Hw 5

Xiang Yang Ng

June 2, 2018

Downloading all the libraries and set the working directory

```
setwd("/Users/Xiang/OneDrive/Desktop/Econ-144/Homework/Hw 5")
library(lattice)
library(foreign)
library(MASS)
library(car)
```

```
## Loading required package: carData
```

```
require(stats)
require(stats4)
```

```
## Loading required package: stats4
```

```
library(KernSmooth)
```

```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```
library(fastICA)
library(cluster)
library(leaps)
library(mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-23. For overview type 'help("mgcv-package")'.
```

```
library(rpart)
library(pan)
library(mgcv)
library(DAAG)
```

```
##
## Attaching package: 'DAAG'
```

```
## The following object is masked from 'package:car':
##
##     vif
```

```
## The following object is masked from 'package:MASS':
##
##     hills
```

```
library("TTR")
library(tis)
```

```
##
## Attaching package: 'tis'
```

```
## The following object is masked from 'package:TTR':
##
##     lags
```

```
## The following object is masked from 'package:mgcv':
##
##     ti
```

```
require("datasets")
require(graphics)
library("forecast")
```

```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:tis':
##
##     easter
```

```
## The following object is masked from 'package:nlme':
##
##     getResponse
```

```
#install.packages("astsa")
#require(astsa)
library(fpp)
```

```
## Loading required package: fma
```

```
##
## Attaching package: 'fma'
```

```
## The following objects are masked from 'package:DAAG':
##
##     milk, ozone
```

```
## The following objects are masked from 'package:MASS':
##
##     cement, housing, petrol
```

```
## Loading required package: expsmooth
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: tseries
```

```
library(strucchange)
```

```
## Loading required package: sandwich
```

```
library(tseries)
library(rugarch)
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'rugarch'
```

```
## The following object is masked from 'package:stats':
##
##     sigma
```

Problem 1

```
#Clear the previous environment
rm(list = ls(all=T))
library(fGarch)
```

```
## Loading required package: timeDate
```

```
##
## Attaching package: 'timeDate'
```

```
## The following objects are masked from 'package:tis':
##
##     dayOfWeek, dayOfYear, isHoliday
```

```
## Loading required package: timeSeries
```

```
##
## Attaching package: 'timeSeries'
```

```
## The following object is masked from 'package:zoo':
##
##     time<-
```

```
## The following objects are masked from 'package:tis':
##
##     description, interpNA
```

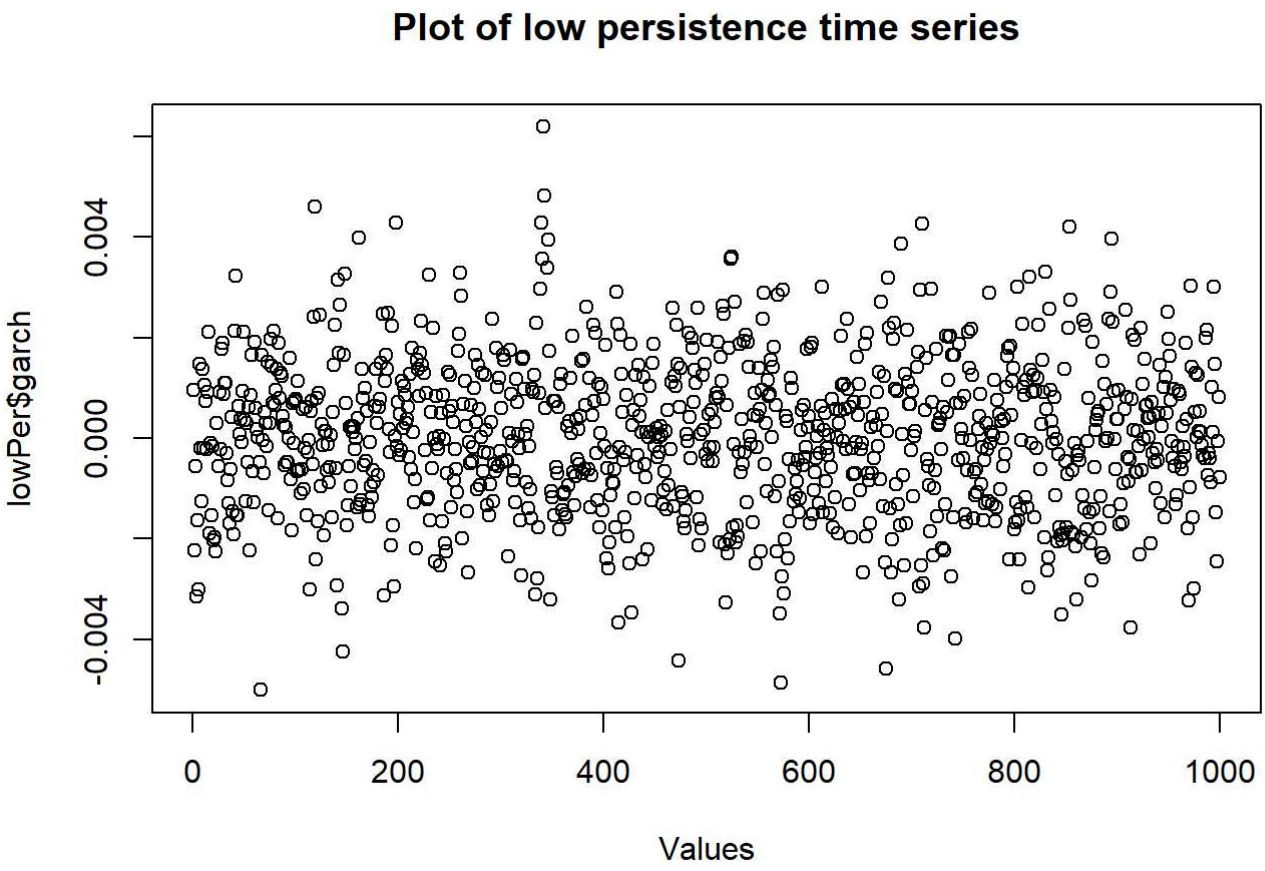
```
## Loading required package: fBasics
```

```
##
## Attaching package: 'fBasics'
```

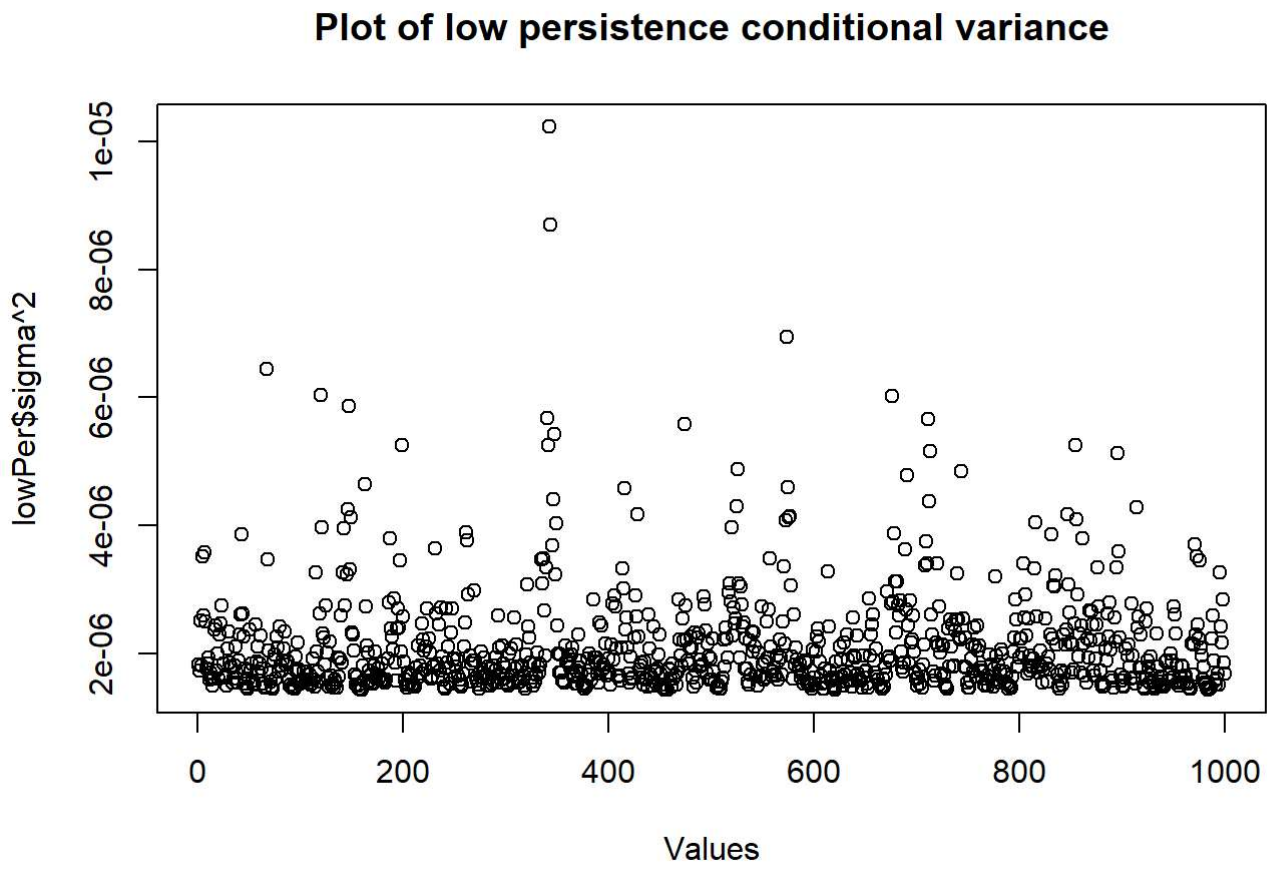
```
## The following object is masked from 'package:TTR':
##
##     volatility
```

```
## The following object is masked from 'package:car':
##
##     densityPlot
```

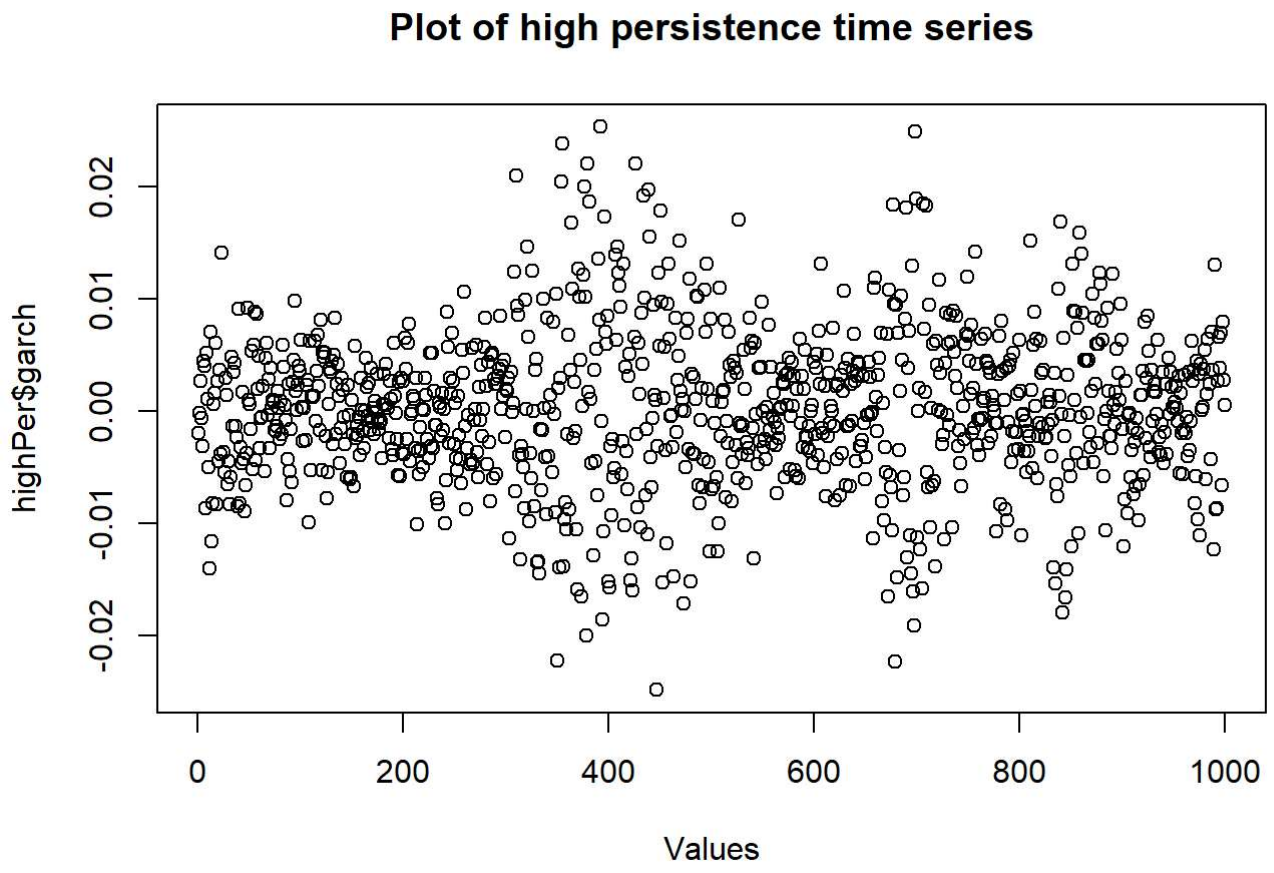
```
# model=ugarchspec(
# variance.model = list(model = "sGARCH", garchOrder = c(2, 2)),
# mean.model = list(armaOrder = c(2, 2), include.mean = TRUE),
# distribution.model = "sstd")
#
# #Sanity check: explore the model parameters
# #mxreg is the number of external regressors
# model@model$pars

#Crete the low and high persistence GARCH simulations
lowPer = garchSim(spec = garchSpec(model = list(alpha = 0.2, beta = 0.3)), n = 1000, n.start = 1
00, extended = TRUE)
highPer = garchSim(spec = garchSpec(model = list(alpha = 0.1, beta = 0.88)), n = 1000, n.start =
 100, extended = TRUE)

#plotting the scatterplots and histograms of the low and persistence simulations
plot(lowPer$garch, main = "Plot of low persistence time series", xlab = "Values")
```



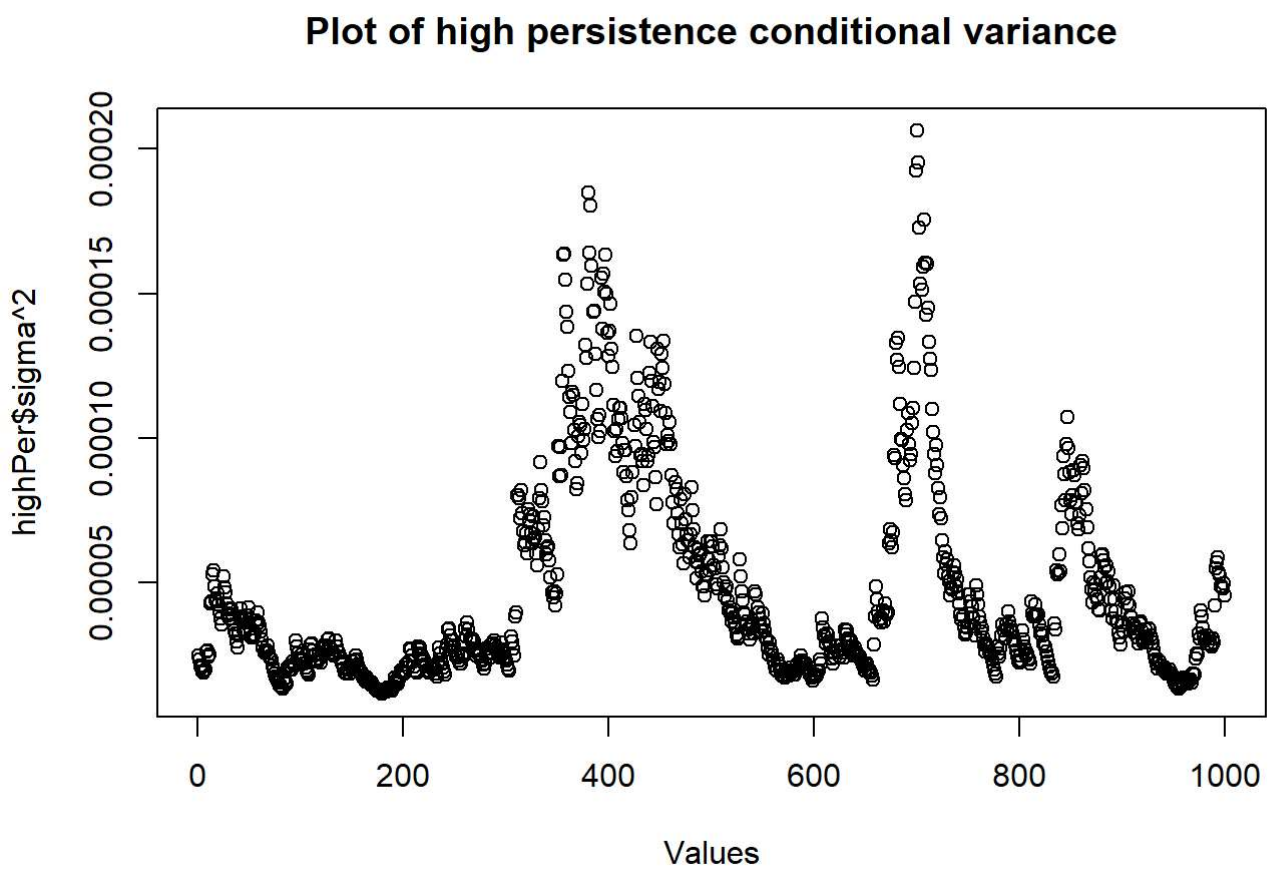

```
plot(lowPer$sigma^2, main = "Plot of low persistence conditional variance", xlab = "Values")
```



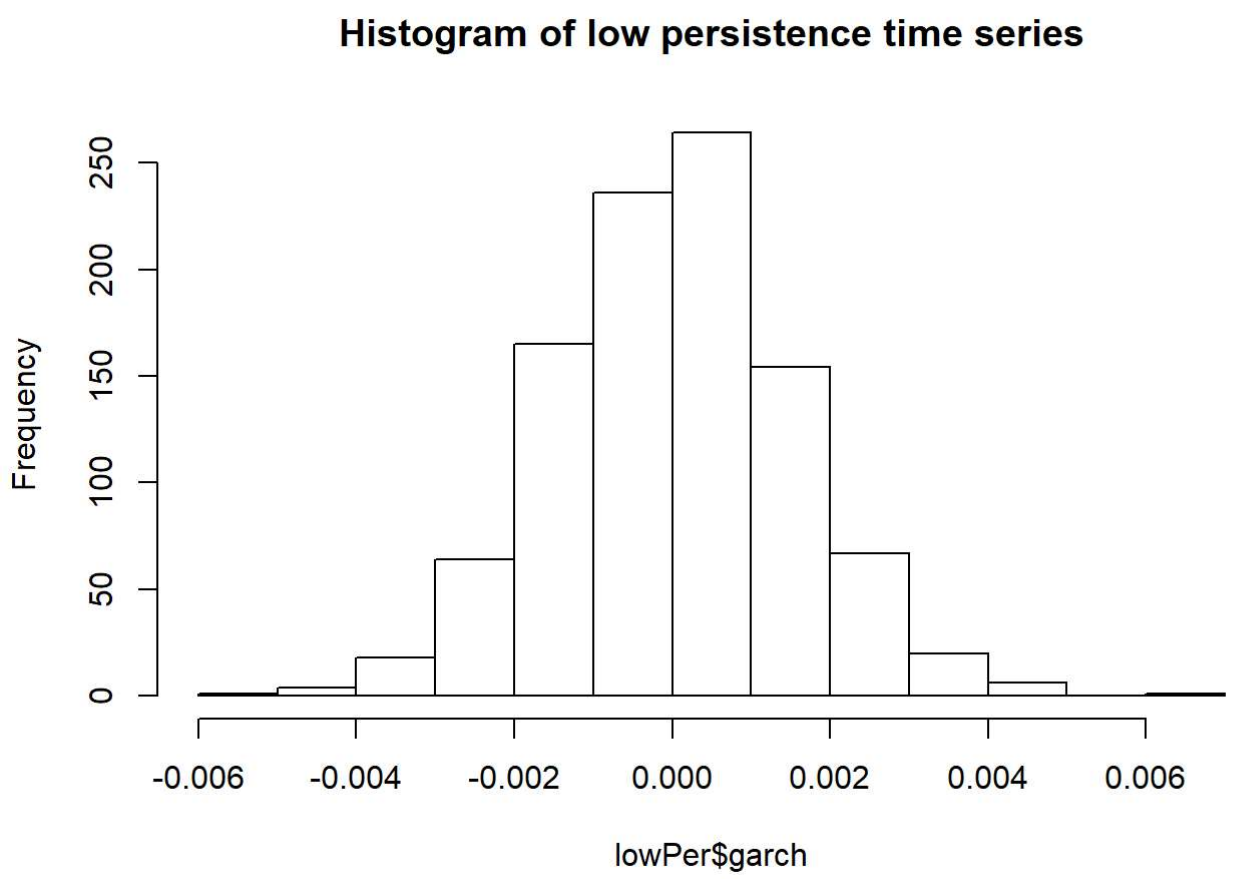

```
plot(highPer$garch, main = "Plot of high persistence time series", xlab = "Values")
```



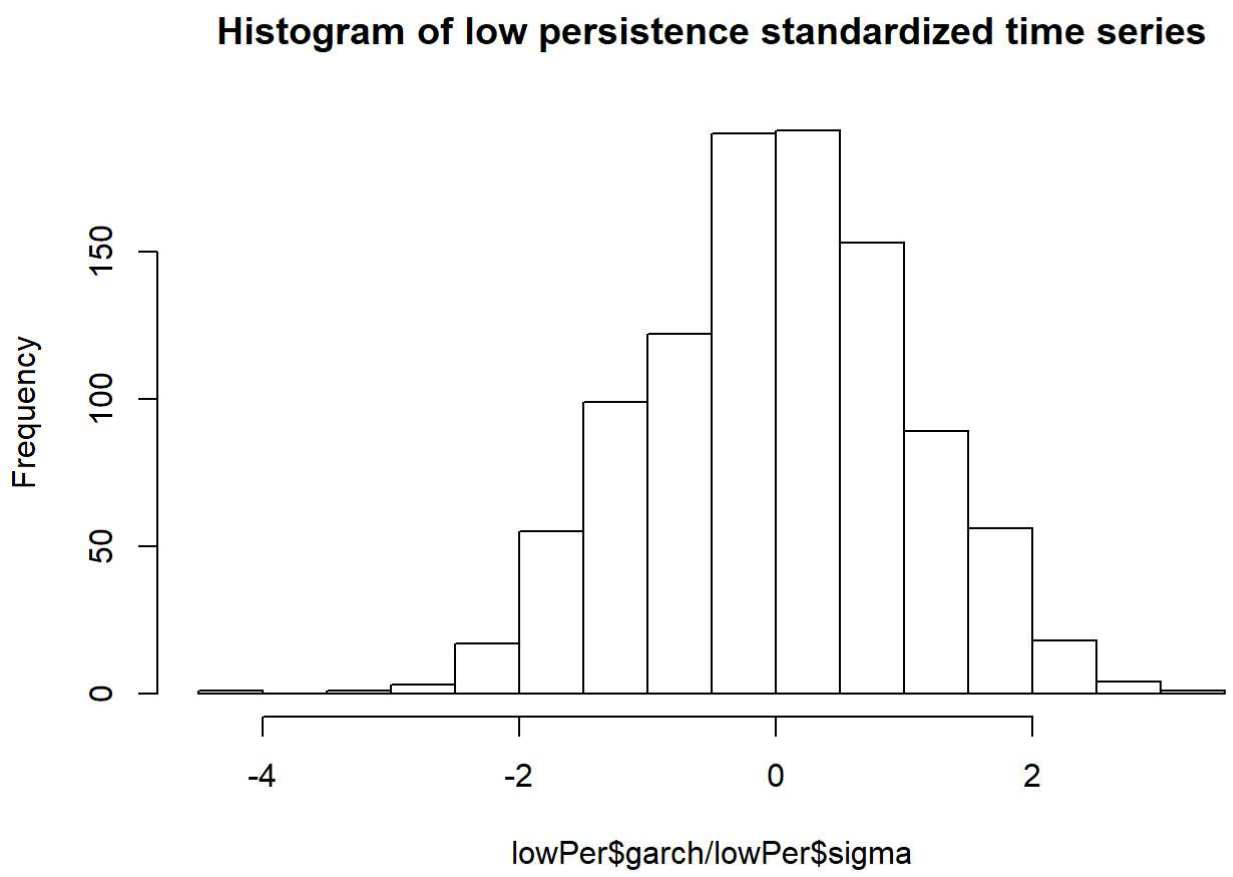

```
plot(highPer$sigma^2, main = "Plot of high persistence conditional variance", xlab = "Values")
```



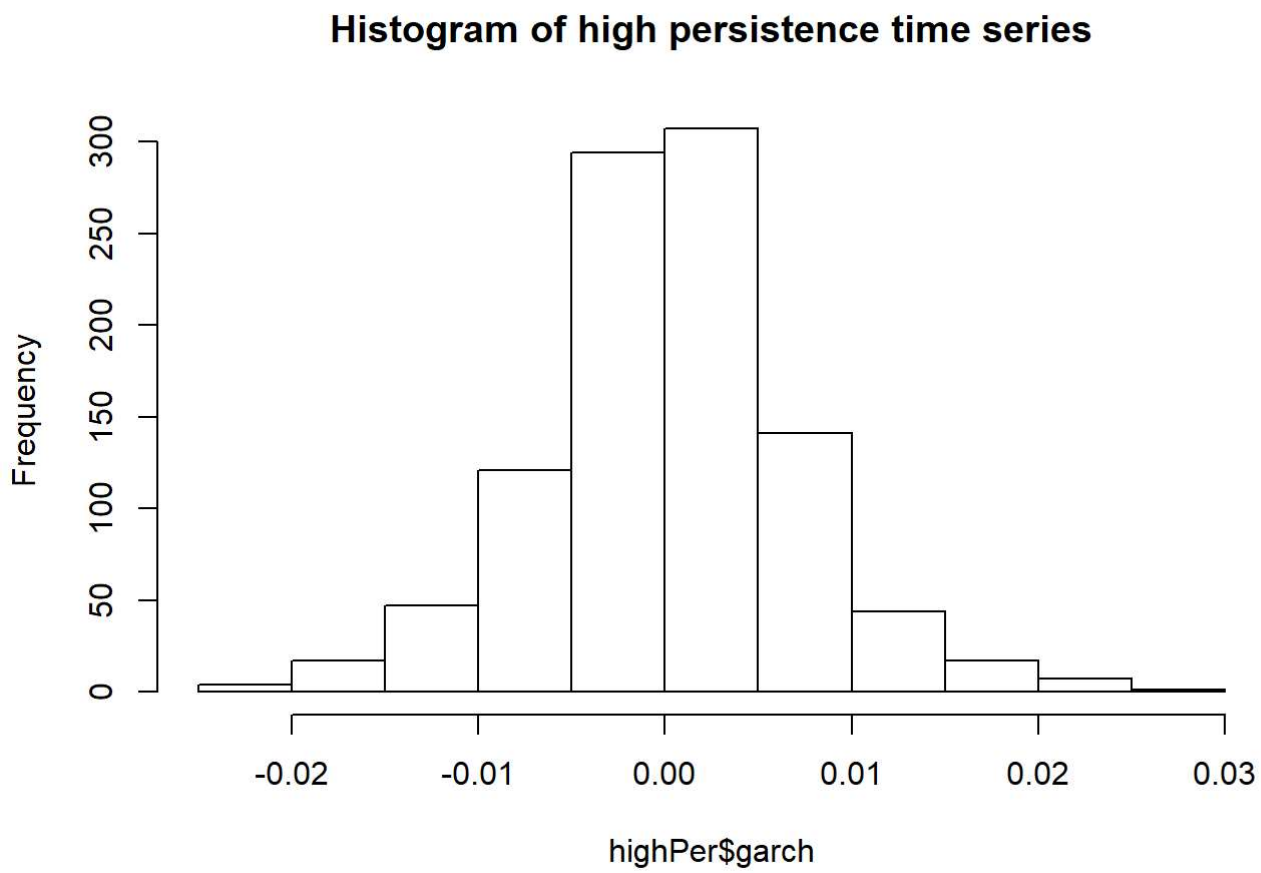

```
hist(lowPer$garch, main = "Histogram of low persistence time series")
```



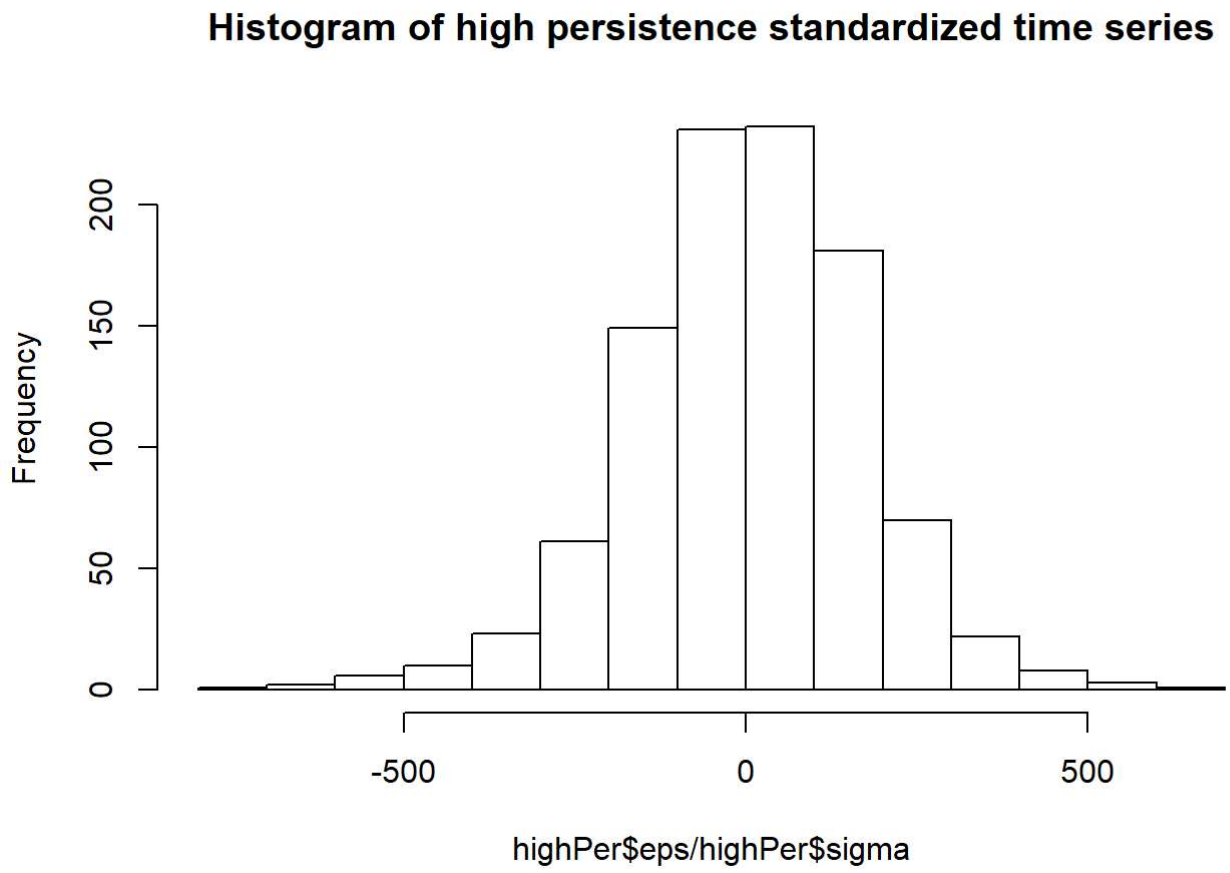

```
hist(lowPer$garch/lowPer$sigma, main = "Histogram of low persistence standardized time series")
```




```
hist(highPer$garch, main = "Histogram of high persistence time series")
```

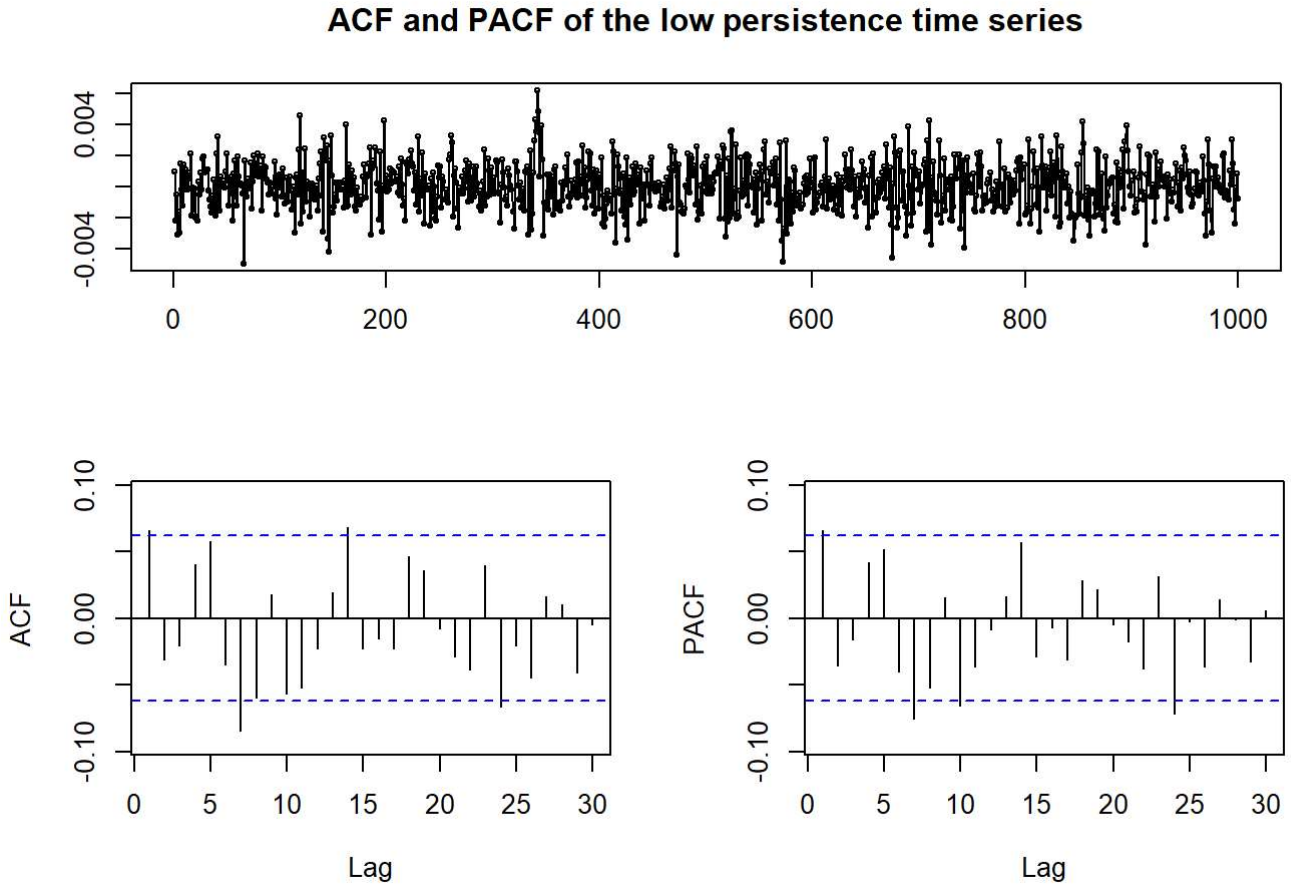



```
hist(highPer$eps/highPer$sigma, main = "Histogram of high persistence standardized time series")
```

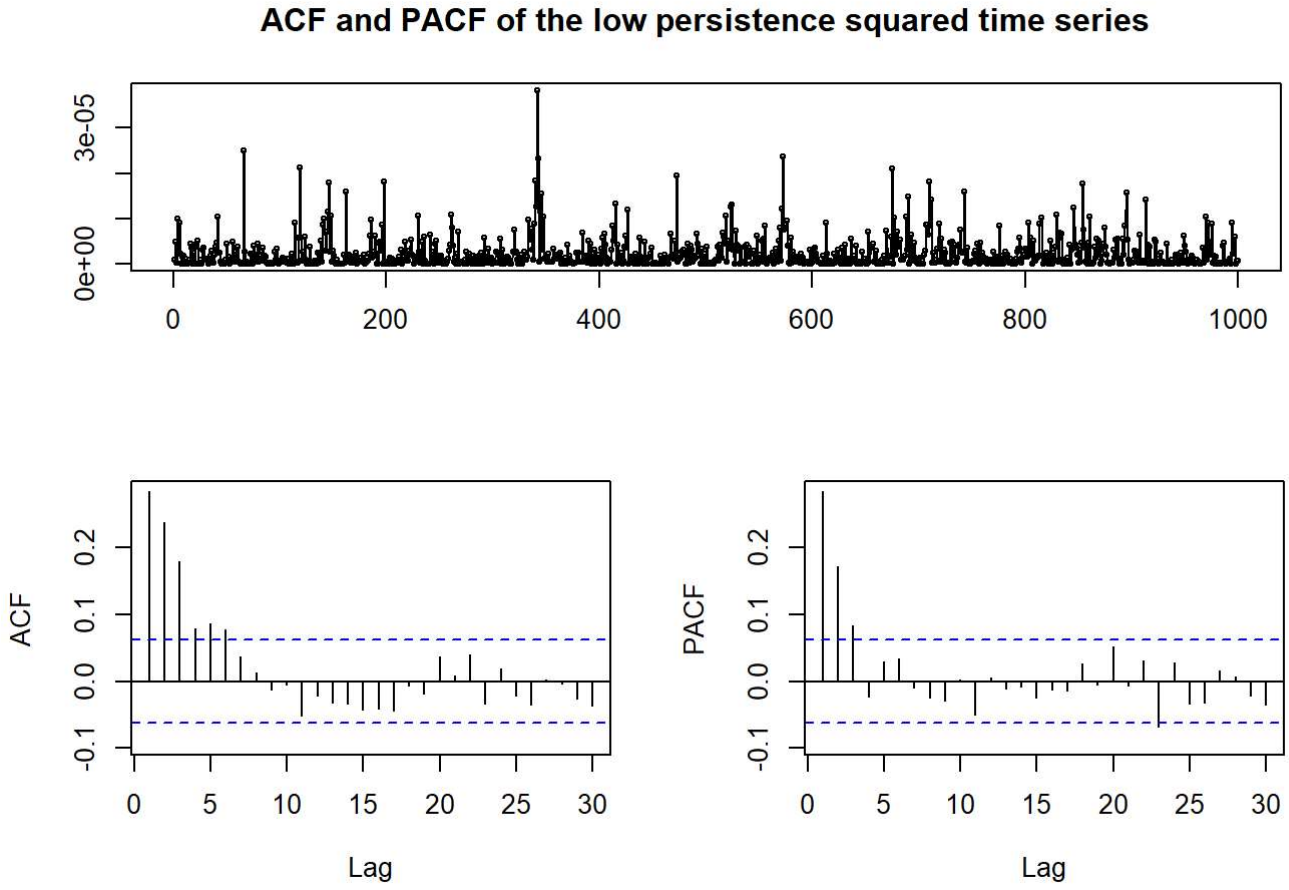


Looking at the plots of low persistence, we see that the time series plot is scattered rather evenly and the conditional variance plot is less volatile, compared to those of the high persistent plots. The histograms of both low and high persistence time series plots show that it cannot be modeled as a normal distribution, but after standardizing, it seems that it can be modeled by a normal distribution.

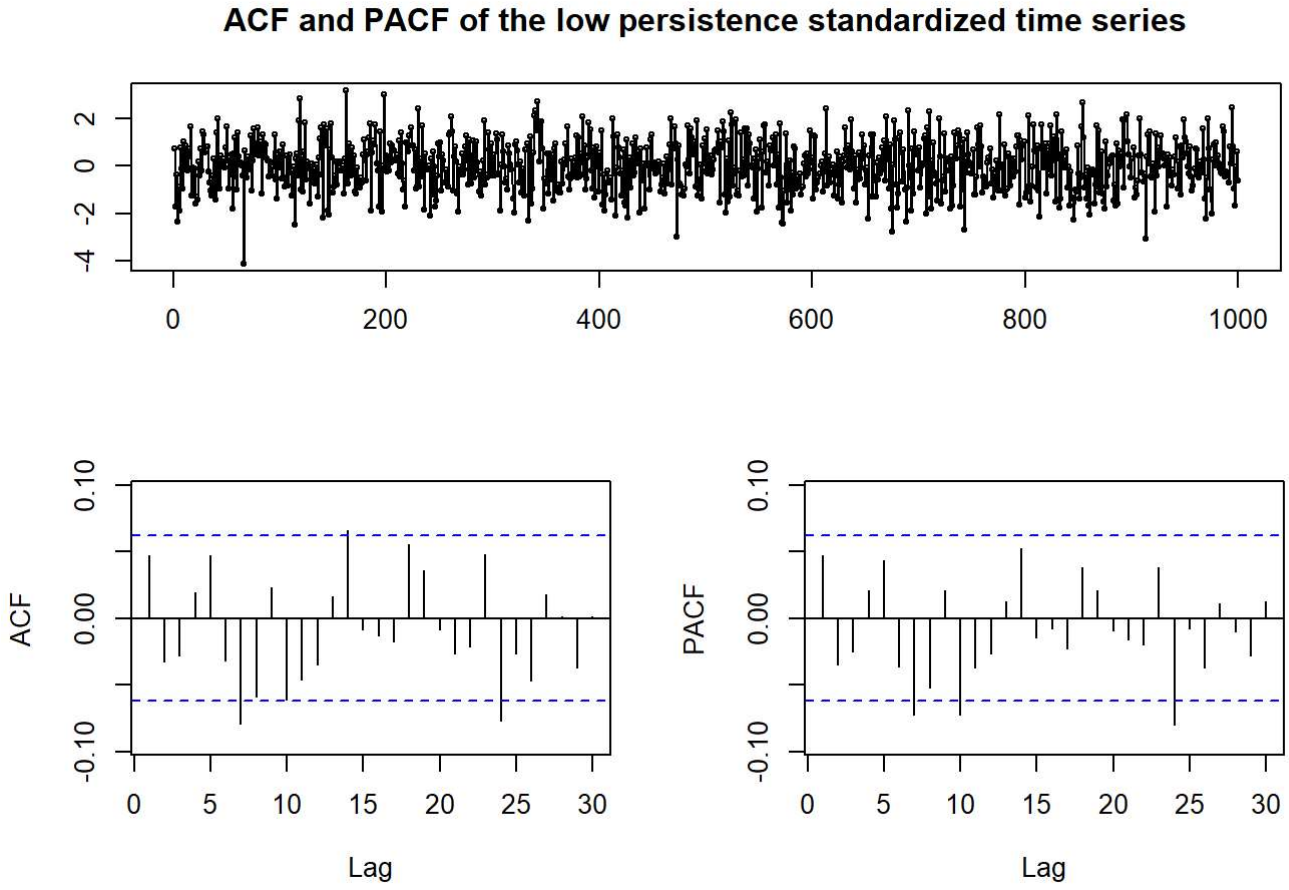
```
tsdisplay(lowPer$garch, main = "ACF and PACF of the low persistence time series")
```



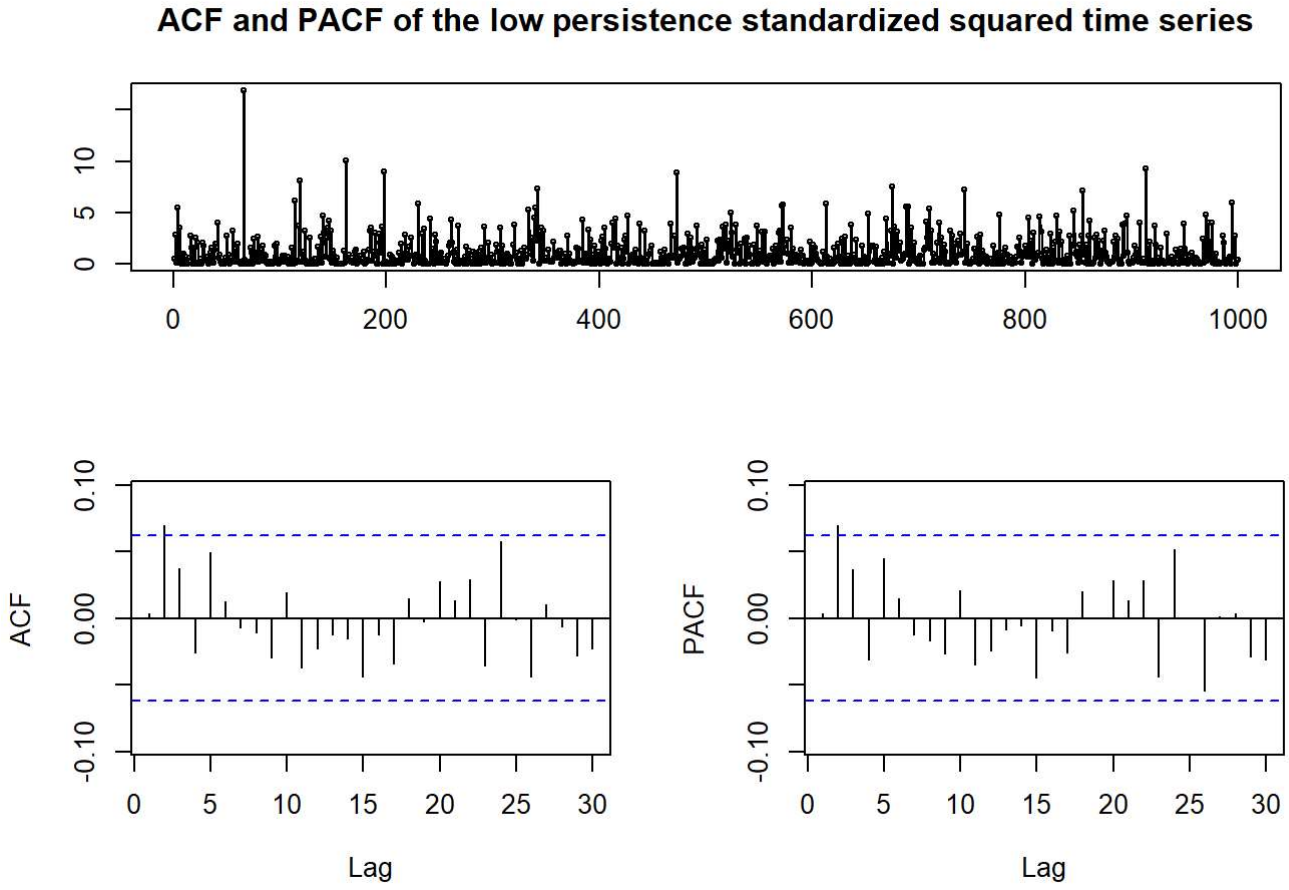

```
tsdisplay(lowPer$garch^2, main = "ACF and PACF of the low persistence squared time series")
```



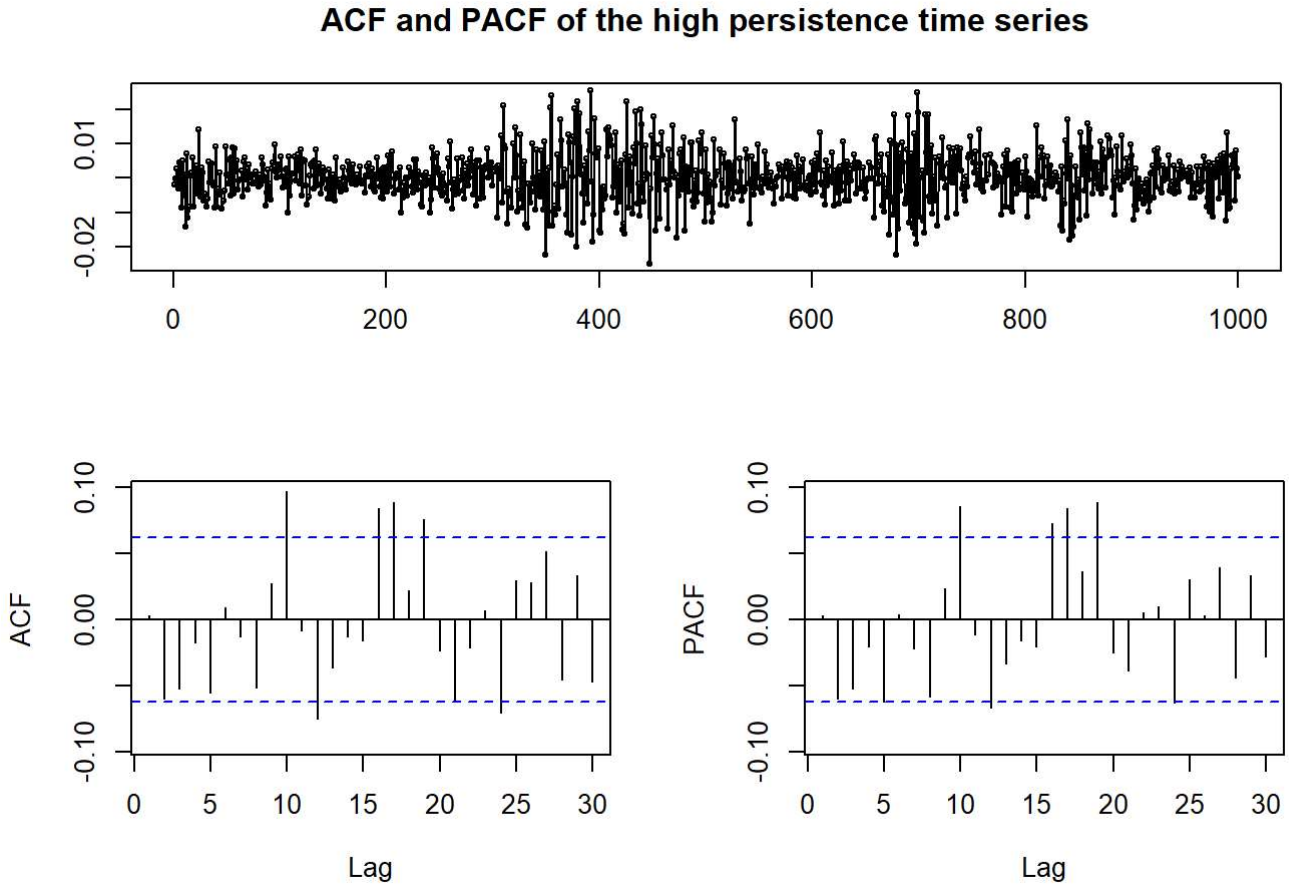

```
tsdisplay(lowPer$garch/lowPer$sigma, main = "ACF and PACF of the low persistence standardized ti
me series")
```



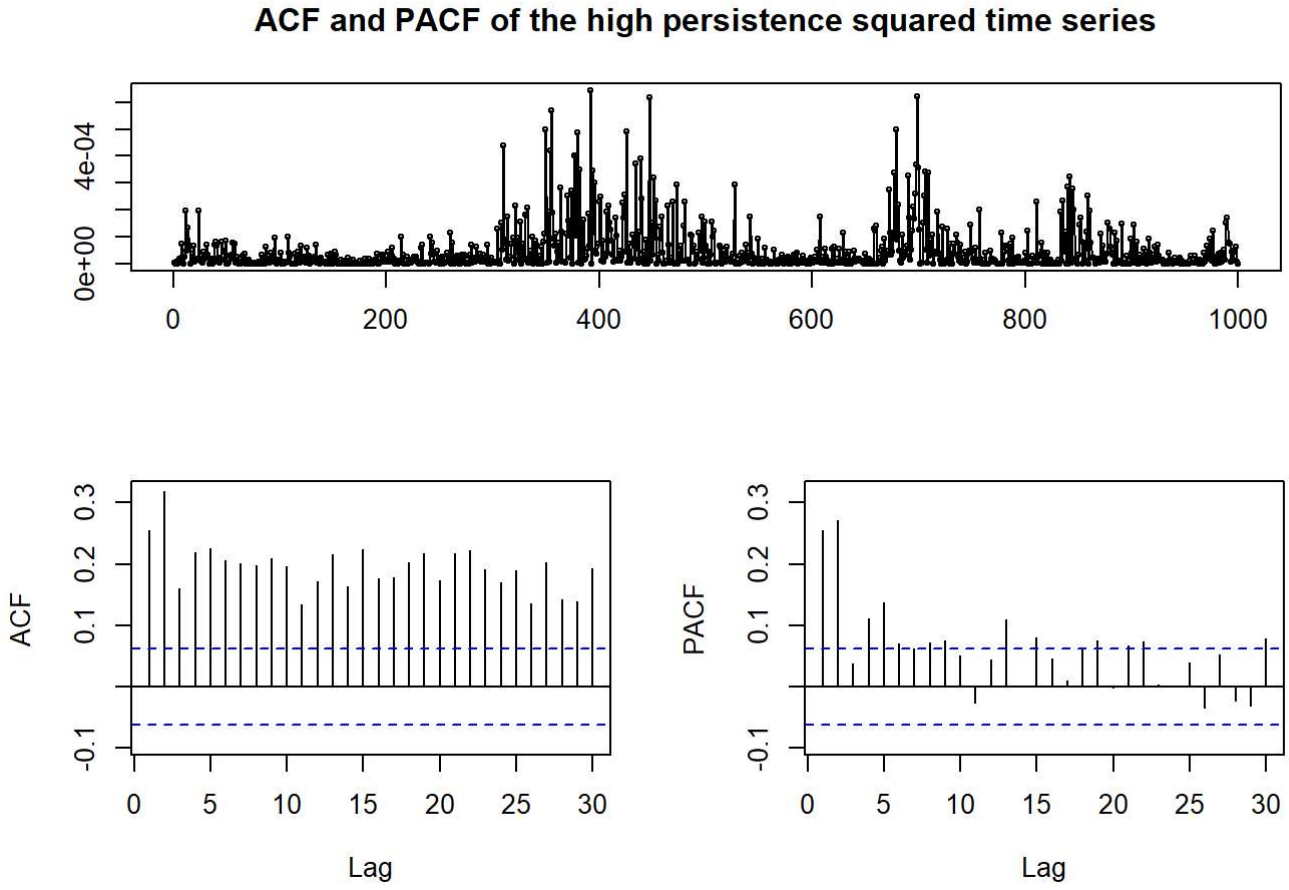

```
tsdisplay((lowPer$garch/lowPer$sigma)^2, main = "ACF and PACF of the low persistence standardize
d squared time series")
```



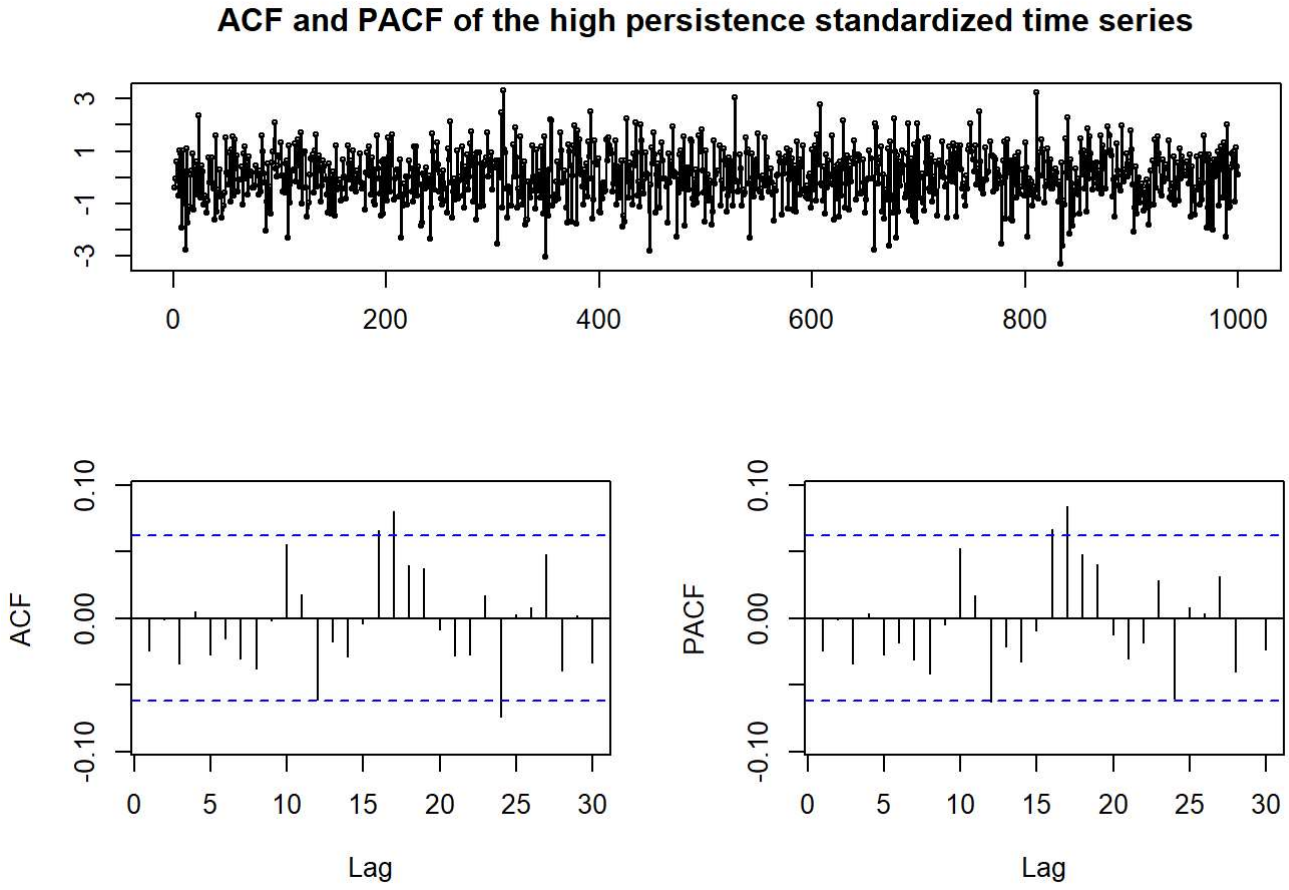

```
tsdisplay(highPer$garch, main = "ACF and PACF of the high persistence time series")
```



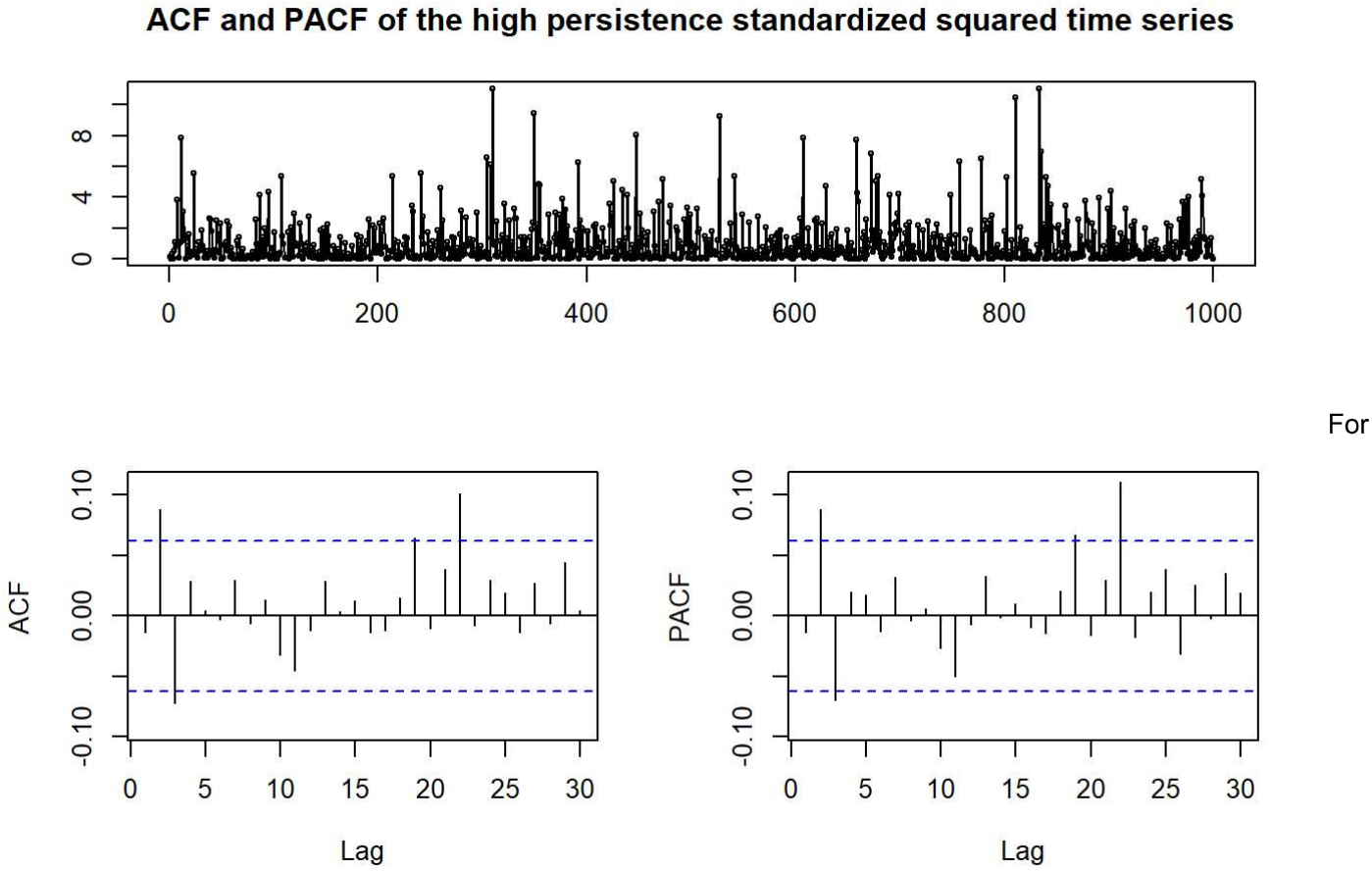

```
tsdisplay(highPer$garch^2, main = "ACF and PACF of the high persistence squared time series")
```




```
tsdisplay(highPer$garch/highPer$sigma, main = "ACF and PACF of the high persistence standardized
 time series")
```




```
tsdisplay((highPer$garch/highPer$sigma)^2, main = "ACF and PACF of the high persistence standard
ized squared time series")
```



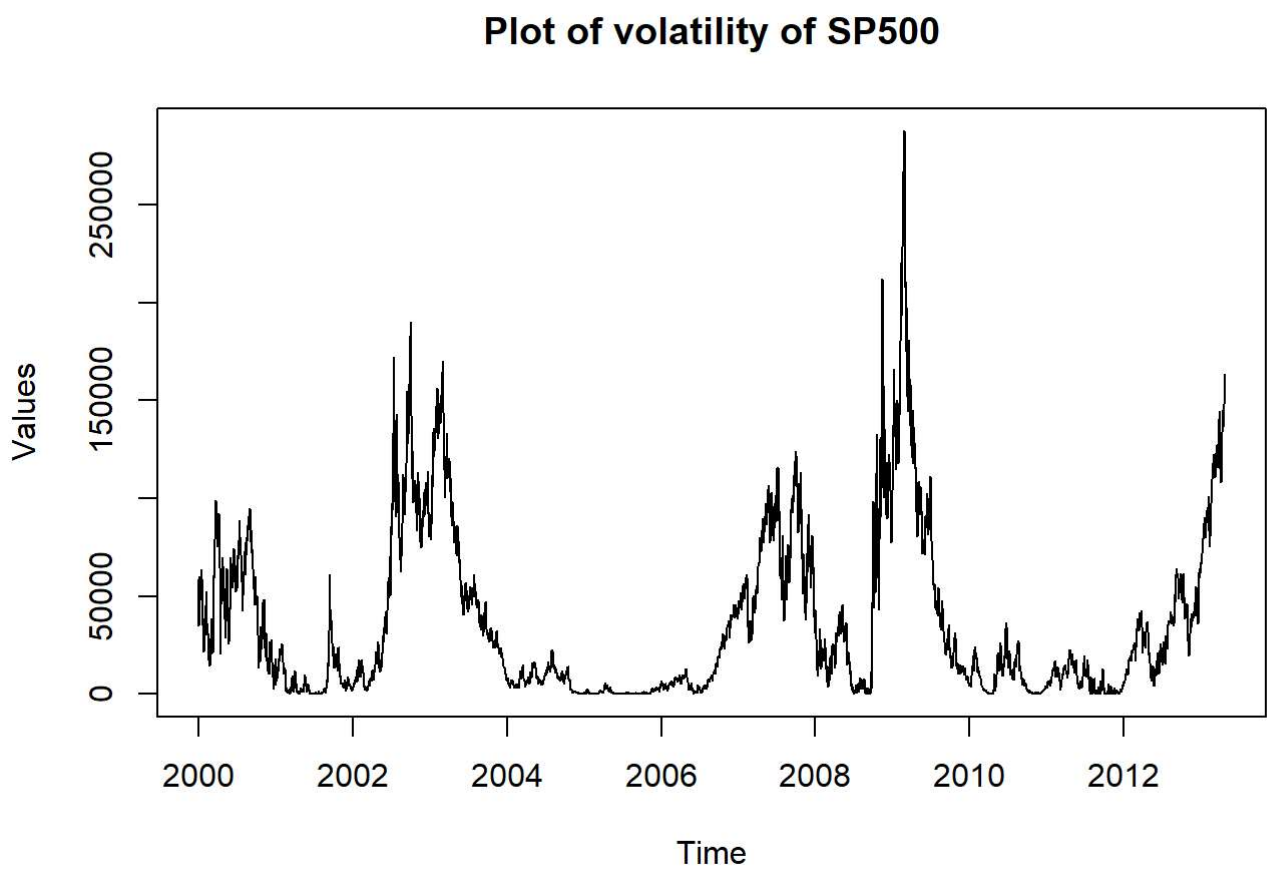
For both the low and high persistence ACF and PACF of the time series, we see that there is almost no spikes. But after squaring them, we see that there are spikes that are apparent, especially the high persistence one. Standardizing them thus eliminates those spikes.

Problem 2

```
#Clear the pervious environment and download the SP500 data
library(openxlsx)
rm(list = ls(all=T))
data = read.xlsx("Chapter14_exercises_data.xlsx", sheet = "Exercise 3, 4, 10")
```

Obtain the time series and plot the volatility

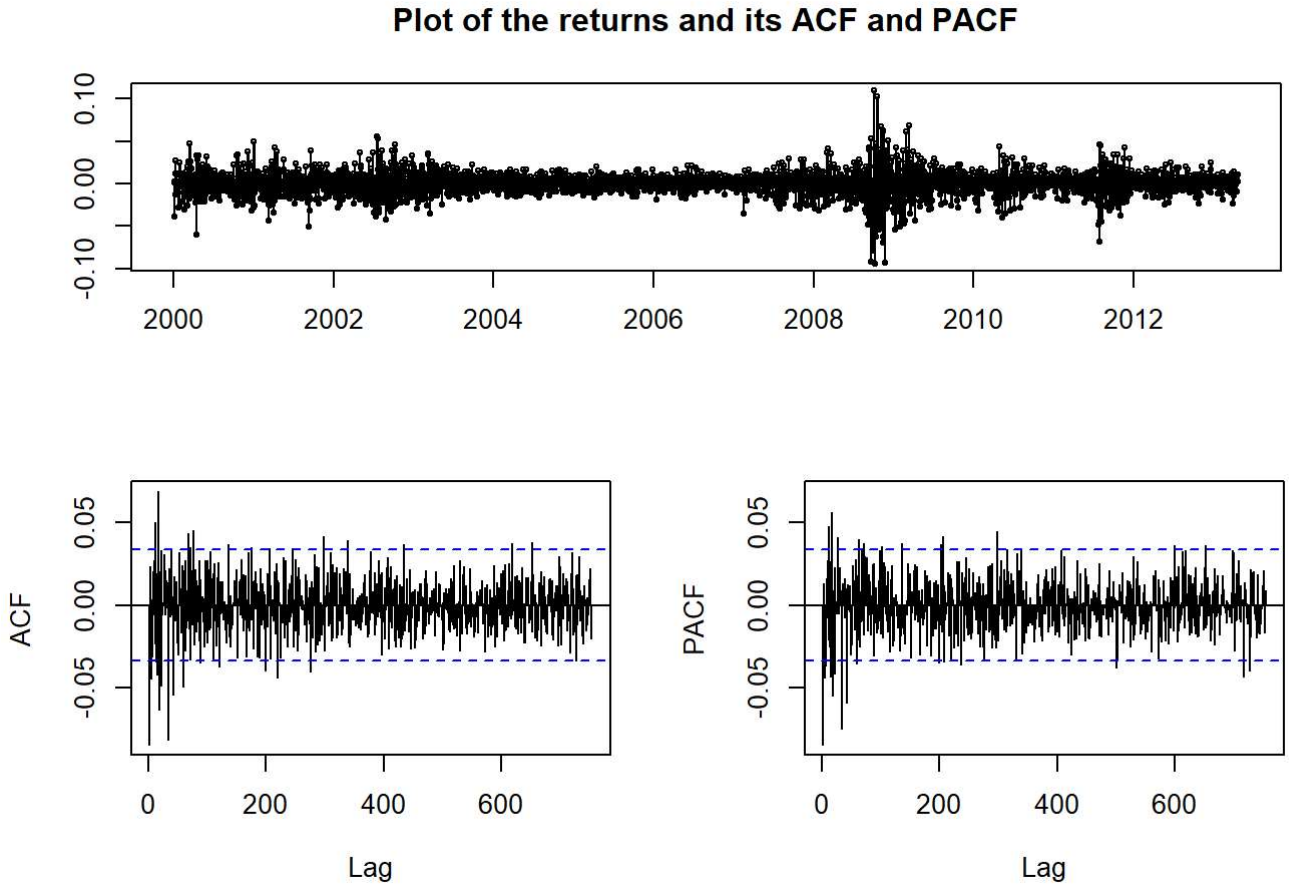
```
close_ts = ts(data$Close, start = 2000, freq = 252)
plot(volatility(close_ts, n = 10, calc = "close", N = 252, mean0 = FALSE), ylab = "Values", main
 = "Plot of volatility of SP500")
```



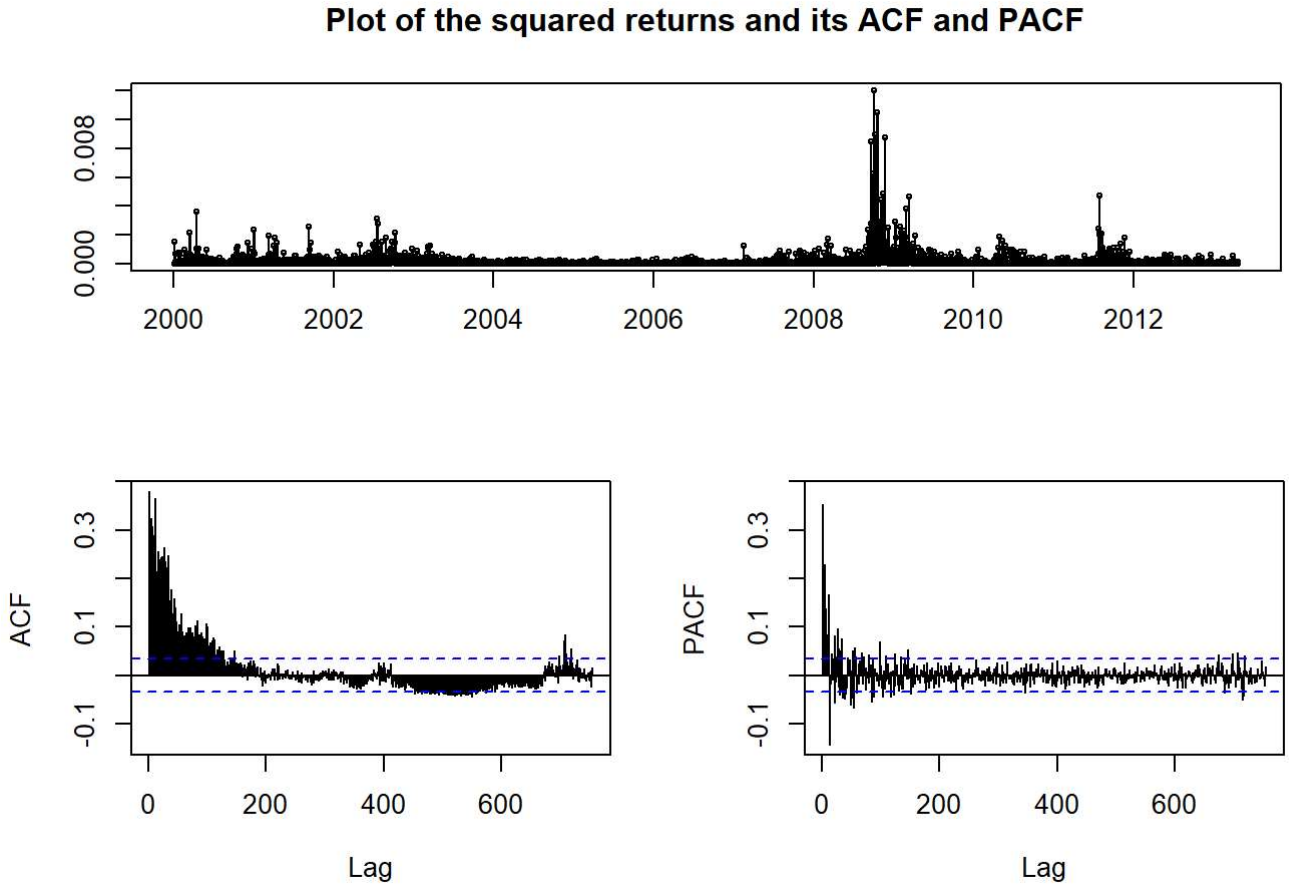
The volatility from the past is less volatile compared to more recent times. This means that the errors are heteroskedastic.

Compute the ACF and PACF of the returns and the squred returns

```
close_returns_ts = diff(log(close_ts))
tsdisplay(close_returns_ts, main = "Plot of the returns and its ACF and PACF")
```




```
tsdisplay(close_returns_ts^2, main = "Plot of the squared returns and its ACF and PACF")
```



The ACF and PACF of the returns seem to have not much spikes, but after squaring, the spikes become more apparent, which is evident of heteroskedastic errors.

From the PACF of the squared returns, we first fit an ARCH(14) to see if this wipes out the dynamics

```
model=ugarchspec(
variance.model = list(model = "sGARCH", garchOrder = c(14, 0)),
mean.model = list(armaOrder = c(1, 2), include.mean = TRUE),
distribution.model = "sstd")

#Sanity check: explore the model parameters
model@model$pars
```

```
##          Level Fixed Include Estimate LB UB
## mu           0     0       1        1 NA NA
## ar1          0     0       1        1 NA NA
## ma1          0     0       1        1 NA NA
## ma2          0     0       1        1 NA NA
## arfima       0     0       0        0 NA NA
## archm        0     0       0        0 NA NA
## mxreg        0     0       0        0 NA NA
## omega        0     0       1        1 NA NA
## alpha1       0     0       1        1 NA NA
## alpha2       0     0       1        1 NA NA
## alpha3       0     0       1        1 NA NA
## alpha4       0     0       1        1 NA NA
## alpha5       0     0       1        1 NA NA
## alpha6       0     0       1        1 NA NA
## alpha7       0     0       1        1 NA NA
## alpha8       0     0       1        1 NA NA
## alpha9       0     0       1        1 NA NA
## alpha10      0     0       1        1 NA NA
## alpha11      0     0       1        1 NA NA
## alpha12      0     0       1        1 NA NA
## alpha13      0     0       1        1 NA NA
## alpha14      0     0       1        1 NA NA
## beta         0     0       0        0 NA NA
## gamma        0     0       0        0 NA NA
## eta1         0     0       0        0 NA NA
## eta2         0     0       0        0 NA NA
## delta        0     0       0        0 NA NA
## lambda       0     0       0        0 NA NA
## vxreg        0     0       0        0 NA NA
## skew         0     0       1        1 NA NA
## shape        0     0       1        1 NA NA
## ghlambda     0     0       0        0 NA NA
## xi           0     0       0        0 NA NA
```

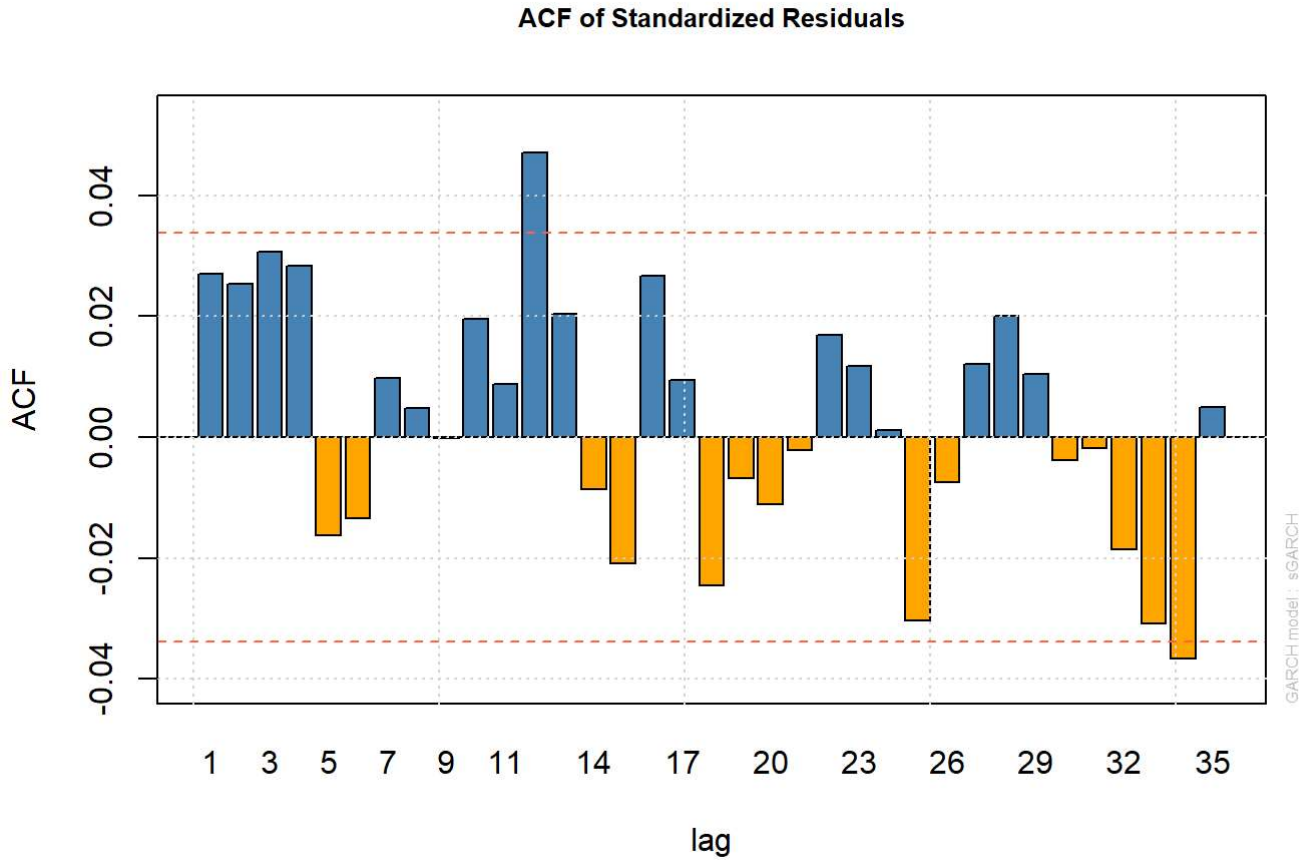
```
# Fit the model to the data
modelfit=ugarchfit(spec=model,data=close_returns_ts)
modelfit
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(14,0)
## Mean Model   : ARFIMA(1,0,2)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##          Estimate  Std. Error   t value Pr(>|t|)
## mu       0.000425    0.000114  3.713810 0.000204
## ar1      0.656990    0.101935  6.445177 0.000000
## ma1     -0.739345    0.103789 -7.123528 0.000000
## ma2     -0.003027    0.024942 -0.121367 0.903400
## omega    0.000017    0.000003  6.640732 0.000000
## alpha1   0.000023    0.015499  0.001465 0.998831
## alpha2   0.123933    0.025279  4.902518 0.000001
## alpha3   0.090752    0.023290  3.896670 0.000098
## alpha4   0.093483    0.024970  3.743850 0.000181
## alpha5   0.103839    0.024722  4.200249 0.000027
## alpha6   0.060895    0.022867  2.662940 0.007746
## alpha7   0.068826    0.023619  2.914056 0.003568
## alpha8   0.097648    0.025755  3.791498 0.000150
## alpha9   0.052169    0.021143  2.467449 0.013608
## alpha10  0.071828    0.024184  2.970002 0.002978
## alpha11  0.071156    0.022838  3.115677 0.001835
## alpha12  0.022189    0.020910  1.061204 0.288597
## alpha13  0.036222    0.019541  1.853596 0.063797
## alpha14  0.023243    0.019648  1.182939 0.236833
## skew     0.871525    0.021154 41.198389 0.000000
## shape    8.668793    1.309150  6.621696 0.000000
##
## Robust Standard Errors:
##          Estimate  Std. Error   t value Pr(>|t|)
## mu       0.000425    0.000128  3.308307 0.000939
## ar1      0.656990    0.093392  7.034718 0.000000
## ma1     -0.739345    0.096819 -7.636367 0.000000
## ma2     -0.003027    0.023586 -0.128341 0.897879
## omega    0.000017    0.000002  7.746091 0.000000
## alpha1   0.000023    0.020205  0.001124 0.999103
## alpha2   0.123933    0.029175  4.247933 0.000022
## alpha3   0.090752    0.021874  4.148918 0.000033
## alpha4   0.093483    0.025748  3.630692 0.000283
## alpha5   0.103839    0.023924  4.340358 0.000014
## alpha6   0.060895    0.023195  2.625364 0.008656
## alpha7   0.068826    0.024628  2.794612 0.005196
## alpha8   0.097648    0.027534  3.546407 0.000391
## alpha9   0.052169    0.021158  2.465621 0.013678
## alpha10  0.071828    0.025107  2.860923 0.004224
```

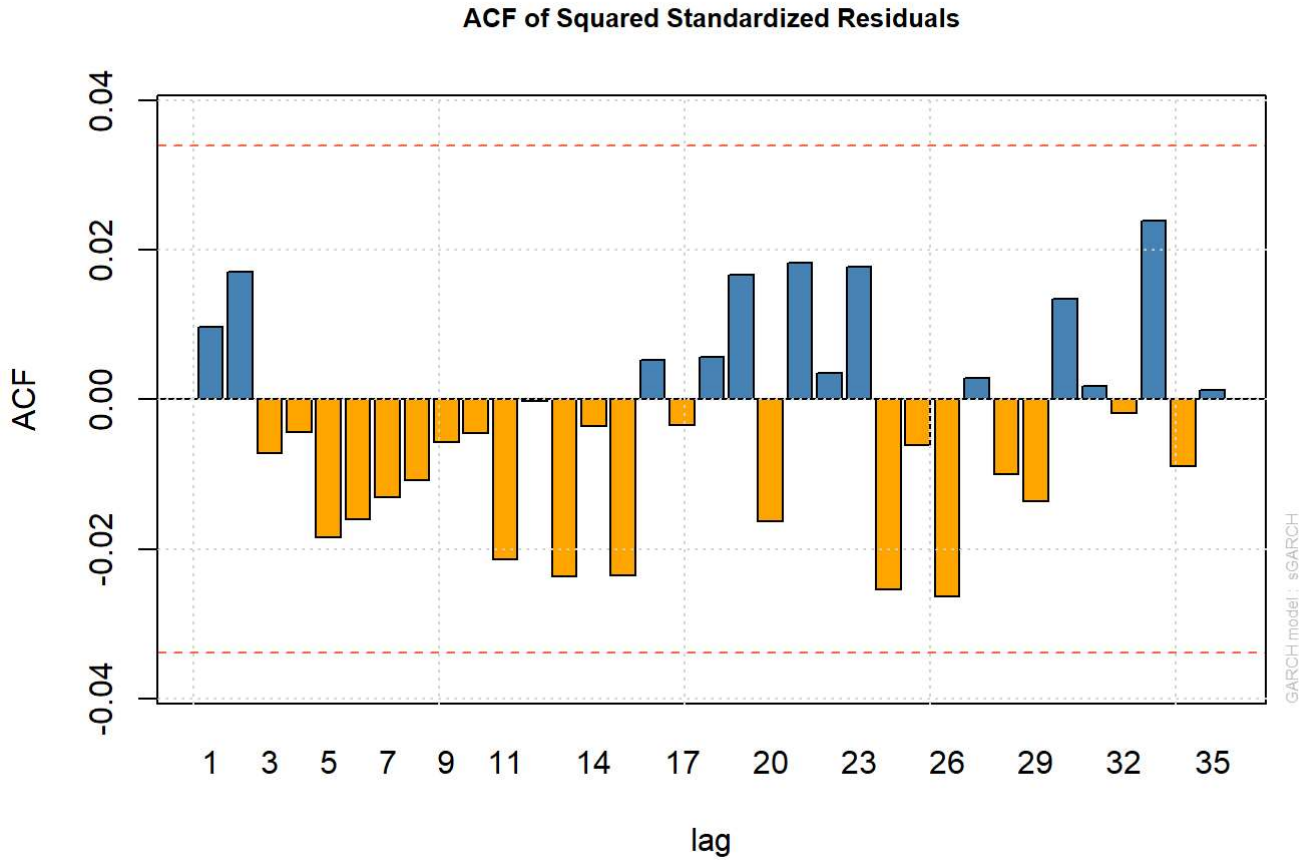
```
## alpha11  0.071156    0.022872  3.111062 0.001864
## alpha12  0.022189    0.023039  0.963139 0.335478
## alpha13  0.036222    0.018054  2.006301 0.044824
## alpha14  0.023243    0.021946  1.059080 0.289563
## skew     0.871525    0.020938 41.624852 0.000000
## shape    8.668793    1.290188  6.719015 0.000000
##
## LogLikelihood : 10562.41
##
## Information Criteria
## ------------------------------------
##
## Akaike       -6.2859
## Bayes        -6.2476
## Shibata      -6.2860
## Hannan-Quinn -6.2722
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic   p-value
## Lag[1]                      2.463 1.165e-01
## Lag[2*(p+q)+(p+q)-1][8]     9.186 1.037e-09
## Lag[4*(p+q)+(p+q)-1][14]   12.918 1.267e-02
## d.o.f=3
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.3114  0.5768
## Lag[2*(p+q)+(p+q)-1][41]  13.3106  0.9434
## Lag[4*(p+q)+(p+q)-1][69]  23.4210  0.9682
## d.o.f=14
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[15]     1.852 0.500 2.000  0.1735
## ARCH Lag[17]     1.975 1.496 1.887  0.5519
## ARCH Lag[19]     2.796 2.483 1.802  0.6796
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  no.parameters>20 (not available)
## Individual Statistics:
## mu      0.72678
## ar1     0.06460
## ma1     0.07812
## ma2     0.07547
## omega   0.39797
## alpha1  0.39202
## alpha2  0.33022
## alpha3  0.22682
## alpha4  0.06451
```

```
## alpha5  0.30806
## alpha6  0.06746
## alpha7  0.19640
## alpha8  0.42664
## alpha9  0.06595
## alpha10 0.06631
## alpha11 0.22675
## alpha12 0.10274
## alpha13 0.13737
## alpha14 0.09817
## skew    0.29464
## shape   0.48941
##
## Asymptotic Critical Values (10% 5% 1%)
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## -----------------------------------
##                    t-value      prob sig
## Sign Bias           1.9478 5.153e-02   *
## Negative Sign Bias  0.6558 5.120e-01
## Positive Sign Bias  2.7006 6.957e-03 ***
## Joint Effect       30.7001 9.831e-07 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     45.61    0.0005633
## 2    30     57.23    0.0013465
## 3    40     59.74    0.0178812
## 4    50     69.32    0.0295527
##
##
## Elapsed time : 5.391899
```

```
plot(modelfit, which = 10)
```




```
plot(modelfit, which = 11)
```



From the plot of the standardized residuals' ACF and PACF, we can see that all of the dynamics have been wiped out, which means that the model is good for fitting the data.

Alternatively, we can try a GARCH(2,1) model

```
model1=ugarchspec(
variance.model = list(model = "sGARCH", garchOrder = c(2, 1)),
mean.model = list(armaOrder = c(1, 2), include.mean = TRUE),
distribution.model = "sstd")

#Sanity check: explore the model parameters
model1@model$pars
```

```
##          Level Fixed Include Estimate LB UB
## mu           0     0       1        1 NA NA
## ar1          0     0       1        1 NA NA
## ma1          0     0       1        1 NA NA
## ma2          0     0       1        1 NA NA
## arfima       0     0       0        0 NA NA
## archm        0     0       0        0 NA NA
## mxreg        0     0       0        0 NA NA
## omega        0     0       1        1 NA NA
## alpha1       0     0       1        1 NA NA
## alpha2       0     0       1        1 NA NA
## beta1        0     0       1        1 NA NA
## gamma        0     0       0        0 NA NA
## eta1         0     0       0        0 NA NA
## eta2         0     0       0        0 NA NA
## delta        0     0       0        0 NA NA
## lambda       0     0       0        0 NA NA
## vxreg        0     0       0        0 NA NA
## skew         0     0       1        1 NA NA
## shape        0     0       1        1 NA NA
## ghlambda     0     0       0        0 NA NA
## xi           0     0       0        0 NA NA
```

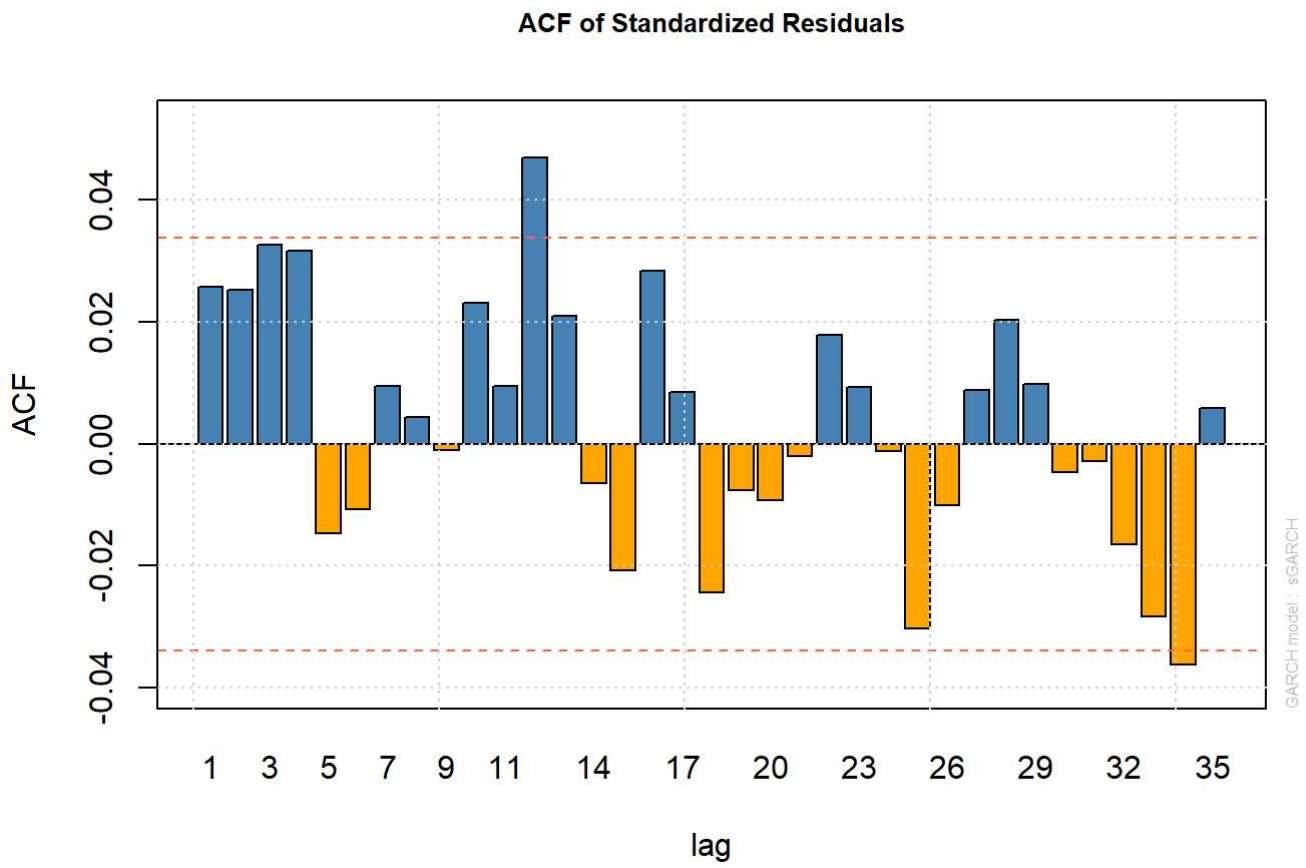
```
# Fit the model to the data
modelfit1=ugarchfit(spec=model1,data=close_returns_ts)
modelfit1
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(2,1)
## Mean Model   : ARFIMA(1,0,2)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.000415    0.000104  3.975991 0.000070
## ar1     0.666493    0.085211  7.821649 0.000000
## ma1    -0.751232    0.084971 -8.841068 0.000000
## ma2    -0.001071    0.023781 -0.045028 0.964085
## omega   0.000002    0.000005  0.345797 0.729495
## alpha1  0.000000    0.015087  0.000004 0.999997
## alpha2  0.114254    0.075671  1.509875 0.131075
## beta1   0.876439    0.074541 11.757883 0.000000
## skew    0.868241    0.015416 56.321103 0.000000
## shape   8.646566    2.167533  3.989128 0.000066
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.000415    0.000654  0.634042 0.526053
## ar1     0.666493    0.738742  0.902200 0.366951
## ma1    -0.751232    0.795377 -0.944498 0.344915
## ma2    -0.001071    0.126475 -0.008467 0.993245
## omega   0.000002    0.000068  0.024605 0.980370
## alpha1  0.000000    0.055717  0.000001 0.999999
## alpha2  0.114254    1.098267  0.104031 0.917145
## beta1   0.876439    1.056892  0.829261 0.406957
## skew    0.868241    0.202476  4.288123 0.000018
## shape   8.646566   35.411835  0.244172 0.807098
##
## LogLikelihood : 10563.09
##
## Information Criteria
## ------------------------------------
##
## Akaike       -6.2928
## Bayes        -6.2746
## Shibata      -6.2929
## Hannan-Quinn -6.2863
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic   p-value
## Lag[1]                       2.237 1.347e-01
## Lag[2*(p+q)+(p+q)-1][8]      9.486 1.512e-10
```

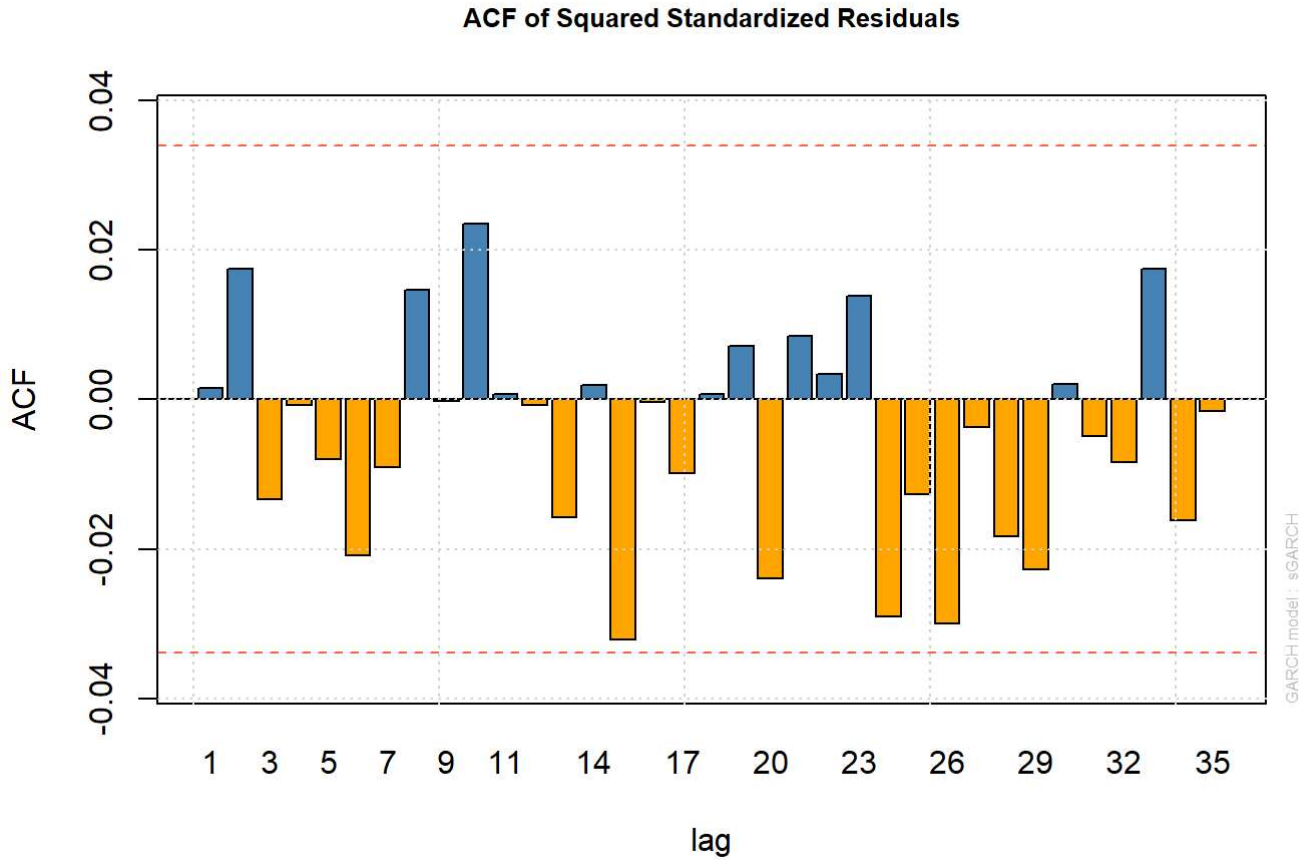
```
## Lag[4*(p+q)+(p+q)-1][14]    13.448 7.942e-03
## d.o.f=3
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                    0.007389  0.9315
## Lag[2*(p+q)+(p+q)-1][8]   2.164297  0.8329
## Lag[4*(p+q)+(p+q)-1][14]  3.863433  0.8929
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[4]  0.001639 0.500 2.000  0.9677
## ARCH Lag[6]  1.153584 1.461 1.711  0.7041
## ARCH Lag[8]  1.815845 2.368 1.583  0.7785
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  97.2087
## Individual Statistics:
## mu      0.74768
## ar1     0.08013
## ma1     0.08969
## ma2     0.07869
## omega  20.71622
## alpha1  0.20460
## alpha2  0.18219
## beta1   0.17402
## skew    0.37676
## shape   0.50164
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.29 2.54 3.05
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                     t-value      prob sig
## Sign Bias           1.8561 6.353e-02   *
## Negative Sign Bias  0.5861 5.579e-01
## Positive Sign Bias  2.7640 5.741e-03 ***
## Joint Effect       29.7542 1.555e-06 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     44.02    0.0009383
## 2    30     53.35    0.0038376
## 3    40     55.11    0.0451717
## 4    50     67.53    0.0407127
```

```
##
##
## Elapsed time : 1.860773
```

```
plot(modelfit1, which = 10)
```




```
plot(modelfit1, which = 11)
```



We see that the model fits the data equally well looking at how the standardized residuals' ACF and PACF have almost no spikes. This means that a GARCH model is better than just an ARCH model considering GARCH models require less parameters than ARCH models.

Problem 3

1- and 2-step ahead forecasts of the volatility

```
modelfor = ugarchforecast(modelfit1, data = NULL, n.ahead = 2, n.roll = 0, out.sample = 0)
print("Volatility forecast")
```

```
## [1] "Volatility forecast"
```

```
modelfor@forecast$sigmaFor
```

```
##     1979-03-08 16:00:00
## T+1         0.008749874
## T+2         0.008347120
```

```
low95 = modelfor@forecast$seriesFor - 1.96*modelfor@forecast$sigmaFor
high95 = modelfor@forecast$seriesFor + 1.96*modelfor@forecast$sigmaFor
forecasted_values = data.frame(low95, modelfor@forecast$seriesFor ,high95)
colnames(forecasted_values) = c("Low95", "Forecasted value", "High95")
print("Returns forecast")
```

```
## [1] "Returns forecast"
```

```
forecasted_values
```

```
##           Low95 Forecasted value     High95
## T+1 -0.01785895    -0.0007091943 0.01644056
## T+2 -0.01669769    -0.0003373368 0.01602302
```

Problem 4

```
#clear previous environment and obtain NYSE closing price
library(quantmod)
```

```
## Loading required package: xts
```

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
##
## Attaching package: 'quantmod'
```

```
## The following object is masked from 'package:tis':
##
##     Lag
```

```
rm(list = ls(all=T))
getSymbols("NYA")
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).
```

```
## Warning: NYA contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## [1] "NYA"
```

a. Get the AR model of the data

```
NYSE_returns_ts = diff(log(NYA$NYA.Close[complete.cases(NYA$NYA.Close)]))
NYSE_returns_ts = NYSE_returns_ts[complete.cases(NYSE_returns_ts)]
armodel = ar(NYSE_returns_ts)
summary(armodel)
```

```
##              Length Class  Mode
## order           1   -none- numeric
## ar             34   -none- numeric
## var.pred        1   -none- numeric
## x.mean          1   -none- numeric
## aic            35   -none- numeric
## n.used          1   -none- numeric
## order.max       1   -none- numeric
## partialacf     34   -none- numeric
## resid        2807   -none- numeric
## method          1   -none- character
## series          1   -none- character
## frequency       1   -none- numeric
## call            2   -none- call
## asy.var.coef 1156   -none- numeric
```

```
armodel
```

```
##
## Call:
## ar(x = NYSE_returns_ts)
##
## Coefficients:
##       1        2        3        4        5        6        7        8
## -0.0837  -0.0454   0.0124  -0.0255  -0.0542   0.0041  -0.0151   0.0244
##       9       10       11       12       13       14       15       16
## -0.0133   0.0217  -0.0111   0.0284   0.0084  -0.0384  -0.0347   0.0581
##      17       18       19       20       21       22       23       24
##  0.0151  -0.0542  -0.0004   0.0493  -0.0264   0.0220  -0.0109  -0.0126
##      25       26       27       28       29       30       31       32
##  0.0244   0.0080   0.0336  -0.0264   0.0110   0.0288   0.0294  -0.0073
##      33       34
## -0.0094  -0.0651
##
## Order selected 34  sigma^2 estimated as  0.0001733
```

From here we get that the order of an AR model fitting the data is 34

Now we will fit a GARCH(1,1) model

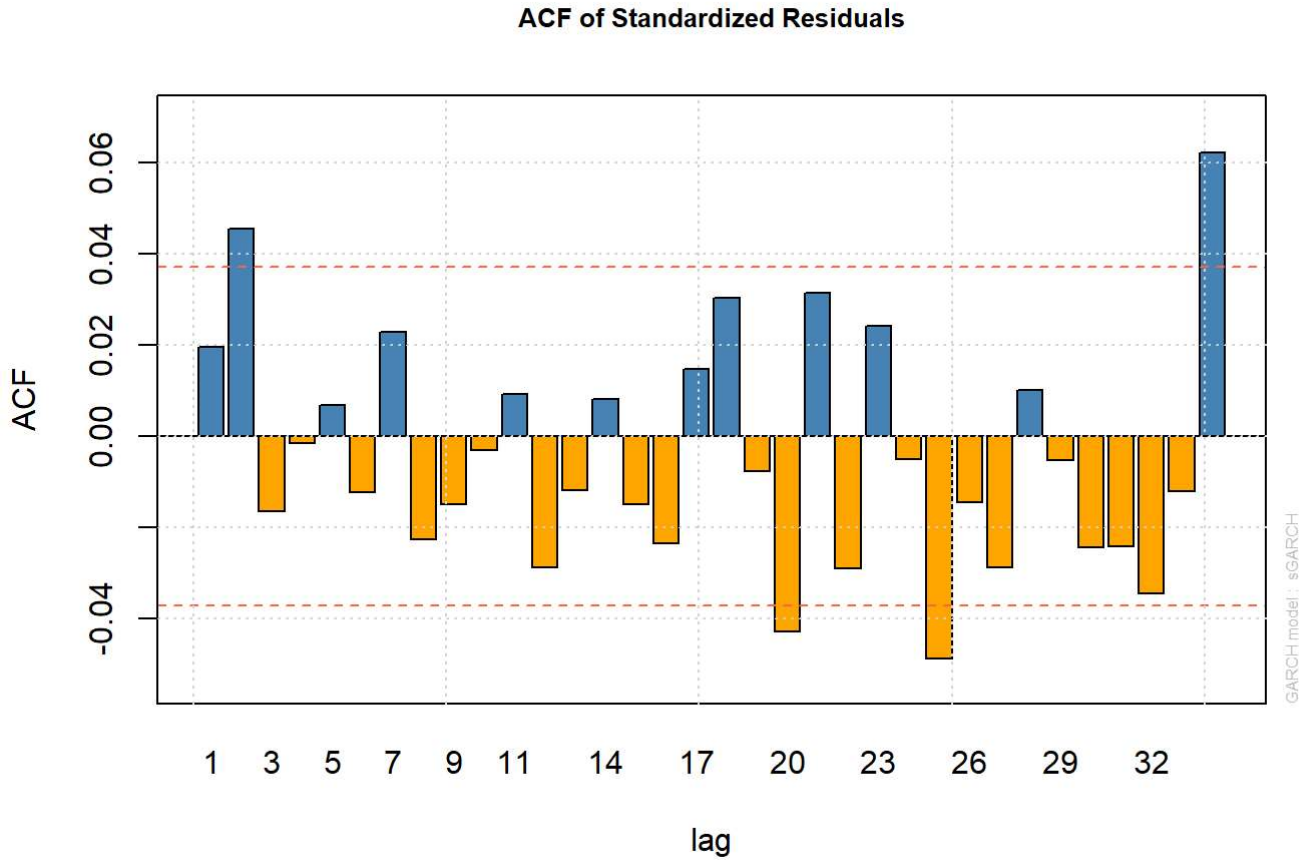
```
model=ugarchspec(
variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
distribution.model = "sstd")

# Fit the model to the data
modelfit=ugarchfit(spec=model,data=residuals(armodel)[complete.cases(residuals(armodel))])
modelfit
```

```
##
## *----------------------------------*
## *          GARCH Model Fit         *
## *----------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000358    0.000148  2.42444 0.015332
## omega   0.000001    0.000001  0.65709 0.511122
## alpha1  0.106288    0.024441  4.34877 0.000014
## beta1   0.892683    0.021685 41.16637 0.000000
## skew    0.839067    0.020209 41.51893 0.000000
## shape   6.426354    0.429179 14.97359 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000358    0.000193  1.85877  0.06306
## omega   0.000001    0.000009  0.09879  0.92131
## alpha1  0.106288    0.183437  0.57943  0.56230
## beta1   0.892683    0.160704  5.55485  0.00000
## skew    0.839067    0.063243 13.26727  0.00000
## shape   6.426354    4.545411  1.41381  0.15742
##
## LogLikelihood : 8932.253
##
## Information Criteria
## ------------------------------------
##
## Akaike       -6.4380
## Bayes        -6.4251
## Shibata      -6.4380
## Hannan-Quinn -6.4333
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      1.072 0.30058
## Lag[2*(p+q)+(p+q)-1][2]     3.948 0.07786
## Lag[4*(p+q)+(p+q)-1][5]     6.160 0.08245
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                  7.109e-05 0.99327
```

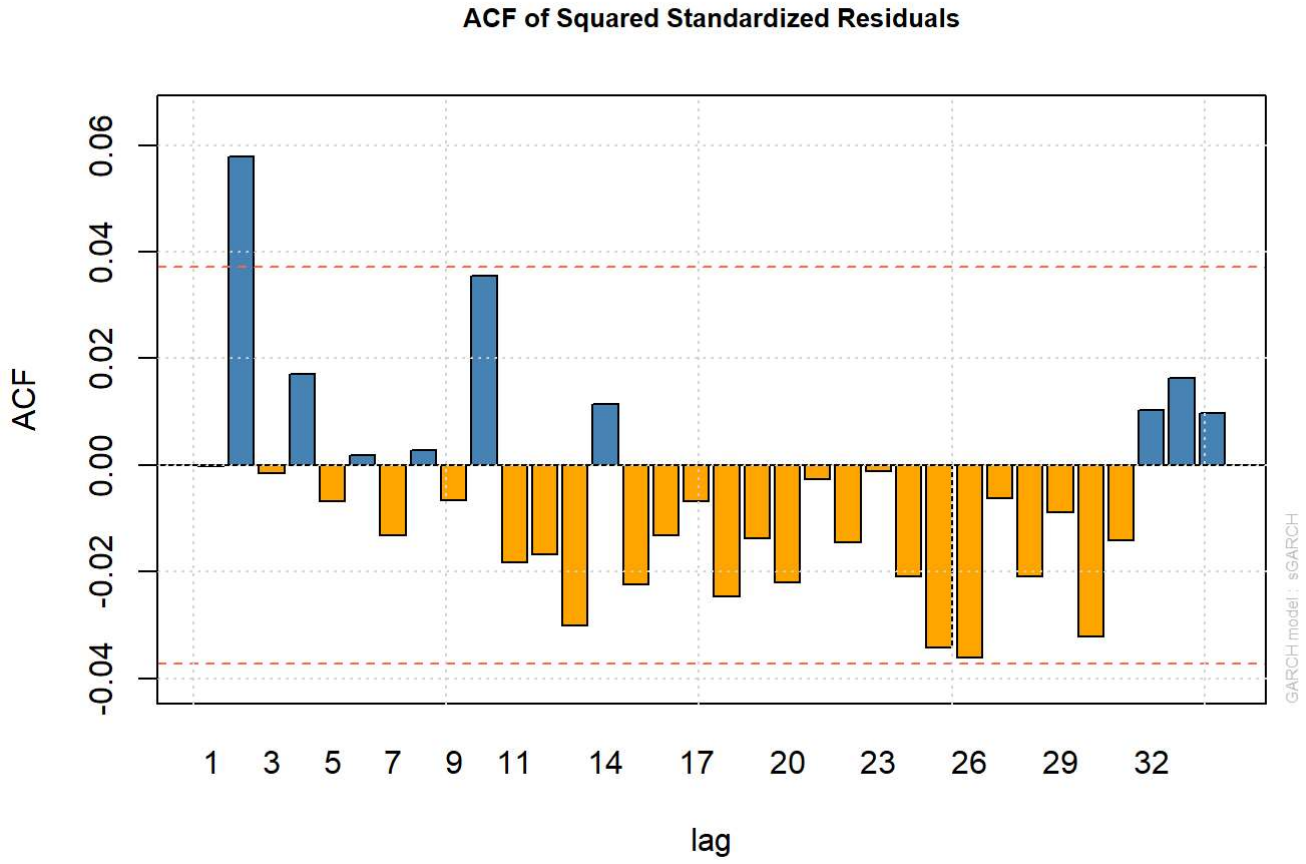
```
## Lag[2*(p+q)+(p+q)-1][5] 7.766e+00 0.03368
## Lag[4*(p+q)+(p+q)-1][9] 9.033e+00 0.08002
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]  0.006651 0.500 2.000  0.9350
## ARCH Lag[5]  0.728104 1.440 1.667  0.8149
## ARCH Lag[7]  0.997488 2.315 1.543  0.9140
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  78.6101
## Individual Statistics:
## mu      0.04776
## omega  16.87823
## alpha1  0.64478
## beta1   0.89372
## skew    0.12158
## shape   0.51319
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.49 1.68 2.12
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value      prob sig
## Sign Bias            3.115 1.859e-03 ***
## Negative Sign Bias   1.662 9.672e-02   *
## Positive Sign Bias   1.655 9.813e-02   *
## Joint Effect        22.413 5.351e-05 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     37.32     0.007233
## 2    30     49.11     0.011244
## 3    40     53.07     0.065902
## 4    50     67.70     0.039518
##
##
## Elapsed time : 0.5564549
```

Check the ACF and PACF of the returns and the squared standardized residuals

```
plot(modelfit, which = 10)
```




```
plot(modelfit, which = 11)
```



We can see that almost all the dynamics of the data has been wiped out and we also took care of the heteroskedasticity of the errors.

b)fitting both the AR model and GARCH together(but since the AR model takes too many orders, we use an ARMA model instead). After using auto.arima, we find that the optimal ARMA model is 1,1 respectively.

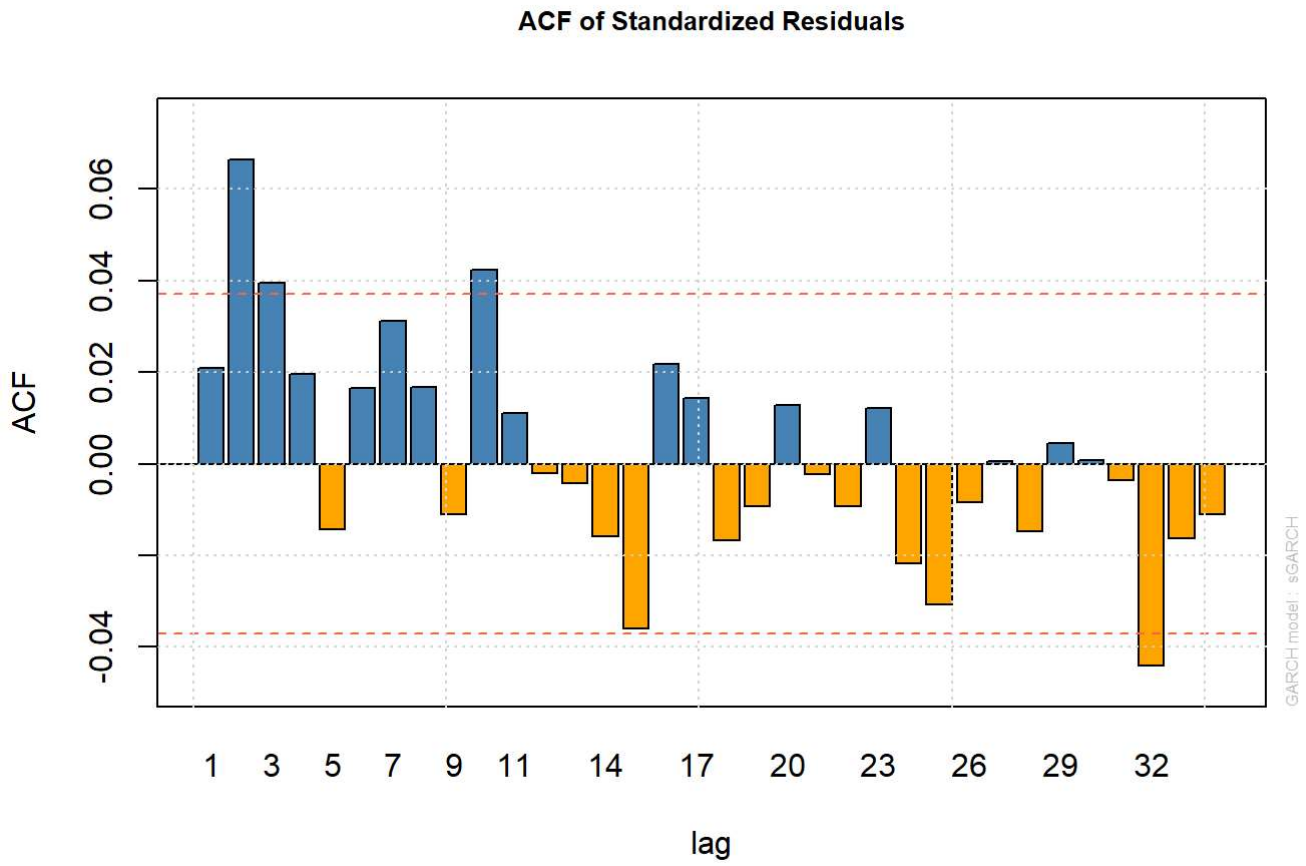
```
model=ugarchspec(
variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
mean.model = list(armaOrder = c(1, 1), include.mean = TRUE),
distribution.model = "sstd")

# Fit the model to the data
modelfit=ugarchfit(spec=model,data= NYSE_returns_ts)
modelfit
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(1,0,1)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.000475    0.000098   4.85915 0.000001
## ar1     0.769525    0.052218  14.73664 0.000000
## ma1    -0.846305    0.042480 -19.92224 0.000000
## omega   0.000001    0.000001   0.74458 0.456524
## alpha1  0.109846    0.023528   4.66864 0.000003
## beta1   0.889098    0.020807  42.73013 0.000000
## skew    0.812318    0.019223  42.25721 0.000000
## shape   6.235060    0.410125  15.20282 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.000475    0.000165   2.88217 0.003949
## ar1     0.769525    0.056919  13.51971 0.000000
## ma1    -0.846305    0.059637 -14.19089 0.000000
## omega   0.000001    0.000008   0.11886 0.905383
## alpha1  0.109846    0.170155   0.64556 0.518562
## beta1   0.889098    0.147927   6.01036 0.000000
## skew    0.812318    0.096252   8.43948 0.000000
## shape   6.235060    3.974363   1.56882 0.116690
##
## LogLikelihood : 9089.121
##
## Information Criteria
## ------------------------------------
##
## Akaike       -6.4703
## Bayes        -6.4534
## Shibata      -6.4704
## Hannan-Quinn -6.4642
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic   p-value
## Lag[1]                      1.238 2.658e-01
## Lag[2*(p+q)+(p+q)-1][5]    14.346 0.000e+00
## Lag[4*(p+q)+(p+q)-1][9]    18.187 7.591e-07
## d.o.f=2
## H0 : No serial correlation
##
```

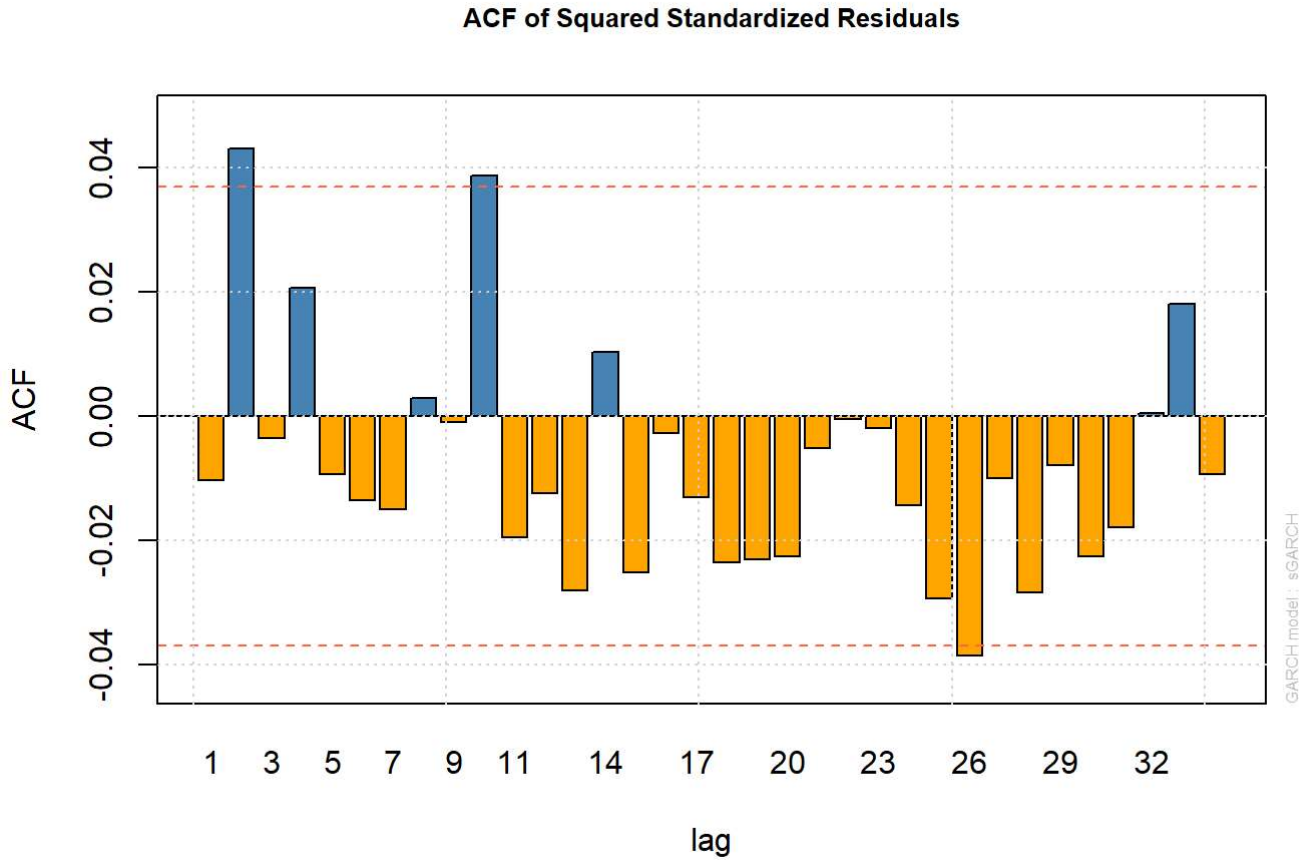
```
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.3031  0.5819
## Lag[2*(p+q)+(p+q)-1][5]    5.0133  0.1520
## Lag[4*(p+q)+(p+q)-1][9]    6.3345  0.2616
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.03669 0.500 2.000  0.8481
## ARCH Lag[5]   1.13660 1.440 1.667  0.6929
## ARCH Lag[7]   1.79758 2.315 1.543  0.7601
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  80.4474
## Individual Statistics:
## mu      0.08615
## ar1     0.01309
## ma1     0.01782
## omega  16.55317
## alpha1  0.63467
## beta1   0.91328
## skew    0.14081
## shape   0.32431
##
## 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            3.386 7.181e-04 ***
## Negative Sign Bias   2.012 4.434e-02  **
## Positive Sign Bias   1.696 8.994e-02   *
## Joint Effect        25.508 1.209e-05 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     57.35    1.009e-05
## 2    30     56.15    1.817e-03
## 3    40     76.35    3.247e-04
## 4    50     82.83    1.800e-03
##
##
## Elapsed time : 0.62885
```

Plot the ACF of the residuals and squared residuals

```
plot(modelfit, which = 10)
```




```
plot(modelfit, which = 11)
```

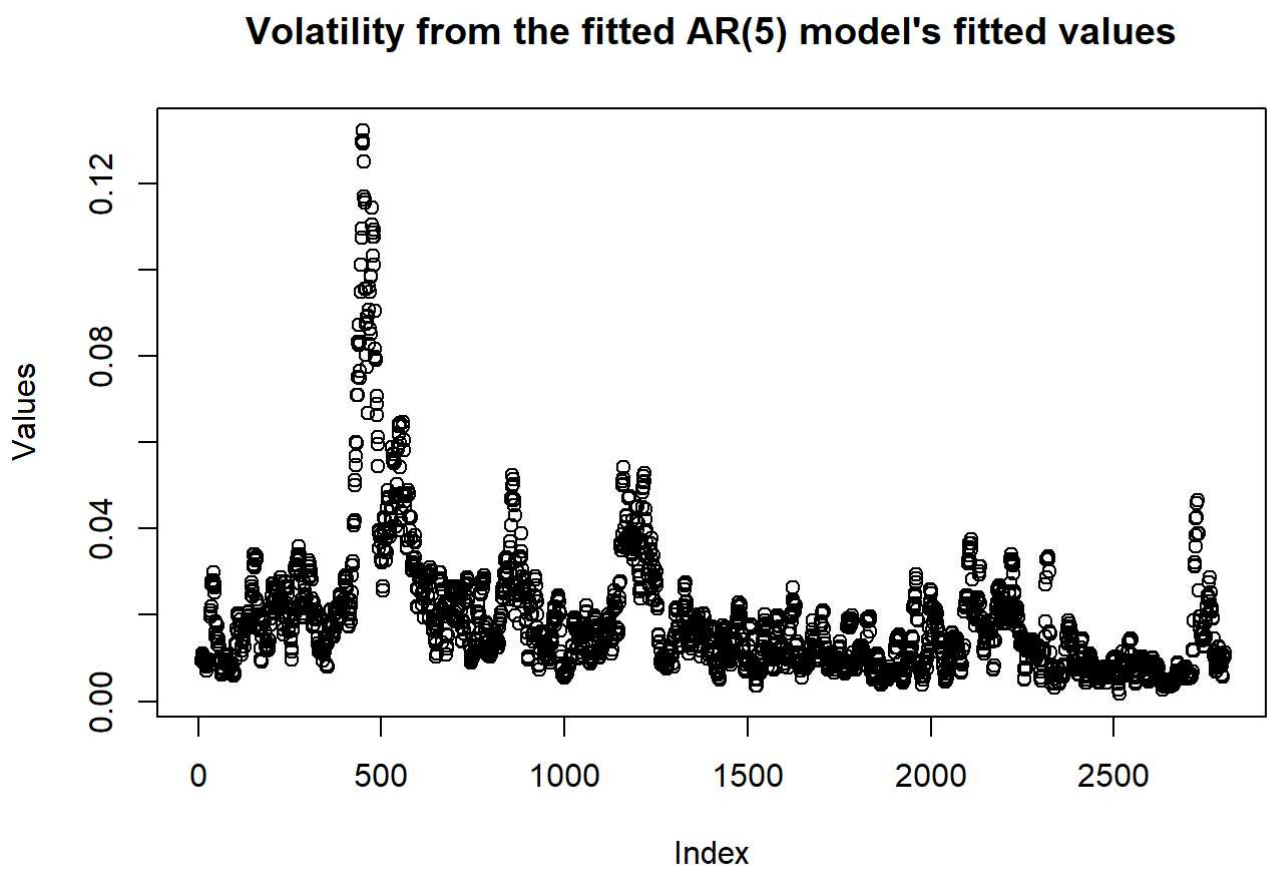


We can see again that the dynamics are almost wiped off since there are not many spikes in the residuals.

Problem 5

Note: we don't want to clear previous environment since we still want to work on the same data sets. a)

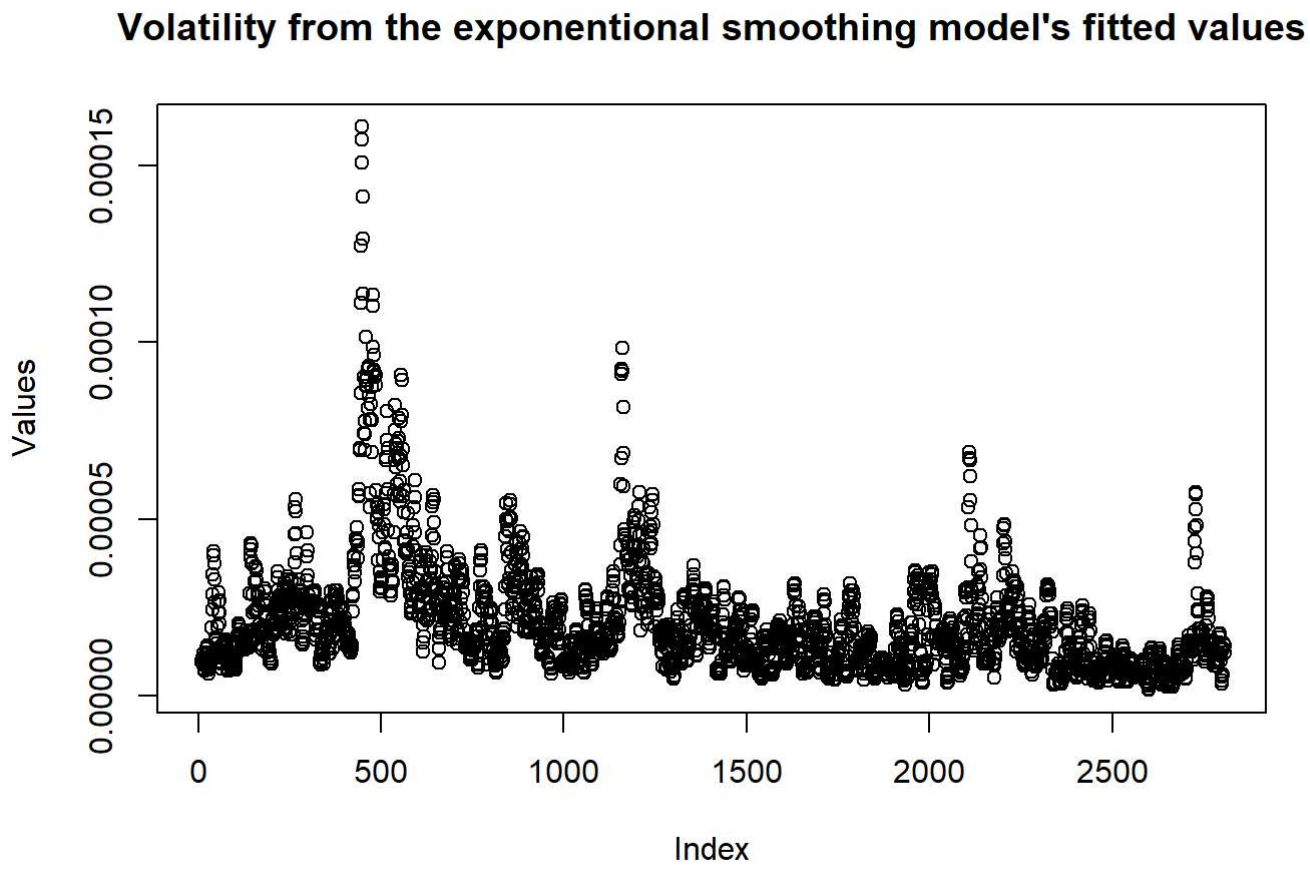
```
#Fit an AR(5) model
ar5 = ar(NYSE_returns_ts, order.max = 5, aic = FALSE)
# Standard deviation of the spread

plot(sqrt(252) * runSD(fitted(ar5)[complete.cases(fitted(ar5))], 10), main = "Volatility from th
e fitted AR(5) model's fitted values", ylab = "Values")
```



As we can see, the standard deviation spikes in the past periods and slowly stabilizes as time pases. This means that we still have not taken into account the heteroskedasticity of the errors.

b. Fitting an exponential smoothing model to the data

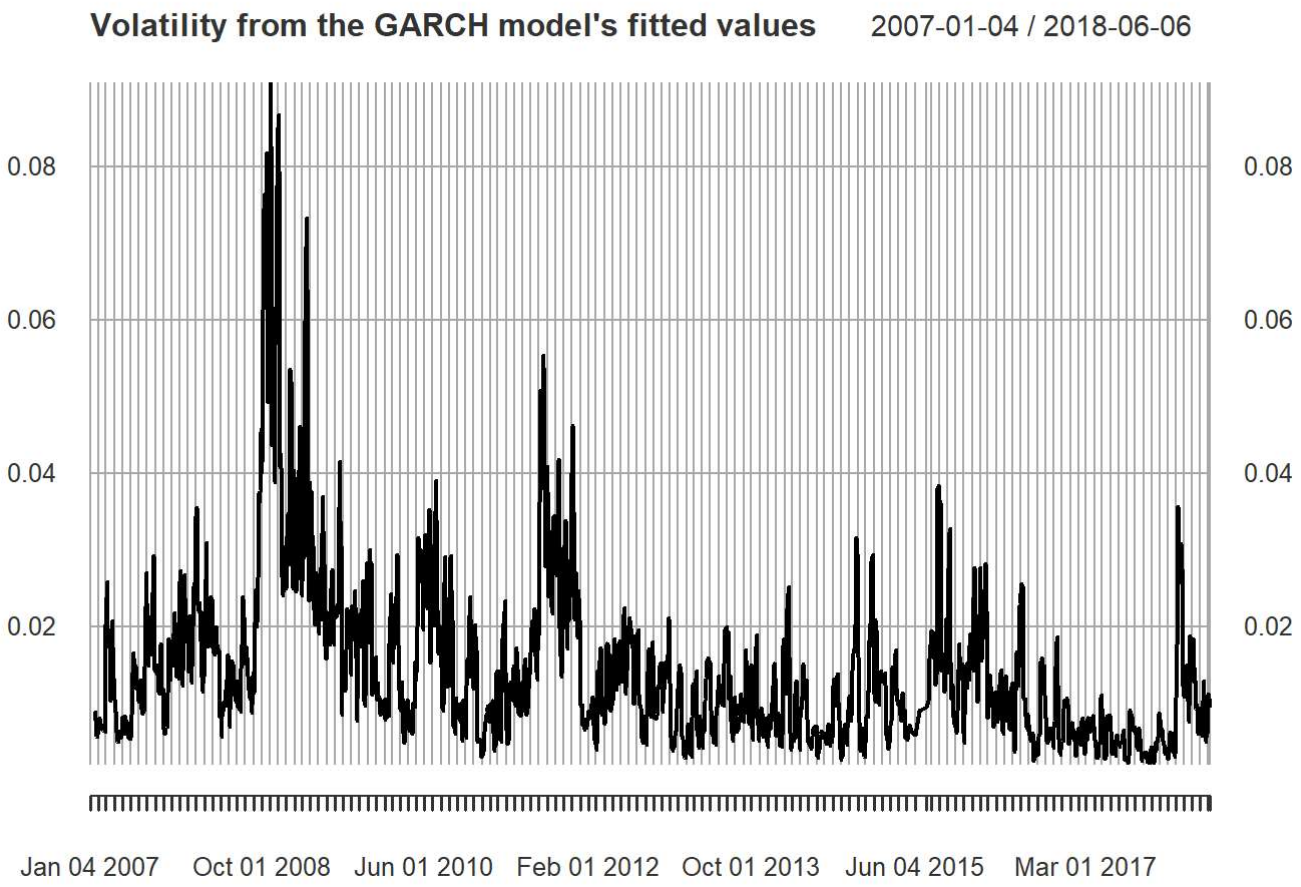
```
esModel = ets(NYSE_returns_ts)

plot(sqrt(252) * runSD(fitted(esModel)[complete.cases(fitted(esModel))], 10), main = "Volatility
 from the exponentional smoothing model's fitted values", ylab = "Values")
```



The volatility plot looks similar as in (a), i.e the errors are heteroskedastic.

c. Fitting a GARCH model that we used in problem 4

```
plot(sqrt(252) * runSD(fitted(modelfit)[complete.cases(fitted(modelfit))], 10), main = "Volatili
ty from the GARCH model's fitted values", ylab = "Values")
```



Interestingly enough, the volatitility plot of the ARMA+GARCH model would yield the same shape.

d. Even though the shape of the volatility series is similar for all 3, the values are not the same. The GARCH model is more preferable, since the GARCH model takes into account that the variance might be varying, which would create a more accurate picture of the volatility plot, unlike the AR(5) and exponential smoothing models.

Problem 6

a. The model is ARCH(1): y_t = 76 + eps_t sigsqr^2 = 3 + 0.6 eps_(t-1)^2

```
rm(list = ls(all=T))
#if y_t = 92
eps_t = 92-76
sigsqr1 = 3 + 0.6*(eps_t)^2
sigsqr = vector()
sigsqr[1] = sigsqr1
sigsqr[2] = 3 + 0.6*sigsqr[1]
alpha = 0.6
for (i in 3:100){
  sigsqr[i] = 0.6^(i-1)*sigsqr[i-1]
  for (j in 0:i-2){
    sigsqr[i] = sigsqr[i] + alpha^j
  }
}
print("The forecast of the conditional variance for the next 10 days")
```

```
## [1] "The forecast of the conditional variance for the next 10 days"
```

```
sigsqr[1:10]
```

```
##  [1] 156.600000  96.960000  40.950044  15.249654   8.596800   7.418532
##  [7]   7.173923   7.075284   7.021292   6.990009
```

b. This question requires us to find the 90% confidence interval for the value. So [72,80] = 76 +/- 1.645*sigma

```
sig = (80-76)/1.645
print("The required conditional variance")
```

```
## [1] "The required conditional variance"
```

```
sig
```

```
## [1] 2.431611
```

```
print ("The forecast of the conditional variance for the next 100 days")
```

```
## [1] "The forecast of the conditional variance for the next 100 days"
```

```
sigsqr
```

```
##   [1] 156.600000  96.960000  40.950044  15.249654   8.596800   7.418532
##   [7]   7.173923   7.075284   7.021292   6.990009   6.971594   6.960667
##  [13]   6.954154   6.950262   6.947932   6.946536   6.945699   6.945197
##  [19]   6.944896   6.944715   6.944607   6.944542   6.944503   6.944480
##  [25]   6.944466   6.944457   6.944452   6.944449   6.944447   6.944446
##  [31]   6.944445   6.944445   6.944445   6.944445   6.944445   6.944445
##  [37]   6.944444   6.944444   6.944444   6.944444   6.944444   6.944444
##  [43]   6.944444   6.944444   6.944444   6.944444   6.944444   6.944444
##  [49]   6.944444   6.944444   6.944444   6.944444   6.944444   6.944444
##  [55]   6.944444   6.944444   6.944444   6.944444   6.944444   6.944444
##  [61]   6.944444   6.944444   6.944444   6.944444   6.944444   6.944444
##  [67]   6.944444   6.944444   6.944444   6.944444   6.944444   6.944444
##  [73]   6.944444   6.944444   6.944444   6.944444   6.944444   6.944444
##  [79]   6.944444   6.944444   6.944444   6.944444   6.944444   6.944444
##  [85]   6.944444   6.944444   6.944444   6.944444   6.944444   6.944444
##  [91]   6.944444   6.944444   6.944444   6.944444   6.944444   6.944444
##  [97]   6.944444   6.944444   6.944444   6.944444
```

We can see that the conditional variance converges to 6.944 as the period increases to infinity. This means that the conditional variance will never drop to the level so that there is a 90% probablity that the temperature will stay between 72 and 80.

c. Given our current constraint, we can't forecast a bad weather since the conditional forecast converges to a constant even after 20 days.