##   
## $p  
## [1] 2  
##   
## $q  
## [1] 2

# function gives us a (2,2) model  
  
#we pick the garch(2,2) model for having the lowest aic value.  
garchMSFT <- garchFit(~arma(1,3)+garch(2,2), data = MSFTLogRet, trace = FALSE)   
  
summary(garchMSFT)

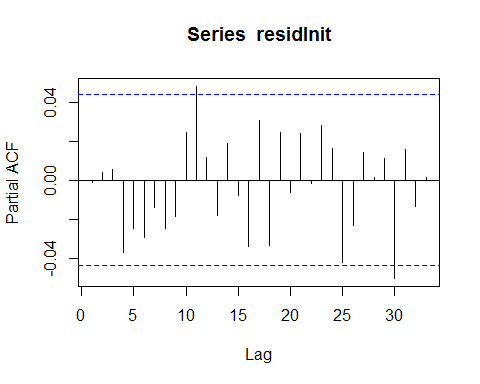
##   
## Title:  
## GARCH Modelling   
##   
## Call:  
## garchFit(formula = ~arma(1, 3) + garch(2, 2), data = MSFTLogRet,   
## trace = FALSE)   
##   
## Mean and Variance Equation:  
## data ~ arma(1, 3) + garch(2, 2)  
## <environment: 0x000000001ea34ef8>  
## [data = MSFTLogRet]  
##   
## Conditional Distribution:  
## norm   
##   
## Coefficient(s):  
## mu ar1 ma1 ma2 ma3   
## 1.3861e-03 -5.1947e-01 5.0488e-01 -2.5073e-03 -2.6050e-02   
## omega alpha1 alpha2 beta1 beta2   
## 4.8023e-05 7.4710e-02 1.3198e-01 1.0000e-08 6.1343e-01   
##   
## Std. Errors:  
## based on Hessian   
##   
## Error Analysis:  
## Estimate Std. Error t value Pr(>|t|)   
## mu 1.386e-03 5.206e-04 2.663 0.00775 \*\*   
## ar1 -5.195e-01 2.066e-01 -2.515 0.01191 \*   
## ma1 5.049e-01 2.085e-01 2.422 0.01544 \*   
## ma2 -2.507e-03 3.020e-02 -0.083 0.93384   
## ma3 -2.605e-02 2.673e-02 -0.975 0.32970   
## omega 4.802e-05 1.162e-05 4.134 3.57e-05 \*\*\*  
## alpha1 7.471e-02 2.589e-02 2.885 0.00391 \*\*   
## alpha2 1.320e-01 3.207e-02 4.116 3.85e-05 \*\*\*  
## beta1 1.000e-08 7.719e-02 0.000 1.00000   
## beta2 6.134e-01 6.924e-02 8.859 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Log Likelihood:  
## 5573.376 normalized: 2.768691   
##   
## Description:  
## Wed Oct 18 20:42:27 2017 by user: satsuma   
##   
##   
## Standardised Residuals Tests:  
## Statistic p-Value   
## Jarque-Bera Test R Chi^2 4511.137 0   
## Shapiro-Wilk Test R W 0.9358768 0   
## Ljung-Box Test R Q(10) 9.330544 0.5010473  
## Ljung-Box Test R Q(15) 16.24885 0.3657082  
## Ljung-Box Test R Q(20) 24.59555 0.2173443  
## Ljung-Box Test R^2 Q(10) 3.11053 0.9787019  
## Ljung-Box Test R^2 Q(15) 3.611567 0.9987575  
## Ljung-Box Test R^2 Q(20) 4.660579 0.9998403  
## LM Arch Test R TR^2 3.334375 0.9926875  
##   
## Information Criterion Statistics:  
## AIC BIC SIC HQIC   
## -5.527448 -5.499592 -5.527497 -5.517223

## d) Check the model by analyzing standardized residuals.

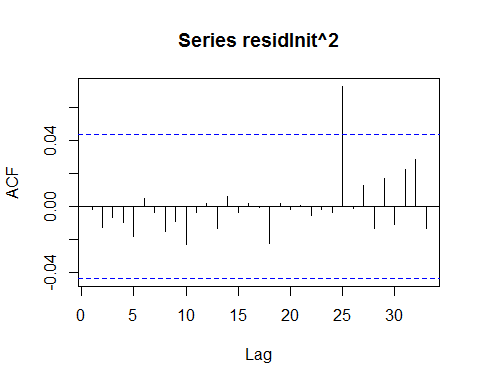
residInit <-residuals(garchMSFT, standardize = TRUE)   
  
acf(residInit)



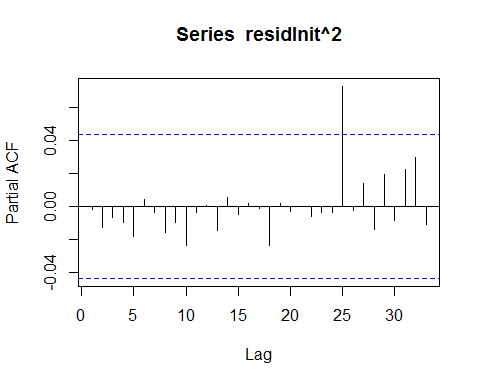
pacf(residInit)



acf(residInit^2)



pacf(residInit^2)



summary(garchMSFT)

##   
## Title:  
## GARCH Modelling   
##   
## Call:  
## garchFit(formula = ~arma(1, 3) + garch(2, 2), data = MSFTLogRet,   
## trace = FALSE)   
##   
## Mean and Variance Equation:  
## data ~ arma(1, 3) + garch(2, 2)  
## <environment: 0x000000001c3deb18>  
## [data = MSFTLogRet]  
##   
## Conditional Distribution:  
## norm   
##   
## Coefficient(s):  
## mu ar1 ma1 ma2 ma3   
## 1.3861e-03 -5.1947e-01 5.0488e-01 -2.5073e-03 -2.6050e-02   
## omega alpha1 alpha2 beta1 beta2   
## 4.8023e-05 7.4710e-02 1.3198e-01 1.0000e-08 6.1343e-01   
##   
## Std. Errors:  
## based on Hessian   
##   
## Error Analysis:  
## Estimate Std. Error t value Pr(>|t|)   
## mu 1.386e-03 5.206e-04 2.663 0.00775 \*\*   
## ar1 -5.195e-01 2.066e-01 -2.515 0.01191 \*   
## ma1 5.049e-01 2.085e-01 2.422 0.01544 \*   
## ma2 -2.507e-03 3.020e-02 -0.083 0.93384   
## ma3 -2.605e-02 2.673e-02 -0.975 0.32970   
## omega 4.802e-05 1.162e-05 4.134 3.57e-05 \*\*\*  
## alpha1 7.471e-02 2.589e-02 2.885 0.00391 \*\*   
## alpha2 1.320e-01 3.207e-02 4.116 3.85e-05 \*\*\*  
## beta1 1.000e-08 7.719e-02 0.000 1.00000   
## beta2 6.134e-01 6.924e-02 8.859 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Log Likelihood:  
## 5573.376 normalized: 2.768691   
##   
## Description:  
## Fri Oct 20 16:20:56 2017 by user: satsuma   
##   
##   
## Standardised Residuals Tests:  
## Statistic p-Value   
## Jarque-Bera Test R Chi^2 4511.137 0   
## Shapiro-Wilk Test R W 0.9358768 0   
## Ljung-Box Test R Q(10) 9.330544 0.5010473  
## Ljung-Box Test R Q(15) 16.24885 0.3657082  
## Ljung-Box Test R Q(20) 24.59555 0.2173443  
## Ljung-Box Test R^2 Q(10) 3.11053 0.9787019  
## Ljung-Box Test R^2 Q(15) 3.611567 0.9987575  
## Ljung-Box Test R^2 Q(20) 4.660579 0.9998403  
## LM Arch Test R TR^2 3.334375 0.9926875  
##   
## Information Criterion Statistics:  
## AIC BIC SIC HQIC   
## -5.527448 -5.499592 -5.527497 -5.517223

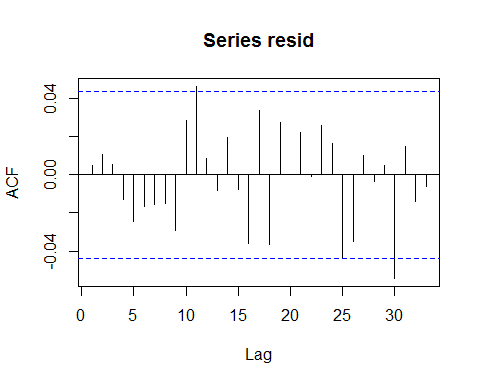
P-values are gradually more significant than 0.05, therefore the the model is adequate.

## e) Rebuild and check the model using Student t innovations.

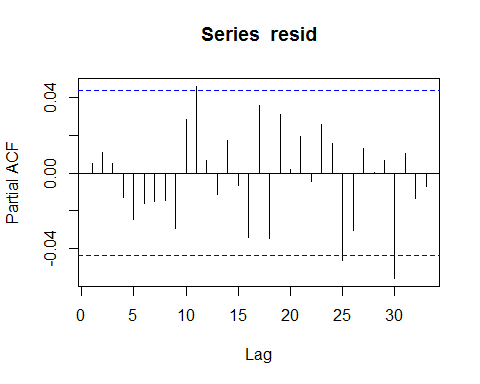
garchMSFTRefined <- garchFit(~arma(1,3)+garch(2,2), data = MSFTLogRet, trace = FALSE, cond.dist = 'std')

## Warning in sqrt(diag(fit$cvar)): созданы NaN

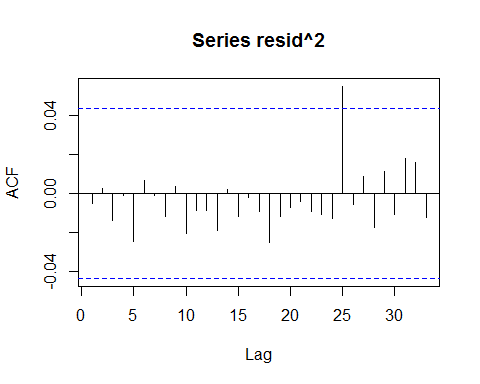
resid <- residuals(garchMSFTRefined, standardize = TRUE)   
  
acf(resid)



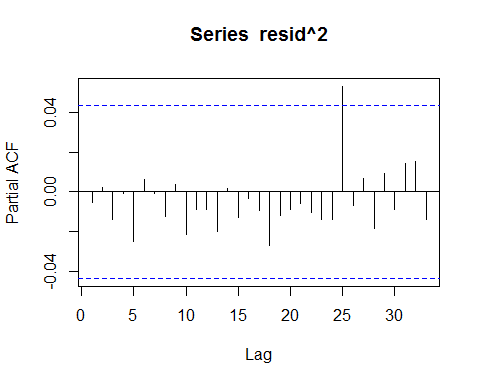
pacf(resid)



acf(resid^2)



pacf(resid^2)



summary(garchMSFTRefined)

##   
## Title:  
## GARCH Modelling   
##   
## Call:  
## garchFit(formula = ~arma(1, 3) + garch(2, 2), data = MSFTLogRet,   
## cond.dist = "std", trace = FALSE)   
##   
## Mean and Variance Equation:  
## data ~ arma(1, 3) + garch(2, 2)  
## <environment: 0x000000001ba3e510>  
## [data = MSFTLogRet]  
##   
## Conditional Distribution:  
## std   
##   
## Coefficient(s):  
## mu ar1 ma1 ma2 ma3   
## 2.8340e-04 5.7008e-01 -5.9404e-01 1.1034e-02 -2.8558e-02   
## omega alpha1 alpha2 beta1 beta2   
## 1.1531e-05 7.4098e-02 1.0000e-08 6.1406e-01 2.6712e-01   
## shape   
## 3.9969e+00   
##   
## Std. Errors:  
## based on Hessian   
##   
## Error Analysis:  
## Estimate Std. Error t value Pr(>|t|)   
## mu 2.834e-04 1.334e-04 2.125 0.0336 \*   
## ar1 5.701e-01 1.355e-01 4.207 2.59e-05 \*\*\*  
## ma1 -5.940e-01 1.368e-01 -4.343 1.40e-05 \*\*\*  
## ma2 1.103e-02 2.529e-02 0.436 0.6626   
## ma3 -2.856e-02 2.017e-02 -1.416 0.1569   
## omega 1.153e-05 NA NA NA   
## alpha1 7.410e-02 NA NA NA   
## alpha2 1.000e-08 NA NA NA   
## beta1 6.141e-01 NA NA NA   
## beta2 2.671e-01 NA NA NA   
## shape 3.997e+00 3.414e-01 11.706 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Log Likelihood:  
## 5746.734 normalized: 2.854811   
##   
## Description:  
## Fri Oct 20 16:20:57 2017 by user: satsuma   
##   
##   
## Standardised Residuals Tests:  
## Statistic p-Value   
## Jarque-Bera Test R Chi^2 5787.489 0   
## Shapiro-Wilk Test R W 0.9305872 0   
## Ljung-Box Test R Q(10) 6.808433 0.7433978  
## Ljung-Box Test R Q(15) 12.35386 0.6520709  
## Ljung-Box Test R Q(20) 21.51286 0.367518   
## Ljung-Box Test R^2 Q(10) 2.985734 0.9817578  
## Ljung-Box Test R^2 Q(15) 4.30931 0.9965347  
## Ljung-Box Test R^2 Q(20) 6.179061 0.9986331  
## LM Arch Test R TR^2 3.555197 0.9901935  
##   
## Information Criterion Statistics:  
## AIC BIC SIC HQIC   
## -5.698693 -5.668051 -5.698752 -5.687445

Once again, all the P-values are gradually more significant than 0.05, therefore the the model is adequate.  
Acf graphs give us a reason to suspect this model to have a serial correlation. However Ljung-Box tests give us an opposite result assuring us that it's ok.

## f) Build and check an ARMA-APACRH model (order=2).

autoFitArima(data = MSFTLogRet, max.order = 2, modelType = "aparch", armaP = 1, armaQ = 3)

## $aic  
## AIC   
## -5.532452   
##   
## $p  
## [1] 2  
##   
## $q  
## [1] 2

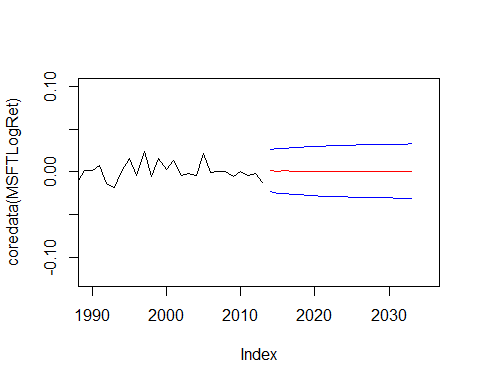
#function gave us (2,2) order  
  
aparchMSFT <- garchFit(~arma(1,3)+aparch(2,2), data = MSFTLogRet, trace = FALSE, delta = 2, include.delta = FALSE)   
summary(aparchMSFT)

##   
## Title:  
## GARCH Modelling   
##   
## Call:  
## garchFit(formula = ~arma(1, 3) + aparch(2, 2), data = MSFTLogRet,   
## delta = 2, include.delta = FALSE, trace = FALSE)   
##   
## Mean and Variance Equation:  
## data ~ arma(1, 3) + aparch(2, 2)  
## <environment: 0x000000002506a7c8>  
## [data = MSFTLogRet]  
##   
## Conditional Distribution:  
## norm   
##   
## Coefficient(s):  
## mu ar1 ma1 ma2 ma3   
## 1.1440e-03 -5.0397e-01 4.9620e-01 8.8429e-03 -2.7950e-02   
## omega alpha1 alpha2 gamma1 gamma2   
## 5.6171e-05 8.2664e-02 1.1993e-01 -1.6565e-01 4.9919e-01   
## beta1 beta2   
## 1.0683e-01 4.5580e-01   
##   
## Std. Errors:  
## based on Hessian   
##   
## Error Analysis:  
## Estimate Std. Error t value Pr(>|t|)   
## mu 1.144e-03 4.949e-04 2.312 0.02079 \*   
## ar1 -5.040e-01 1.536e-01 -3.280 0.00104 \*\*   
## ma1 4.962e-01 1.558e-01 3.186 0.00144 \*\*   
## ma2 8.843e-03 2.902e-02 0.305 0.76059   
## ma3 -2.795e-02 2.634e-02 -1.061 0.28868   
## omega 5.617e-05 1.118e-05 5.023 5.10e-07 \*\*\*  
## alpha1 8.266e-02 2.807e-02 2.944 0.00324 \*\*   
## alpha2 1.199e-01 5.117e-02 2.344 0.01909 \*   
## gamma1 -1.656e-01 1.607e-01 -1.031 0.30268   
## gamma2 4.992e-01 2.482e-01 2.011 0.04431 \*   
## beta1 1.068e-01 1.082e-01 0.988 0.32335   
## beta2 4.558e-01 9.772e-02 4.665 3.09e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Log Likelihood:  
## 5580.413 normalized: 2.772187   
##   
## Description:  
## Wed Oct 18 20:42:36 2017 by user: satsuma   
##   
##   
## Standardised Residuals Tests:  
## Statistic p-Value   
## Jarque-Bera Test R Chi^2 4444.57 0   
## Shapiro-Wilk Test R W 0.9377138 0   
## Ljung-Box Test R Q(10) 10.77998 0.374915   
## Ljung-Box Test R Q(15) 19.3175 0.1997053  
## Ljung-Box Test R Q(20) 27.59446 0.1193522  
## Ljung-Box Test R^2 Q(10) 3.257955 0.9746853  
## Ljung-Box Test R^2 Q(15) 4.0822 0.9974556  
## Ljung-Box Test R^2 Q(20) 4.940168 0.9997475  
## LM Arch Test R TR^2 3.542574 0.9903504  
##   
## Information Criterion Statistics:  
## AIC BIC SIC HQIC   
## -5.532452 -5.499025 -5.532522 -5.520182

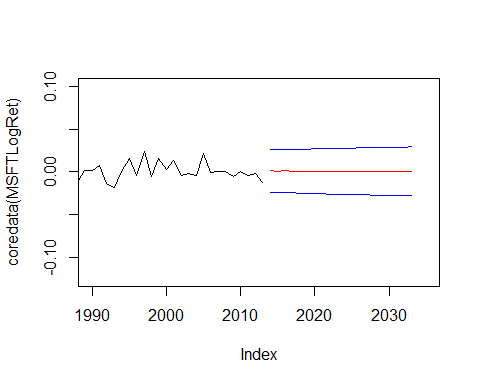
We can see that the AIC criteria is worse than in the case of the Garch Model. However it gives better Ljung-Box test results.

## g) Make and plot forecasts based on the above models.

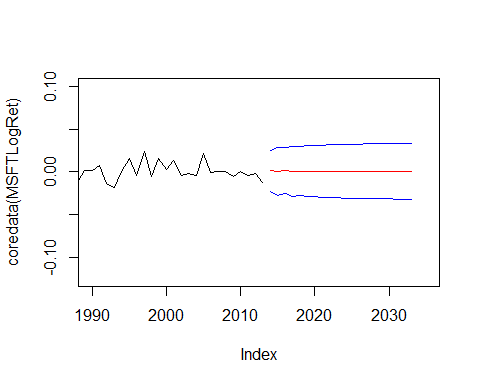
garchPredict <- predict(garchMSFT, n.ahead = 20)  
plot(coredata(MSFTLogRet),xlim=c(1990,2035),type='l')  
lines(garchPredict$meanForecast,x=seq(2014,2033,1), col='Red')  
lines(garchPredict$meanForecast + 2\*garchPredict$standardDeviation,x=seq(2014,2033,1), col='Blue')  
lines(garchPredict$meanForecast - 2\*garchPredict$standardDeviation,x=seq(2014,2033,1), col='Blue')



garchRefinedPredict <- predict(garchMSFTRefined, n.ahead = 20)  
plot(coredata(MSFTLogRet),xlim=c(1990,2035),type='l')  
lines(garchRefinedPredict$meanForecast,x=seq(2014,2033,1), col='Red')  
lines(garchRefinedPredict$meanForecast + 2\*garchRefinedPredict$standardDeviation,x=seq(2014,2033,1), col='Blue')  
lines(garchRefinedPredict$meanForecast - 2\*garchRefinedPredict$standardDeviation,x=seq(2014,2033,1), col='Blue')



aparchPredict <- predict(aparchMSFT, n.ahead = 20)  
plot(coredata(MSFTLogRet),xlim=c(1990,2035),type='l')  
lines(aparchPredict$meanForecast,x=seq(2014,2033,1), col='Red')  
lines(aparchPredict$meanForecast + 2\*aparchPredict$standardDeviation,x=seq(2014,2033,1), col='Blue')  
lines(aparchPredict$meanForecast - 2\*aparchPredict$standardDeviation,x=seq(2014,2033,1), col='Blue')



On the graphs you can the predictions as a red line and the blue lines represent the 95% trust worth interval (+- (2\*sigma)).

As we can see, the most volatile predictions are made with the APARCH model, which is also more reliable.

# Black Box Methods

# 1 Predicting price movements. Get the log returns rt on some stock and S&P 500 index using getSymbols command. Define the direction of the price movement and, similarly, the market movement Mt for S&P 500 index.

I've chosen PPL stocks. Although I have them already, the start year they have before is 2009. I want to take them from the 2010 year. Also S&P for the very same time.

Ppl = getSymbols('PPL', src = 'yahoo', auto.assign = FALSE, from = as.Date("2010-01-03"), to = as.Date("2017-10-17"))

SP2 = getSymbols('^GSPC', src = 'yahoo', auto.assign = FALSE, from = as.Date("2010-01-03"), to = as.Date("2017-10-17"))

Now let's get the log returns. Then we have to convert the data to 1's and 0's. For that task let's write a function to avoid copy-paste.

PplLogRet2 <- getLogReturns(Ad(Ppl))  
SPLogRet2 <- getLogReturns(Ad(SP2))  
  
getDirectionCoeffsVector <- function(data)  
{  
 N <- length(data)  
 idx=c(1:N)[data>0]  
 jdx=c(1:N)[data<=0]  
 A=rep(0,N);A[idx]=1;A[jdx]=0  
 return (A)  
}  
  
PplDir <- getDirectionCoeffsVector(PplLogRet2)  
SpDir <- getDirectionCoeffsVector(SPLogRet2)

## a) Divide the data set into the training and forecast subsets. Clearly state this partition in your report.

Usually the data is divided in the 2:1 proportions train and test data respectively. Let's roughly divide like that: 1200:561.

N <-length(PplDir)  
K <-length(PplDir) - 200  
PplTrainData <- cbind(PplDir[1:K], PplLogRet2[1:K])  
PplTestData <- cbind(PplDir[(K+1):N], PplLogRet2[(K+1):N])  
  
SpTrainData <- cbind(SpDir[1:K], SPLogRet2[1:K])  
SpTestData <- cbind(SpDir[(K+1):N], SPLogRet2[(K+1):N])

## b) Fit a linear logistic regression model for P(Dt = 1) using Dtв€’i, Mtв€’i, i = 1,2,3 as explanatory variables. Use only the training subset for estimation. Discuss statistical significance of the coeп¬ѓcients. Refine the model if needed.

Firstly we have to create the training and test data.

#this function will create a table we'll use for training and forecasts.  
createLagTable <- function(data1, data2)  
{  
 N <- length(data1[,1])  
 return (cbind(Ai = as.vector(data1[,1][4:N]), Ai1 = as.vector(data1[,1][3:(N-1)]), Ai2 = as.vector(data1[,1][2:(N-2)]), Ai3 = as.vector(data1[,1][1:(N-3)]),  
 Mi = as.vector(data2[,1][4:N]), Mi1 = as.vector(data2[,1][3:(N-1)]), Mi2 = as.vector(data2[,1][2:(N-2)]), Mi3 = as.vector(data2[,1][1:(N-3)])))  
}  
  
coeffsTable <- data.frame(createLagTable(PplTrainData, SpTrainData))  
testTable <-data.frame(createLagTable(PplTestData, SpTestData))

Now let's create a model.

PplGlm <- glm(Ai~Ai1+Ai2+Ai3+Mi1+Mi2+Mi3,family="binomial", data = coeffsTable)  
summary(PplGlm)

##   
## Call:  
## glm(formula = Ai ~ Ai1 + Ai2 + Ai3 + Mi1 + Mi2 + Mi3, family = "binomial",   
## data = coeffsTable)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.352 -1.197 1.035 1.145 1.280   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.28686 0.12472 2.300 0.0215 \*  
## Ai1 -0.16788 0.09981 -1.682 0.0926 .  
## Ai2 -0.03917 0.09994 -0.392 0.6951   
## Ai3 -0.19874 0.09991 -1.989 0.0467 \*  
## Mi1 -0.11795 0.10007 -1.179 0.2385   
## Mi2 0.05826 0.10013 0.582 0.5606   
## Mi3 0.05571 0.10015 0.556 0.5780   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2433.3 on 1756 degrees of freedom  
## Residual deviance: 2423.0 on 1750 degrees of freedom  
## AIC: 2437  
##   
## Number of Fisher Scoring iterations: 3

We can see that intercept and 2 other coefficients are significant: first and third lag. This means that to refine this model we'll have to throw away all the other coefficients, only leaving the significant ones.

PplGlm <- glm(Ai~Ai1+Ai3,family="binomial", data = coeffsTable)  
summary(PplGlm)

##   
## Call:  
## glm(formula = Ai ~ Ai1 + Ai3, family = "binomial", data = coeffsTable)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.296 -1.209 1.063 1.146 1.225   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.27455 0.08530 3.219 0.00129 \*\*  
## Ai1 -0.19990 0.09579 -2.087 0.03690 \*   
## Ai3 -0.18638 0.09579 -1.946 0.05170 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2433.3 on 1756 degrees of freedom  
## Residual deviance: 2425.2 on 1754 degrees of freedom  
## AIC: 2431.2  
##   
## Number of Fisher Scoring iterations: 3

## c) Using the model, make predictions for the forecast subset. Specify the threshold you apply. Compute the forecast error.

Our predictions are going to be non-discrete numbers from 0 to 1. In order to match them to 0s and 1s we have right now we'll divide them by them being more than 0.5 (=1) and less (=0).

PplGlmPredict <- predict(PplGlm, newdata = testTable, type="response")  
  
m1P=rep("Down",length(testTable$Ai))  
m1P[PplGlmPredict>0.5]="Up"  
tb <- table(m1P,testTable$Ai)  
glmErrorRate <- (tb[1,2] + tb[2,1])/sum(tb)  
  
glmErrorRate #correct

## [1] 0.4568528

The error rate is roughly 46 percents. This is a huge value, however we can hope for the model to be

## d) Apply the linear discriminant analysis instead of log regression in items 1a-1c. Compute the forecast error.

Let's now apply the LDA analysis with the refined formula. The predictions are going to give us concrete discrete numbers (model$class == (0|1)) this time.

PplLda <- lda(Ai~Ai1+Ai3, data = coeffsTable)  
PplLdaPredict <- predict(object = PplLda, newdata = testTable)  
  
PplLdaP=rep("Down",length(testTable$Ai))  
PplLdaP[PplLdaPredict$class==1]="Up"  
tbLda <- table(PplLdaP,testTable$Ai)  
ErrorRateLda <- (tbLda[1,2] + tbLda[2,1])/sum(tbLda)  
ErrorRateLda

## [1] 0.4568528

With this model we get the same result as GLM. These are bad results.

## e) Apply the quadratic discriminant analysis instead of log regression in items 1a-1c. Compute the forecast error.

Now we'll apply QDA model. The predicted values are once again going to be either 0 or 1. It's also a non-linar model therefore we can't refine it.

PplQda <- qda(Ai~Ai1+Ai3, data = coeffsTable)  
PplQdaPredict <- predict(object = PplQda, newdata = testTable)  
  
PplQdaP=rep("Down",length(testTable$Ai))  
PplQdaP[PplQdaPredict$class == 1]="Up"  
tbQda <- table(PplQdaP,testTable$Ai)  
ErrorRateQda <- (tbQda[1,2] + tbQda[2,1])/sum(tbQda)  
ErrorRateQda

## [1] 0.4568528

In case of QDA we get the same result as we got previously for the previous models.

## f) Employ a 4-3-1 look-forward neural network with direct link for P(Dt = 1) instead of log regression in items 1a-1c. Build a model for two (i = 1,2) lagged variables D, M. Compute the forecast error.

Now we'll do a NNET model this time. Also we'll use just the log data itself. The predictions are once again going to be non-discrete from 0 to 1. Therefore we'll divide them by being more and less than 0.5 just like we did previously.

Ppl.x<-cbind(Ai1 = coeffsTable$Ai1,Ai2 = coeffsTable$Ai2,Mi1 = coeffsTable$Mi1, Mi2 = coeffsTable$Mi2)  
Ppl.nn<-nnet(Ppl.x,coeffsTable$Ai1,size=2,linout=T,skip=T,maxit=10000,decay=1e-2,reltol=1e-7,abstol=1e-7,range=1.0)

## # weights: 17  
## initial value 2073.788234   
## iter 10 value 18.252118  
## iter 20 value 0.085677  
## iter 30 value 0.027311  
## iter 40 value 0.021518  
## iter 50 value 0.013497  
## iter 60 value 0.010038  
## iter 70 value 0.010015  
## iter 80 value 0.010014  
## iter 90 value 0.010000  
## final value 0.010000   
## converged

PplNnetPredict <- predict(Ppl.nn,cbind(Ai1 = testTable$Ai1,Ai2 = testTable$Ai2,Mi1 = testTable$Mi1, Mi2 = testTable$Mi2))  
  
PplNnetP=rep("Down",length(testTable$Ai))  
PplNnetP[PplNnetPredict > 0.5]="Up"  
tbNnet <- table(PplNnetP,testTable$Ai)  
ErrorRateNnet <- (tbNnet[1,2] + tbNnet[2,1])/sum(tbNnet)  
ErrorRateNnet

## [1] 0.5228426

We can see that it gets worse with the neural networks model: 52% wrong answers

## g) Compare the predictive power (i.e, forecast error) of methods 1b,1d,1e,1f.

Overal, the predictions we got are really bad. The best we've managed to achieve is 54% accuracy, which is very close to the situation where we would've been just guessing. Neural networks predictions are even worse, but it's no surprise since it requires a more thorough set up. GLM, LDA and QDA gave us the same results.