ISYE 6402 Homework 6 Solutions

Background

Individual stock prices tend to exhibit high amounts of non-constant variance, and thus ARIMA models built upon that data would likely exhibit non-constant variance in residuals. In this problem we are going to analyze the Intel stock price data from 2012 through end of 2021. We will use the ARIMA-GARCH to model daily and weekly stock price (adjusted close price at the end of a day for daily data or at the end of the week for weekly data), with a focus on the behavior of its volatility as well as forecasting both the price and the volatility.

##Data import and cleaning

```
## Libraries used within this homework are uploaded here
library(zoo,warn.conflicts=FALSE)
library(lubridate,warn.conflicts=FALSE)
library(mgcv,warn.conflicts=FALSE)
library(rugarch,warn.conflicts=FALSE)
```

```
#importing the data
dailydata <- read.csv("INTCDaily.csv", head = TRUE)
weeklydata <- read.csv("INTCWeekly.csv", head = TRUE)
#cleaning the data
#dates to date format
weeklydata$Date<-as.Date(weeklydata$Date,format='%m/%d/%y')
dailydata$Date<-as.Date(dailydata$Date,format='%m/%d/%y')

#prices to timeseries format
INTWeekly <- ts(weeklydata$Adj.Close,start=c(2012,1,1),freq=52)
INTDaily <- ts(dailydata$Adj.Close,start=c(2012,1,1),freq=252)
```

#Question 1: Exploratory Data Analysis (20 points)

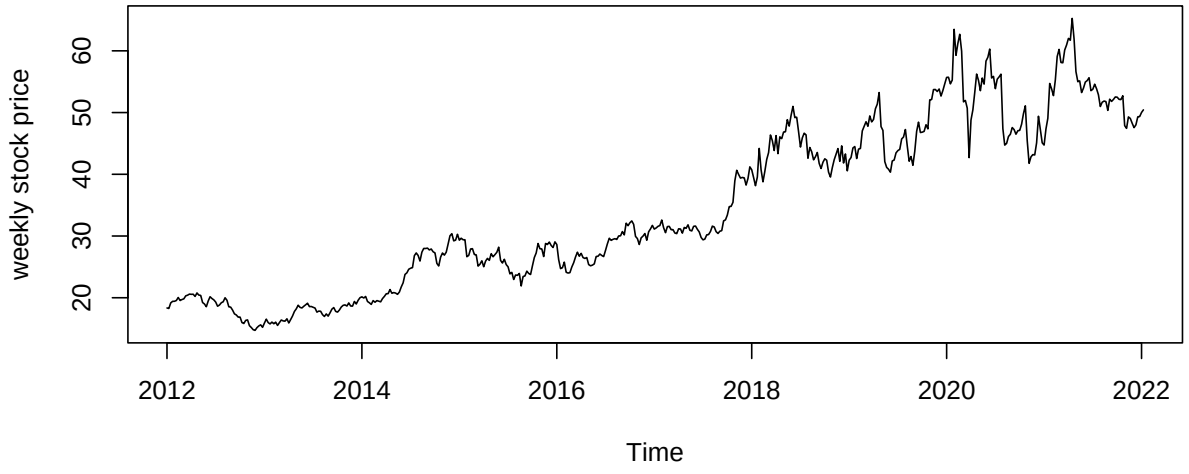
1a. Based on your intuition, when would you use daily vs weekly stock price data?

I will use weekly stock price data, since it is probably more stable and curve is more smooth with less variation.

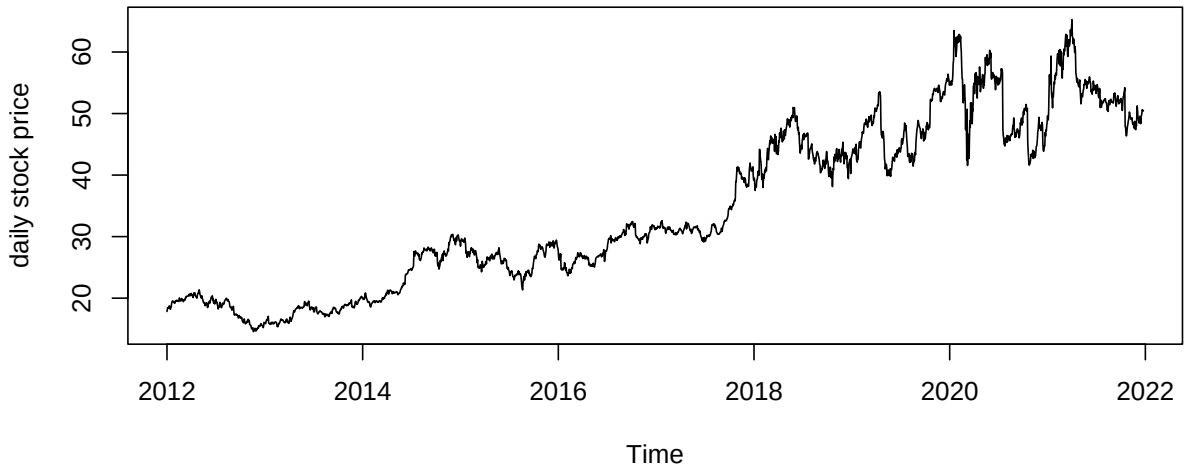
Response: Question 1a

1b. Plot the time series plots comparing daily vs weekly data. How do the daily vs weekly time series data compare?

```
ts.plot(INTWeekly,ylab="weekly stock price")
```




```
ts.plot(INTDaily,ylab="daily stock price")
```

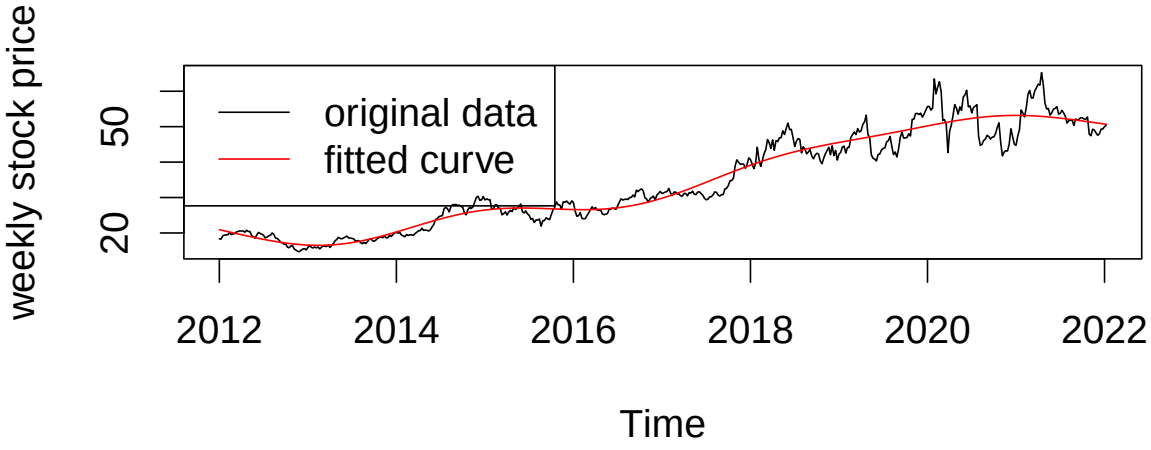


Response: Question 1b

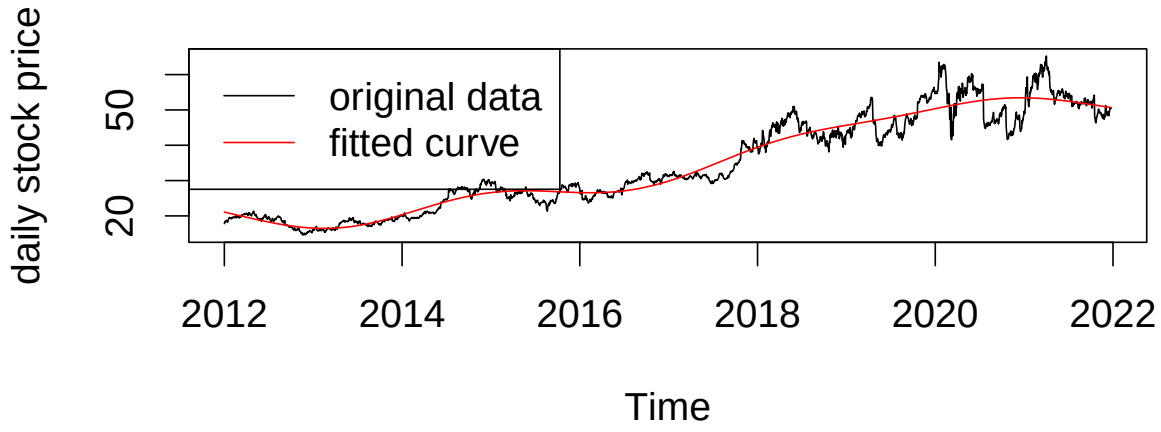
1c. Fit a non-parametric trend using splines regression to both the daily and weekly time-series data. Overlay the fitted trends. How do the trends compare?

```
time.pts = c(1:length(INTWeekly))
time.pts = c(time.pts - min(time.pts))/max(time.pts)
gam.fit.weekly = gam(INTWeekly~s(time.pts))
fitted.weekly= fitted(gam.fit.weekly)

time.pts = c(1:length(INTDaily))
```

```
time.pts = c(time.pts - min(time.pts))/max(time.pts)
gam.fit.daily = gam(INTDaily~s(time.pts))
fitted.daily= fitted(gam.fit.daily)



par(cex=1.5)

ts.plot(INTWeekly,ylab="weekly stock price")
lines(ts(fitted.weekly,start=c(2012,1,1),freq=52),col='red')
legend('topleft', legend=c("original data",'fitted curve'),lty = 1, col=c("black","red"),cex=1)
```



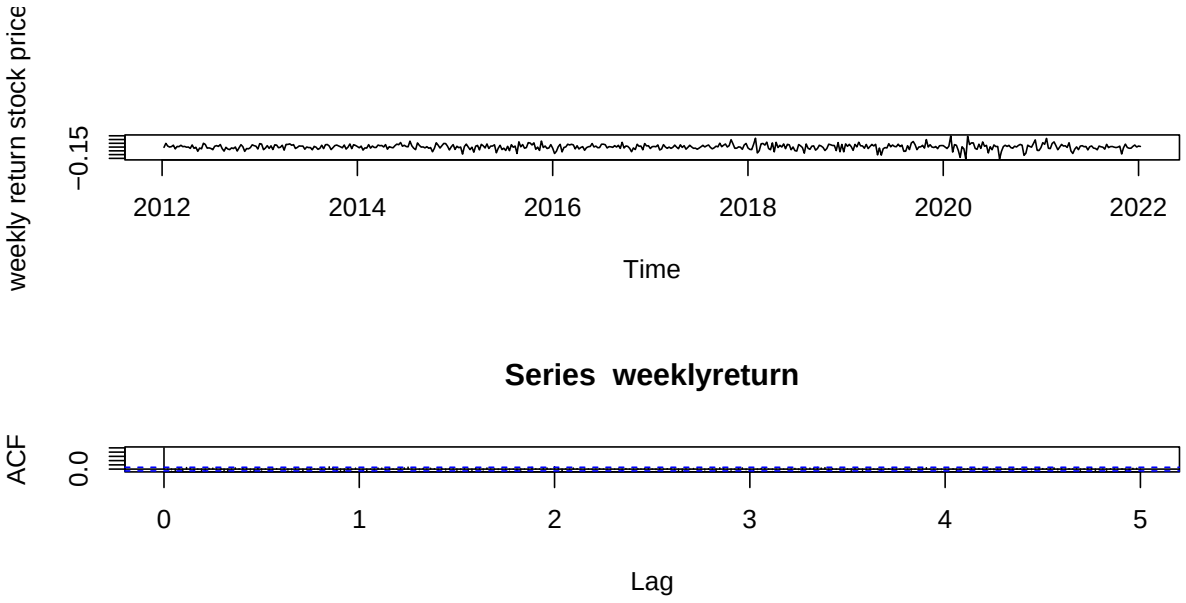

```
ts.plot(INTDaily,ylab="daily stock price")
lines(ts(fitted.daily,start=c(2012,1,1),freq=252),col='red')
legend('topleft', legend=c("original data",'fitted curve'),lty = 1, col=c("black","red"),cex=1)
```



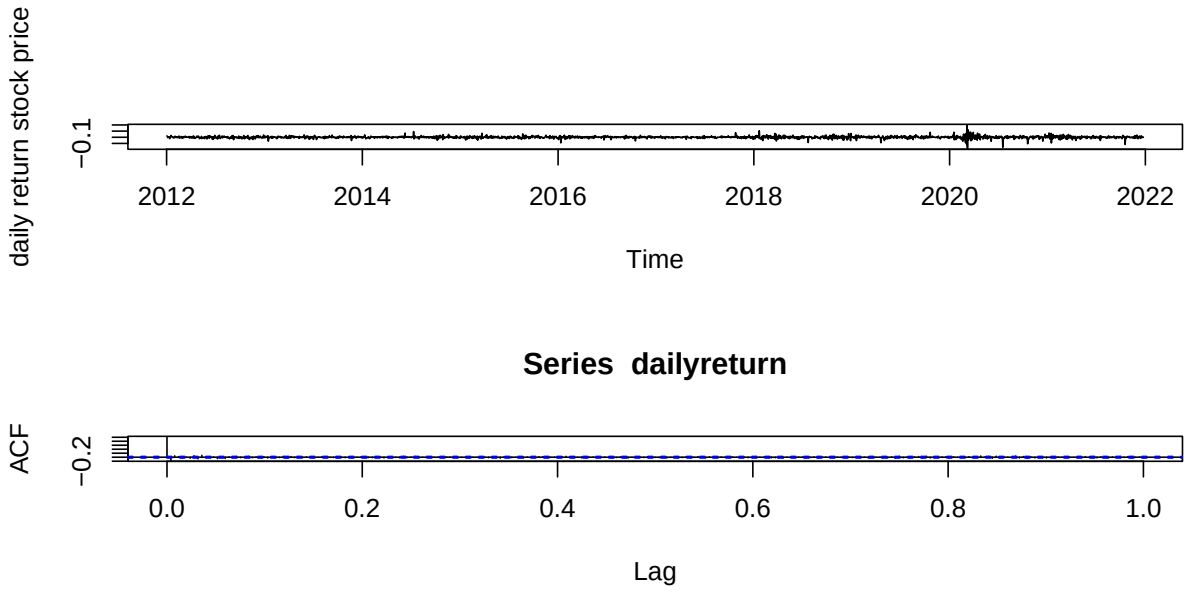
Response: Question 1c Non parametric trend was clearly captured by spline regression. They looked very similar for daily and weekly return data since weekly return price is basically the same thing as daily price with 5 lags.

1d. Consider the return stock price computed as provided in the canvas homework assignment. Apply this formula to compute the return price based on the daily and weekly time series data. Plot the return time series and their corresponding ACF plots. How do the return time series compare in terms of stationarity and serial dependence?

```
weeklyreturn=diff(INTWeekly)/ts(INTWeekly[-length(INTWeekly)],start=c(2012,2),freq=52)
par(mfrow=c(2,1))
plot(weeklyreturn,ylab="weekly return stock price")
acf(weeklyreturn,lag.max=52*5)
```



```
dailyreturn=diff(INTDaily)/ts(INTDaily[-length(INTDaily)],start=c(2012,2),freq=252)
par(mfrow=c(2,1))
plot(dailyreturn,ylab="daily return stock price")
acf(dailyreturn,lag.max=252)
```



Response: Question 1d

The peaks on ACF of weekly return dropped almost immediately after 0 and the peak of daily return also dropped quickly. The time series of return price had constant mean and a bit of strong variation in more year only. All suggested that both of them are weakly stationary

#Question 2: ARIMA(p,d,q) for Stock Price (20 Points)

2a. Divide the data into training and testing data set, where the training data exclude the last week of data (December 27th - December 30th) with the testing data including the last week of data. Apply the iterative model to fit an ARIMA(p,d,q) model with max AR and MA orders of 8 and difference orders 1 and 2 separately to the training datasets of the daily and weekly data. Display the summary of the final model fit.

```
test_modelA <- function(p,d,q,data){
    mod = arima(data, order=c(p,d,q), method="ML")
    n=length(data)
    current.aic <- AIC(mod)
    current.aic<-current.aic-2*(p+q+1)+2*(p+q+1)*n/(n-p-q-2)

    df = data.frame(p,d,q,current.aic)
    names(df) <- c("p","d","q","AIC")

    return(df)
}

orders <- data.frame(Inf,Inf,Inf,Inf)
names(orders) <- c("p","d","q","AIC")
for (p in 0:8){
```

```
    for(d in 1:2) {
        for (q in 0:8) {
            possibleError <- tryCatch(
                orders<-rbind(orders,test_modelA(p,d,q,INTWeekly[1:521])),
                error=function(e) e
            )

            if(inherits(possibleError, "error")) next
        }

    }
}

orders <- orders[order(-orders$AIC),]

tail(orders)
```

```
##     p d q      AIC
## 150 8 1 4 1908.631
## 100 5 1 8 1908.509
## 98  5 1 6 1908.256
## 115 6 1 5 1908.253
## 96  5 1 4 1908.146
## 79  4 1 5 1907.714
```

```
orders <- data.frame(Inf,Inf,Inf,Inf)
names(orders) <- c("p","d","q","AIC")
for (p in 0:8){
    for(d in 1:2) {
        for (q in 0:8) {
            possibleError <- tryCatch(
                orders<-rbind(orders,test_modelA(p,d,q,INTDaily[1:2512])),
                error=function(e) e
            )

            if(inherits(possibleError, "error")) next
        }

    }
}
print(orders)
```

```
##       p   d   q      AIC
## 1   Inf Inf Inf      Inf
## 2     0   1   0 5946.709
## 3     0   1   1 5869.567
## 4     0   1   2 5843.493
## 5     0   1   3 5843.344
## 6     0   1   4 5841.457
## 7     0   1   5 5843.141
## 8     0   1   6 5834.541
## 9     0   1   7 5834.305
## 10    0   1   8 5830.740
```

```
## 11    0   2   0 8120.101
## 12    0   2   1 5954.513
## 13    0   2   2 5877.456
## 14    0   2   3 5851.352
## 15    0   2   4 5851.187
## 16    0   2   5 5849.323
## 17    0   2   6 5851.013
## 18    0   2   7 5842.433
## 19    0   2   8 5842.191
## 20    1   1   0 5857.098
## 21    1   1   1 5854.918
## 22    1   1   2 5844.328
## 23    1   1   3 5829.282
## 24    1   1   4 5843.575
## 25    1   1   5 5843.023
## 26    1   1   6 5821.907
## 27    1   1   7 5823.183
## 28    1   1   8 5817.393
## 29    1   2   0 6921.833
## 30    1   2   1 5864.985
## 31    1   2   2 5862.802
## 32    1   2   3 5852.183
## 33    1   2   4 5837.130
## 34    1   2   5 5838.505
## 35    1   2   6 5840.510
## 36    1   2   7 5829.794
## 37    1   2   8 5831.067
## 38    2   1   0 5851.740
## 39    2   1   1 5849.748
## 40    2   1   2 5842.975
## 41    2   1   3 5845.217
## 42    2   1   4 5825.727
## 43    2   1   5 5827.671
## 44    2   1   6 5799.589
## 45    2   1   7 5825.861
## 46    2   1   8 5803.536
## 47    2   2   0 6508.807
## 48    2   2   1 5859.616
## 49    2   2   2 5857.637
## 50    2   2   3 5850.836
## 51    2   2   4 5838.351
## 52    2   2   5 5839.831
## 53    2   2   6 5840.669
## 54    2   2   7 5833.088
## 55    2   2   8 5812.533
## 56    3   1   0 5843.097
## 57    3   1   1 5827.264
## 58    3   1   2 5835.000
## 59    3   1   3 5846.987
## 60    3   1   4 5830.131
## 61    3   1   5 5825.779
## 62    3   1   6 5825.195
## 63    3   1   7 5797.504
## 64    3   1   8 5798.633
```

```
## 65    3   2   0 6396.359
## 66    3   2   1 5850.944
## 67    3   2   2 5835.127
## 68    3   2   3 5837.053
## 69    3   2   4 5832.436
## 70    3   2   5 5838.669
## 71    3   2   6 5835.981
## 72    3   2   7 5805.586
## 73    3   2   8 5810.869
## 74    4   1   0 5842.925
## 75    4   1   1 5842.136
## 76    4   1   2 5824.928
## 77    4   1   3 5821.560
## 78    4   1   4 5800.379
## 79    4   1   5 5790.224
## 80    4   1   6 5780.806
## 81    4   1   7 5793.719
## 82    4   1   8 5795.736
## 83    4   2   0 6308.649
## 84    4   2   1 5850.788
## 85    4   2   2 5837.085
## 86    4   2   3 5839.011
## 87    4   2   4 5829.448
## 88    4   2   5 5807.811
## 89    4   2   6 5798.185
## 90    4   2   7 5803.184
## 91    4   2   8 5803.892
## 92    5   1   0 5844.447
## 93    5   1   1 5843.745
## 94    5   1   2 5826.541
## 95    5   1   3 5821.256
## 96    5   1   4 5783.552
## 97    5   1   5 5794.233
## 98    5   1   6 5794.211
## 99    5   1   7 5795.945
## 100   5   1   8 5797.742
## 101   5   2   0 6282.260
## 102   5   2   1 5852.317
## 103   5   2   2 5837.529
## 104   5   2   3 5839.909
## 105   5   2   4 5829.177
## 106   5   2   5 5799.522
## 107   5   2   6 5800.246
## 108   5   2   7 5804.923
## 109   5   2   8 5806.928
## 110   6   1   0 5831.765
## 111   6   1   1 5810.066
## 112   6   1   2 5797.680
## 113   6   1   3 5820.181
## 114   6   1   4 5791.945
## 115   6   1   5 5794.779
## 116   6   1   6 5795.976
## 117   6   1   7 5797.996
## 118   6   1   8 5799.665
```

```
## 119   6   2   0 6170.763
## 120   6   2   1 5839.658
## 121   6   2   2 5817.954
## 122   6   2   3 5805.557
## 123   6   2   4 5807.371
## 124   6   2   5 5799.841
## 125   6   2   6 5802.420
## 126   6   2   7 5809.339
## 127   6   2   8 5807.416
## 128   7   1   0 5826.395
## 129   7   1   1 5810.316
## 130   7   1   2 5813.576
## 131   7   1   3 5795.785
## 132   7   1   4 5793.955
## 133   7   1   5 5795.698
## 134   7   1   6 5795.883
## 135   7   1   7 5785.913
## 136   7   1   8 5781.011
## 137   7   2   0 6160.156
## 138   7   2   1 5834.274
## 139   7   2   2 5818.198
## 140   7   2   3 5801.658
## 141   7   2   4 5806.940
## 142   7   2   5 5801.553
## 143   7   2   6 5803.482
## 144   7   2   7 5806.271
## 145   7   2   8 5798.172
## 146   8   1   0 5817.246
## 147   8   1   1 5805.916
## 148   8   1   2 5798.093
## 149   8   1   3 5799.058
## 150   8   1   4 5795.763
## 151   8   1   5 5797.691
## 152   8   1   6 5799.586
## 153   8   1   7 5782.706
## 154   8   1   8 5787.062
## 155   8   2   0 6041.007
## 156   8   2   1 5825.141
## 157   8   2   2 5813.814
## 158   8   2   3 5805.993
## 159   8   2   4 5806.959
## 160   8   2   5 5803.673
## 161   8   2   6 5805.673
## 162   8   2   7 5810.622
## 163   8   2   8 5797.308
```

```
orders <- orders[order(-orders$AIC),]
print(orders)
```

```
##       p   d   q      AIC
## 1   Inf Inf Inf      Inf
## 11    0   2   0 8120.101
## 29    1   2   0 6921.833
## 47    2   2   0 6508.807
```

```
## 65    3   2   0 6396.359
## 83    4   2   0 6308.649
## 101   5   2   0 6282.260
## 119   6   2   0 6170.763
## 137   7   2   0 6160.156
## 155   8   2   0 6041.007
## 12    0   2   1 5954.513
## 2     0   1   0 5946.709
## 13    0   2   2 5877.456
## 3     0   1   1 5869.567
## 30    1   2   1 5864.985
## 31    1   2   2 5862.802
## 48    2   2   1 5859.616
## 49    2   2   2 5857.637
## 20    1   1   0 5857.098
## 21    1   1   1 5854.918
## 102   5   2   1 5852.317
## 32    1   2   3 5852.183
## 38    2   1   0 5851.740
## 14    0   2   3 5851.352
## 15    0   2   4 5851.187
## 17    0   2   6 5851.013
## 66    3   2   1 5850.944
## 50    2   2   3 5850.836
## 84    4   2   1 5850.788
## 39    2   1   1 5849.748
## 16    0   2   5 5849.323
## 59    3   1   3 5846.987
## 41    2   1   3 5845.217
## 92    5   1   0 5844.447
## 22    1   1   2 5844.328
## 93    5   1   1 5843.745
## 24    1   1   4 5843.575
## 4     0   1   2 5843.493
## 5     0   1   3 5843.344
## 7     0   1   5 5843.141
## 56    3   1   0 5843.097
## 25    1   1   5 5843.023
## 40    2   1   2 5842.975
## 74    4   1   0 5842.925
## 18    0   2   7 5842.433
## 19    0   2   8 5842.191
## 75    4   1   1 5842.136
## 6     0   1   4 5841.457
## 53    2   2   6 5840.669
## 35    1   2   6 5840.510
## 104   5   2   3 5839.909
## 52    2   2   5 5839.831
## 120   6   2   1 5839.658
## 86    4   2   3 5839.011
## 70    3   2   5 5838.669
## 34    1   2   5 5838.505
## 51    2   2   4 5838.351
## 103   5   2   2 5837.529
```

```
## 33    1   2   4 5837.130
## 85    4   2   2 5837.085
## 68    3   2   3 5837.053
## 71    3   2   6 5835.981
## 67    3   2   2 5835.127
## 58    3   1   2 5835.000
## 8     0   1   6 5834.541
## 9     0   1   7 5834.305
## 138   7   2   1 5834.274
## 54    2   2   7 5833.088
## 69    3   2   4 5832.436
## 110   6   1   0 5831.765
## 37    1   2   8 5831.067
## 10    0   1   8 5830.740
## 60    3   1   4 5830.131
## 36    1   2   7 5829.794
## 87    4   2   4 5829.448
## 23    1   1   3 5829.282
## 105   5   2   4 5829.177
## 43    2   1   5 5827.671
## 57    3   1   1 5827.264
## 94    5   1   2 5826.541
## 128   7   1   0 5826.395
## 45    2   1   7 5825.861
## 61    3   1   5 5825.779
## 42    2   1   4 5825.727
## 62    3   1   6 5825.195
## 156   8   2   1 5825.141
## 76    4   1   2 5824.928
## 27    1   1   7 5823.183
## 26    1   1   6 5821.907
## 77    4   1   3 5821.560
## 95    5   1   3 5821.256
## 113   6   1   3 5820.181
## 139   7   2   2 5818.198
## 121   6   2   2 5817.954
## 28    1   1   8 5817.393
## 146   8   1   0 5817.246
## 157   8   2   2 5813.814
## 130   7   1   2 5813.576
## 55    2   2   8 5812.533
## 73    3   2   8 5810.869
## 162   8   2   7 5810.622
## 129   7   1   1 5810.316
## 111   6   1   1 5810.066
## 126   6   2   7 5809.339
## 88    4   2   5 5807.811
## 127   6   2   8 5807.416
## 123   6   2   4 5807.371
## 159   8   2   4 5806.959
## 141   7   2   4 5806.940
## 109   5   2   8 5806.928
## 144   7   2   7 5806.271
## 158   8   2   3 5805.993
```

```
## 147   8   1   1 5805.916
## 161   8   2   6 5805.673
## 72    3   2   7 5805.586
## 122   6   2   3 5805.557
## 108   5   2   7 5804.923
## 91    4   2   8 5803.892
## 160   8   2   5 5803.673
## 46    2   1   8 5803.536
## 143   7   2   6 5803.482
## 90    4   2   7 5803.184
## 125   6   2   6 5802.420
## 140   7   2   3 5801.658
## 142   7   2   5 5801.553
## 78    4   1   4 5800.379
## 107   5   2   6 5800.246
## 124   6   2   5 5799.841
## 118   6   1   8 5799.665
## 44    2   1   6 5799.589
## 152   8   1   6 5799.586
## 106   5   2   5 5799.522
## 149   8   1   3 5799.058
## 64    3   1   8 5798.633
## 89    4   2   6 5798.185
## 145   7   2   8 5798.172
## 148   8   1   2 5798.093
## 117   6   1   7 5797.996
## 100   5   1   8 5797.742
## 151   8   1   5 5797.691
## 112   6   1   2 5797.680
## 63    3   1   7 5797.504
## 163   8   2   8 5797.308
## 116   6   1   6 5795.976
## 99    5   1   7 5795.945
## 134   7   1   6 5795.883
## 131   7   1   3 5795.785
## 150   8   1   4 5795.763
## 82    4   1   8 5795.736
## 133   7   1   5 5795.698
## 115   6   1   5 5794.779
## 97    5   1   5 5794.233
## 98    5   1   6 5794.211
## 132   7   1   4 5793.955
## 81    4   1   7 5793.719
## 114   6   1   4 5791.945
## 79    4   1   5 5790.224
## 154   8   1   8 5787.062
## 135   7   1   7 5785.913
## 96    5   1   4 5783.552
## 153   8   1   7 5782.706
## 136   7   1   8 5781.011
## 80    4   1   6 5780.806
```

```
tail(orders)
```

```
##     p d q      AIC
## 154 8 1 8 5787.062
## 135 7 1 7 5785.913
## 96  5 1 4 5783.552
## 153 8 1 7 5782.706
## 136 7 1 8 5781.011
## 80  4 1 6 5780.806
```

```
arima.daily=arima(INTDaily[1:2512], order=c(4,1,6))
arima.weekly = arima(INTWeekly[1:521], order=c(4,1,5))
summary(arima.daily)
```

```
##           Length Class  Mode
## coef        10   -none- numeric
## sigma2       1   -none- numeric
## var.coef   100   -none- numeric
## mask        10   -none- logical
## loglik       1   -none- numeric
## aic          1   -none- numeric
## arma         7   -none- numeric
## residuals 2512   ts     numeric
## call         3   -none- call
## series       1   -none- character
## code         1   -none- numeric
## n.cond       1   -none- numeric
## nobs         1   -none- numeric
## model       10   -none- list
```

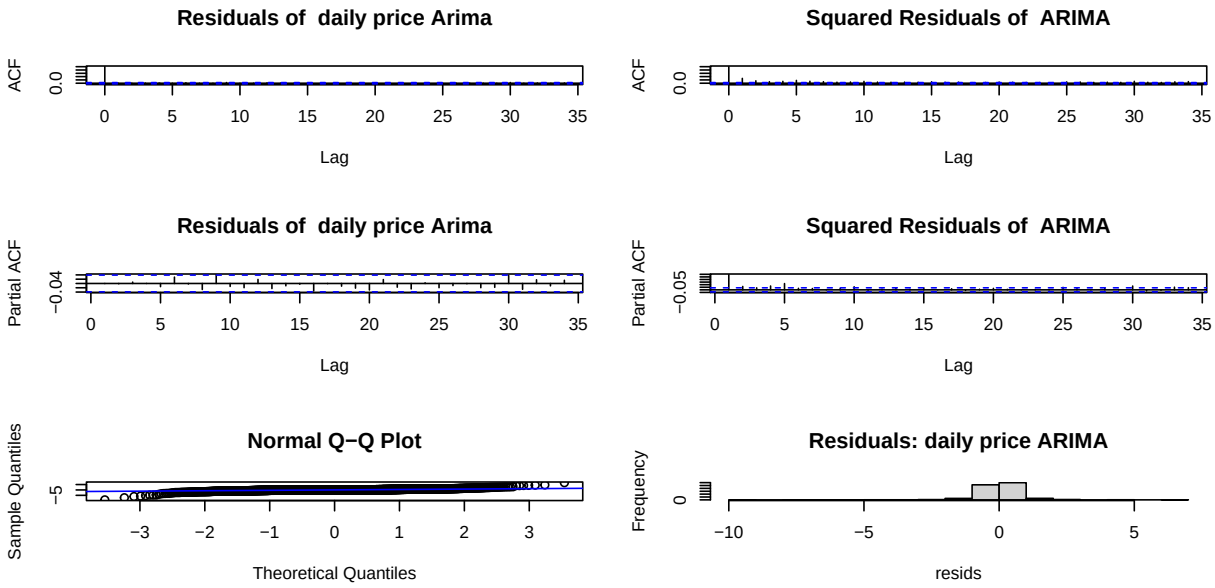
```
summary(arima.weekly)
```

```
##           Length Class  Mode
## coef        9    -none- numeric
## sigma2      1    -none- numeric
## var.coef   81    -none- numeric
## mask        9    -none- logical
## loglik      1    -none- numeric
## aic         1    -none- numeric
## arma        7    -none- numeric
## residuals 521    ts     numeric
## call        3    -none- call
## series      1    -none- character
## code        1    -none- numeric
## n.cond      1    -none- numeric
## nobs        1    -none- numeric
## model      10    -none- list
```

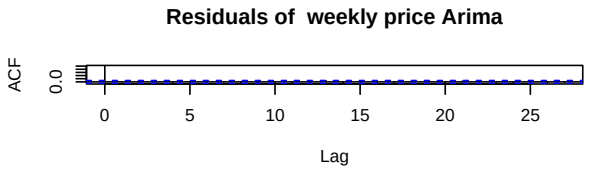
2b. Evaluate the model residuals and squared residuals using the ACF and PACF plots as well as hypothesis testing for serial correlation. What would you conclude based on this analysis?

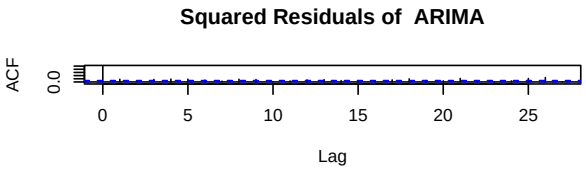
```
resids.daily <- resid(arima.daily)
resids.weekly <- resid(arima.weekly)
residplot=function(type,resids){
```

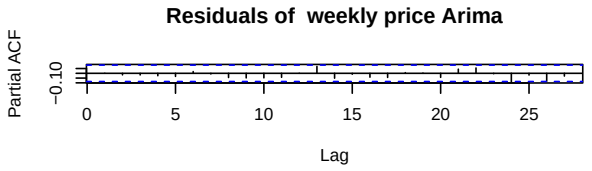
```
  par(mfrow=c(3,2))
  acf(resids,main=paste("Residuals of ",type,"Arima"))
  acf(resids^2,main="Squared Residuals of  ARIMA")
  pacf(resids,main=paste("Residuals of ",type,"Arima"))
  pacf(resids^2,main="Squared Residuals of  ARIMA")

  qqnorm(resids)
  qqline(resids, col="blue")
  hist(resids,main=paste("Residuals:",type, "ARIMA"))
}
residplot("daily price",resids.daily)
```

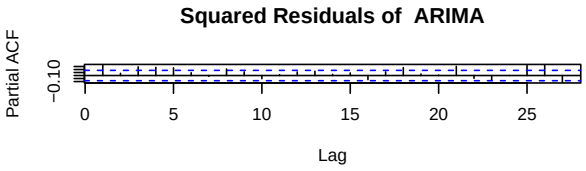


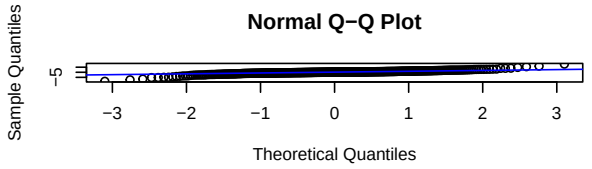

```
residplot('weekly price',resids.weekly)
```

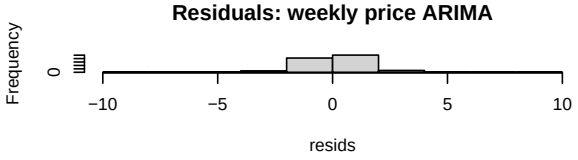













```
Box.test(resids.daily,lag=11,type='Ljung',fitdf=10)
```

```
##
##  Box-Ljung test
##
## data:  resids.daily
## X-squared = 10.782, df = 1, p-value = 0.001025
```

```
Box.test((resids.daily)^2,lag=11,type='Ljung',fitdf=10)
```

```
##
##  Box-Ljung test
##
## data:  (resids.daily)^2
## X-squared = 519.36, df = 1, p-value < 2.2e-16
```

```
Box.test(resids.weekly,lag=10,type='Ljung',fitdf=10)
```

```
##
##  Box-Ljung test
##
## data:  resids.weekly
## X-squared = 4.6423, df = 0, p-value < 2.2e-16
```

```
Box.test((resids.weekly)^2,lag=10,type='Ljung',fitdf=9)
```

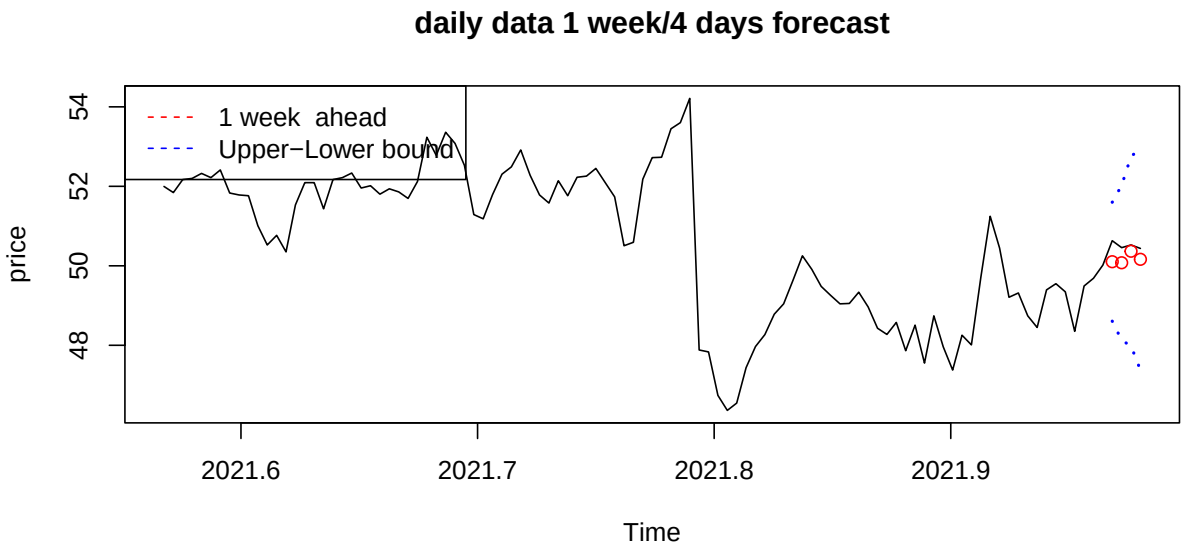
```
##
##  Box-Ljung test
##
## data:  (resids.weekly)^2
## X-squared = 98.118, df = 1, p-value < 2.2e-16
```

*Response: Question 2b

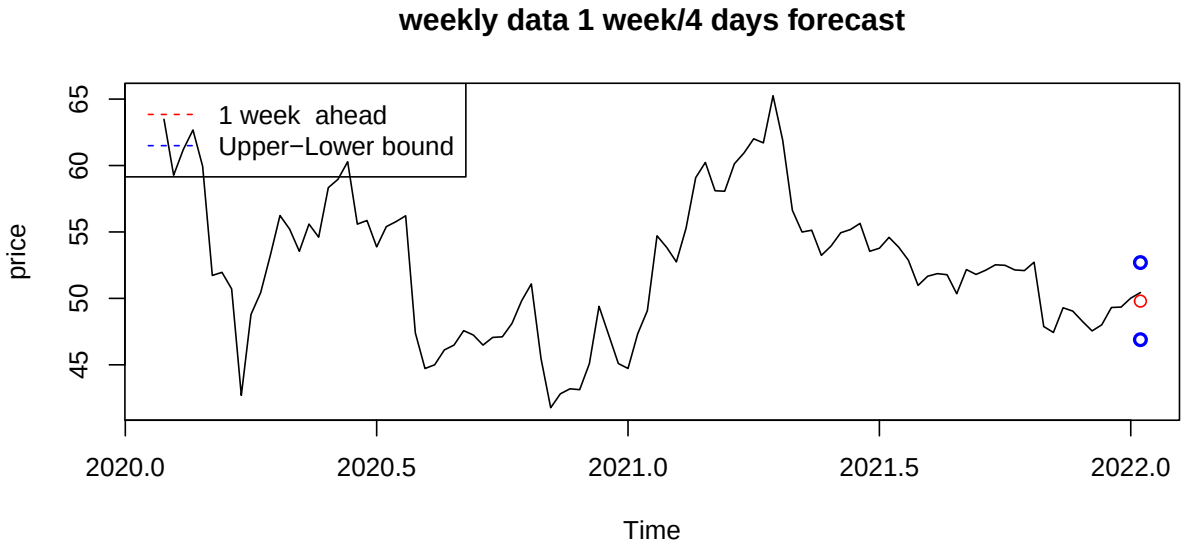
ACF and PACF plot for both daily and weekly data looked like white noise. However, acf and pacf plot for squared residual suggested that there are serial correlation between the variation at different time point. The p_value for Box-Ljung tests are very small, thus rejecting the null hypothesis and suggesting these residuals are correlated despite their acf plot looking like white noise. Moreover, Box_Ljung test suggested squared residuals of these models are clearly correlated.

2c. Apply the models identified in (2a) and forecast the last week of data. Plot the predicted data to compare the predicted values to the actual observed ones. Include 95% confidence intervals for the (mean) forecasts in the corresponding plots.

```
prediction=function(data,model,n_forward,n_backward,type,title){
n <- length(data)
outpred <- predict(model, n.ahead = n_forward) # 95% confidence interval
ubound <- outpred$pred + 1.96 * outpred$se
lbound <- outpred$pred - 1.96 * outpred$se
ymin <- min(data[(n - n_forward - n_backward):n])
ymax <- max(data[(n - n_forward - n_backward):n])

dates<- index(data)

plot(dates[(n - n_forward - n_backward):n], data[(n - n_forward - n_backward):n],
     type = "l", ylim=c(ymin,ymax), xlab = "Time", ylab = "price",main=title)
points(dates[(n-n_forward+1):n], outpred$pred, col = "red")
if (n_forward==1){
points(dates[(n-n_forward+1):n], ubound,lty = 3, lwd = 2, col = "blue")
points(dates[(n-n_forward+1):n], lbound,lty = 3, lwd = 2, col = "blue")}
else{lines(dates[(n-n_forward+1):n], ubound,lty = 3, lwd = 2, col = "blue")
lines(dates[(n-n_forward+1):n], lbound,lty = 3, lwd = 2, col = "blue")}


legend('topleft', legend = c(paste(type," ahead "), "Upper-Lower bound"),
  lty = 2, col = c("red", "blue"))
}

prediction(INTDaily,arima.daily,4,100,'1 week', "daily data 1 week/4 days forecast")
```




```
prediction(INTWeekly,arima.weekly,1,100,'1 week', "weekly data 1 week/4 days forecast")
```



2d. Calculate Mean Absolute Percentage Error (MAPE) and Precision Measure (PM) (PM only for daily data). How many observations are within the prediction bands? Compare the accuracy of the predictions for the daily and weekly time series using these two measures.

```
# outpred <- predict(arima.daily, n.ahead = 4)
# consump_pred=outpred$pred
# consump_true=INTDaily[2513:2516]
#
# ### Mean Absolute Percentage Error (MAPE)
# mean(abs(consump_pred-consump_true)/consump_true)
# ### Precision Measure (PM)
```

```
# sum((consump_pred-consump_true)^2)/sum((consump_true-mean(consump_true))^2)
#
# outpred <- predict(arima.weekly, n.ahead = 1)
# consump_pred=outpred$pred
# consump_true=INTWeekly[522:522]
#
# ### Mean Absolute Percentage Error (MAPE)
# mean(abs(consump_pred-consump_true)/consump_true)
# ### Precision Measure (PM)
# sum((consump_pred-consump_true)^2)/sum((consump_true-mean(consump_true))^2)

mapepm=function(data,model,n_forward){
  n=length(data)
  outpred <- predict(model, n.ahead = n_forward)
  consump_pred=outpred$pred
  consump_true=data[(n-n_forward+1):n]

### Mean Absolute Percentage Error (MAPE)
  print(mean(abs(consump_pred-consump_true)/consump_true))
### Precision Measure (PM)
  print(sum((consump_pred-consump_true)^2)/sum((consump_true-mean(consump_true))^2))

}
mapepm(INTDaily,arima.daily,4)
```

```
## [1] 0.006632701
## [1] 22.53261
```

```
mapepm(INTWeekly,arima.weekly,1)
```

```
## [1] 0.01273197
## [1] Inf
```

Response: Question 2d

All observations are within prediction bands. The accuracy for the prediction is decent judging by MAPE (around 1%), however, according to PM assay, the variation for these predictions are way above those within the observed data suggesting that these predictions are problematic.

Question 3: ARMA(p,q)-GARCH(m,n) for Return Stock Price (20 Points)

3a. Divide the data into training and testing data set, where the training data exclude the last week of data (December 27th - December 30th) with the testing data including the last week of data. Apply the iterative model to fit an ARMA(p,q)-GARCH(m,n) model by selecting the orders for p & q up to 5 and orders for m & n up to 2. Display the summary of the final model fit. Write up the equation of the estimated model.

```
length(dailyreturn)
```

```
## [1] 2515
```

```
length(weeklyreturn)
```

```
## [1] 521
```

```
train=dailyreturn[1:2511]

initialemodel=function(train){

  orders <- data.frame(Inf,Inf,Inf,Inf)
  names(orders) <- c("p","d","q","AIC")
  for (p in 0:5){
      for (q in 0:5) {
            possibleError <- tryCatch(
                orders<-rbind(orders,test_modelA(p,0,q,train)),
                error=function(e) e
                )
            if(inherits(possibleError, "error")) next
        }
    }

  orders <- orders[order(-orders$AIC),]
  print(tail(orders))

  test_modelAGG <- function(m,n){
      spec = ugarchspec(variance.model=list(garchOrder=c(m,n)),
                        mean.model=list(armaOrder=c(test=tail(orders,n=1)[,'p'],test=tail(orders,n=1)[,'q
                                        include.mean=T),
                        distribution.model="std")
      fit = ugarchfit(spec,train, solver = 'hybrid')
      current.bic = infocriteria(fit)[2]
      df = data.frame(m,n,current.bic)
      names(df) <- c("m","n","BIC")
      #print(paste(m,n,current.bic,sep=" "))
      return(df)
  }

  ordersAGG = data.frame(Inf,Inf,Inf)
  names(ordersAGG) <- c("m","n","BIC")

  for (m in 0:2){
    for (n in 0:2){
        possibleError <- tryCatch(
            ordersAGG<-rbind(ordersAGG,test_modelAGG(m,n)),
            error=function(e) e
        )
        if(inherits(possibleError, "error")) next
    }
}
ordersAGG <- ordersAGG[order(-ordersAGG$BIC),]
print(tail(ordersAGG))
return(c(tail(orders,n=1)[,1],tail(orders,n=1)[,3],tail(ordersAGG,n=1)[,1],tail(ordersAGG,n=1)[,2]))
}
```

```
initialemodel(train)
```

```
##    p d q       AIC
## 35 5 0 3 -13025.02
## 30 4 0 4 -13029.37
## 28 4 0 2 -13030.95
## 31 4 0 5 -13042.81
## 36 5 0 4 -13044.09
## 37 5 0 5 -13050.62
##   m n       BIC
## 2 0 1 -5.462108
## 3 0 2 -5.464694
## 9 2 2 -5.534004
## 8 2 1 -5.538088
## 6 1 2 -5.539187
## 5 1 1 -5.541353
```

```
## [1] 5 5 1 1
```

first model arma(5,5)+garch(1,1)

```
#ARMA update
#ARIMA-GARCH: Select ARIMA order
train=dailyreturn[1:2511]
armaupdate=function(m,n,train){

test_modelAGA <- function(p,q){
    spec = ugarchspec(variance.model=list(garchOrder=c(m,n)),
                      mean.model=list(armaOrder=c(p,q),
                                      include.mean=T),
                      distribution.model="std")
    fit = ugarchfit(spec, train, solver = 'hybrid')
    current.bic = infocriteria(fit)[2]
    df = data.frame(p,q,current.bic)
    names(df) <- c("p","q","BIC")

    return(df)
}

ordersAGA = data.frame(Inf,Inf,Inf)
names(ordersAGA) <- c("p","q","BIC")
for (p in 0:5){
    for (q in 0:5){
        possibleError <- tryCatch(
            ordersAGA<-rbind(ordersAGA,test_modelAGA(p,q)),
            error=function(e) e
        )
        if(inherits(possibleError, "error")) next
    }
}
ordersAGA <- ordersAGA[order(-ordersAGA$BIC),]
print(tail(ordersAGA))
```

```
return(c(tail(ordersAGA,n=1)[,1],tail(ordersAGA,n=1)[,2]))
}

armaupdate(1,1,train)
```

```
##    p q       BIC
## 9  1 1 -5.557084
## 4  0 2 -5.557203
## 14 2 0 -5.557237
## 3  0 1 -5.559993
## 8  1 0 -5.560025
## 2  0 0 -5.562078
```

```
## [1] 0 0
```

```
garchupdate=function(p,q,train){
  test_modelAGG <- function(m,n){
    spec = ugarchspec(variance.model=list(garchOrder=c(m,n)),
                      mean.model=list(armaOrder=c(p,q),
                                      include.mean=T), distribution.model="std")
    fit = ugarchfit(spec, train, solver = 'hybrid')
    current.bic = infocriteria(fit)[2]
    df = data.frame(m,n,current.bic)
    names(df) <- c("m","n","BIC")
    return(df)}

  ordersAGG = data.frame(Inf,Inf,Inf)
  names(ordersAGG) <- c("m","n","BIC")

for (m in 0:2){
    for (n in 0:2){
        possibleError <- tryCatch(
            ordersAGG<-rbind(ordersAGG,test_modelAGG(m,n)),
            error=function(e) e
        )
        if(inherits(possibleError, "error")) next
    }
}
ordersAGG <- ordersAGG[order(-ordersAGG$BIC),]
print(tail(ordersAGG))
return(c(tail(ordersAGG[,1],n=1),tail(ordersAGG[,2],n=1)))
}

garchupdate(0,0,train)
```

```
##    m n       BIC
## 5  1 0 -5.522034
## 8  2 0 -5.533824
## 10 2 2 -5.557706
## 9  2 1 -5.559014
## 7  1 2 -5.560824
## 6  1 1 -5.562078
```

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

Second model arma(0,0) + garch(1,1)

```
train=dailyreturn[1:2511]
spec.2 = ugarchspec(variance.model=list(garchOrder=c(1,1)),
                 mean.model=list(armaOrder=c(0, 0),
                 include.mean=T), distribution.model="std")
final.model.2.daily= ugarchfit(spec.2, train, solver = 'hybrid')
print(final.model.2.daily)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : std
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000864    0.000249   3.4668 0.000527
## omega   0.000006    0.000004   1.3855 0.165895
## alpha1  0.059339    0.005843  10.1558 0.000000
## beta1   0.922901    0.005244 175.9943 0.000000
## shape   3.874024    0.163613  23.6779 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000864    0.000246  3.51722 0.000436
## omega   0.000006    0.000021  0.29439 0.768457
## alpha1  0.059339    0.047657  1.24511 0.213092
## beta1   0.922901    0.048211 19.14302 0.000000
## shape   3.874024    1.206159  3.21187 0.001319
##
## LogLikelihood : 7002.76
##
## Information Criteria
## ------------------------------------
##
## Akaike       -5.5737
## Bayes        -5.5621
## Shibata      -5.5737
## Hannan-Quinn -5.5695
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      2.667  0.1025
```

```
## Lag[2*(p+q)+(p+q)-1][2]     2.837  0.1553
## Lag[4*(p+q)+(p+q)-1][5]     4.635  0.1848
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1352  0.7131
## Lag[2*(p+q)+(p+q)-1][5]    0.4856  0.9599
## Lag[4*(p+q)+(p+q)-1][9]    1.0995  0.9818
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.1622 0.500 2.000  0.6872
## ARCH Lag[5]    0.5298 1.440 1.667  0.8748
## ARCH Lag[7]    0.9400 2.315 1.543  0.9231
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  3.3028
## Individual Statistics:
## mu     0.06057
## omega  0.24997
## alpha1 0.26218
## beta1  0.19870
## shape  0.13391
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.28 1.47 1.88
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias          0.02508 0.9800
## Negative Sign Bias 1.44564 0.1484
## Positive Sign Bias 1.20228 0.2294
## Joint Effect       3.54104 0.3155
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     13.60       0.8067
## 2    30     19.72       0.9015
## 3    40     37.83       0.5230
## 4    50     39.92       0.8193
##
##
## Elapsed time : 0.2146881
```

```
final.model.2.daily@fit$coef
```

```
##           mu        omega       alpha1        beta1        shape
## 8.644884e-04 6.094694e-06 5.933865e-02 9.229014e-01 3.874024e+00
```

final model arma(0,0)+garch(1,1), it is converged since it is the same as the second model. for daily return $Y_t = 0.000864$ $\sigma_t^2 = 0.000006 + 0.059339 * \epsilon_{t-1}^2 + 0.922901 * \sigma_{t-1}^2$

for weekly data

```
length(weeklyreturn)
```

```
## [1] 521
```

```
train=weeklyreturn[1:520]
initialemodel(train)
```

```
##    p d q       AIC
## 15 2 0 1 -1978.395
## 10 1 0 2 -1978.396
## 24 3 0 4 -1978.996
## 29 4 0 3 -1979.015
## 9  1 0 1 -1980.378
## 30 4 0 4 -1980.909
##    m n       BIC
## 3  0 1 -3.821564
## 2  0 0 -3.822071
## 10 2 2 -3.830839
## 9  2 1 -3.840353
## 7  1 2 -3.842866
## 6  1 1 -3.844859
```

```
## [1] 4 4 1 1
```

first model arma (4,4)+garch(1,1)

```
armaupdate(1,1,train)
```

```
##    p q       BIC
## 15 2 1 -3.867031
## 16 2 2 -3.877754
## 8  1 0 -3.878533
## 3  0 1 -3.878534
## 10 1 2 -3.880991
## 2  0 0 -3.890537
```

```
## [1] 0 0
```

second model arma(0,0)+garch(1,1)

```
garchupdate(0,0,train)
```

```
##    m n       BIC
## 10 2 2 -3.866981
## 2  0 0 -3.870339
## 5  1 0 -3.875447
## 9  2 1 -3.878068
## 7  1 2 -3.879008
## 6  1 1 -3.890537
```

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

final model arma(0,0)+garch(1,1), it is converged since it is the same as the second model.

```
train=weeklyreturn[1:520]
spec.2 = ugarchspec(variance.model=list(garchOrder=c(1,1)),
                    mean.model=list(armaOrder=c(0, 0),
                    include.mean=T), distribution.model="std")
final.model.2.weekly = ugarchfit(spec.2, train, solver = 'hybrid')
print(final.model.2.weekly)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : std
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.003340    0.001344   2.4855 0.012937
## omega   0.000041    0.000026   1.5971 0.110234
## alpha1  0.059608    0.022887   2.6044 0.009203
## beta1   0.911402    0.032045  28.4414 0.000000
## shape   5.388766    1.283756   4.1977 0.000027
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.003340    0.001294   2.5814 0.009840
## omega   0.000041    0.000023   1.8071 0.070753
## alpha1  0.059608    0.021233   2.8074 0.004995
## beta1   0.911402    0.027729  32.8682 0.000000
## shape   5.388766    1.375455   3.9178 0.000089
##
## LogLikelihood : 1027.174
##
## Information Criteria
```

```
## ------------------------------------
##
## Akaike       -3.9314
## Bayes        -3.8905
## Shibata      -3.9316
## Hannan-Quinn -3.9154
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.02675  0.8701
## Lag[2*(p+q)+(p+q)-1][2]   0.13541  0.8955
## Lag[4*(p+q)+(p+q)-1][5]   2.04803  0.6071
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.01195  0.9129
## Lag[2*(p+q)+(p+q)-1][5]   0.96139  0.8684
## Lag[4*(p+q)+(p+q)-1][9]   1.77049  0.9306
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.3617 0.500 2.000  0.5475
## ARCH Lag[5]    0.9988 1.440 1.667  0.7333
## ARCH Lag[7]    1.2499 2.315 1.543  0.8698
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.2687
## Individual Statistics:
## mu     0.05013
## omega  0.06782
## alpha1 0.07477
## beta1  0.07258
## shape  0.54772
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.28 1.47 1.88
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.5795 0.5625
## Negative Sign Bias  0.2168 0.8284
## Positive Sign Bias  0.4285 0.6685
## Joint Effect        0.3575 0.9489
##
##
```

```
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     26.92       0.1065
## 2    30     32.00       0.3199
## 3    40     50.46       0.1034
## 4    50     53.85       0.2942
##
##
## Elapsed time : 0.091537
```

```
final.model.2.weekly@fit$coef
```

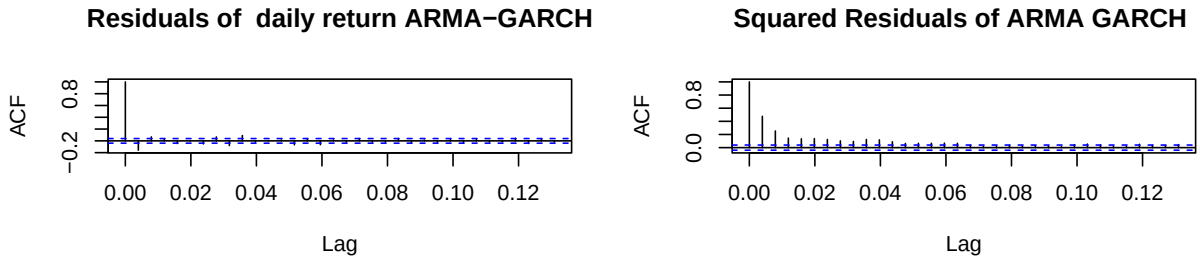
```
##           mu        omega       alpha1        beta1        shape
## 3.340448e-03 4.098275e-05 5.960805e-02 9.114016e-01 5.388766e+00
```

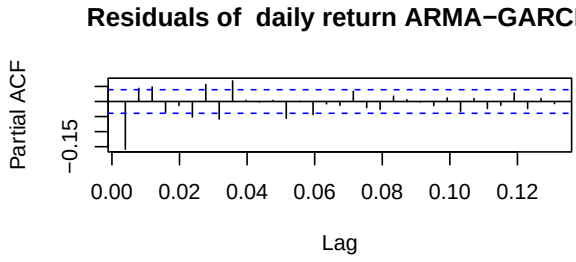
final model arma(0,0)+garch(1,1), it is converged since it is the same as the second model. for weeklyreturn $Y_t = 0.003340$ $\sigma_t^2 = 0.000041 + 0.059608 * \epsilon_{t-1}^2 + 0.911402 * \sigma_{t-1}^2$

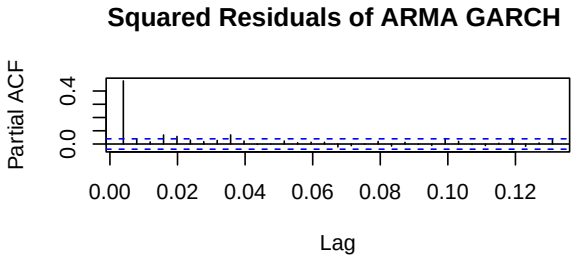
Response: Question 3a See above

3b. Evaluate the model residuals and squared residuals using the ACF and PACF plots as well as hypothesis testing for serial correlation. What would you conclude based on this analysis?

```
garchresids.daily <- ts(residuals(final.model.2.daily),start=c(2012,2),freq=252)

residplot=function(type,resids){
  par(mfrow=c(2,2))
  acf(resids,main=paste("Residuals of ",type,"ARMA-GARCH"))
  acf(resids^2,main="Squared Residuals of ARMA GARCH ")
  pacf(resids,main=paste("Residuals of ",type,"ARMA-GARCH"))
  pacf(resids^2,main="Squared Residuals of ARMA GARCH ")
  qqnorm(resids)
  qqline(resids, col="blue")
  hist(resids,main=paste("Residuals:",type, "ARMA-Garch"))
}
residplot("daily return",garchresids.daily)
```





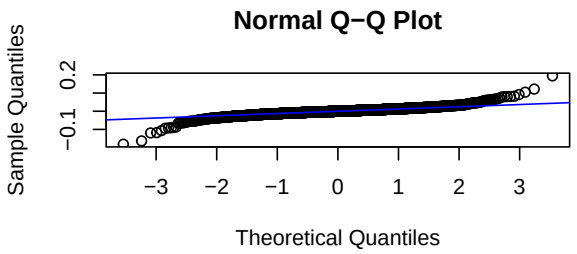


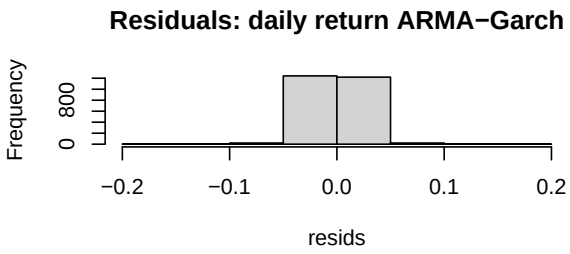

```
Box.test(garchresids.daily,lag=10,type='Ljung')
```

```
##
##  Box-Ljung test
##
## data:  garchresids.daily
## X-squared = 136.77, df = 10, p-value < 2.2e-16
```

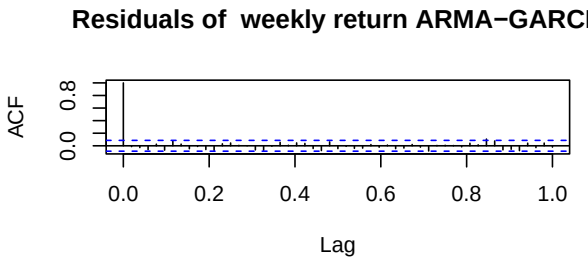
```
Box.test(garchresids.daily^2,lag=10,type='Ljung')
```

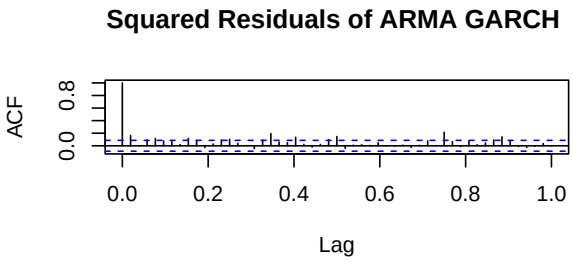
```
##
##  Box-Ljung test
##
## data:  garchresids.daily^2
## X-squared = 1038.7, df = 10, p-value < 2.2e-16
```

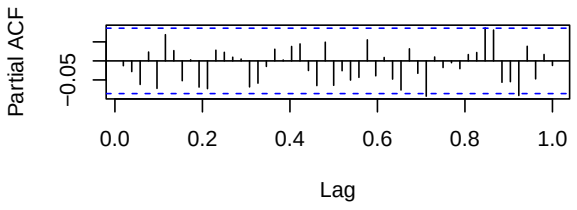


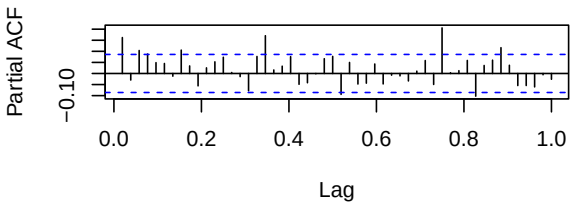



```
garchresids.weekly <- ts(residuals(final.model.2.weekly),start=c(2012,2),freq=52)

residplot=function(type,resids){
  par(mfrow=c(2,2))
  acf(resids,lag.max=52,main=paste("Residuals of ",type,"ARMA-GARCH"))
  acf(resids^2,lag.max=52,main="Squared Residuals of ARMA GARCH ")
  pacf(resids,lag.max=52,main=paste("Residuals of ",type,"ARMA-GARCH"))
  pacf(resids^2,lag.max=52,main="Squared Residuals of ARMA GARCH ")
  qqnorm(resids)
  qqline(resids, col="blue")
  hist(resids,main=paste("Residuals:",type, "ARMA-Garch"))
}
residplot("weekly return",garchresids.weekly)
```







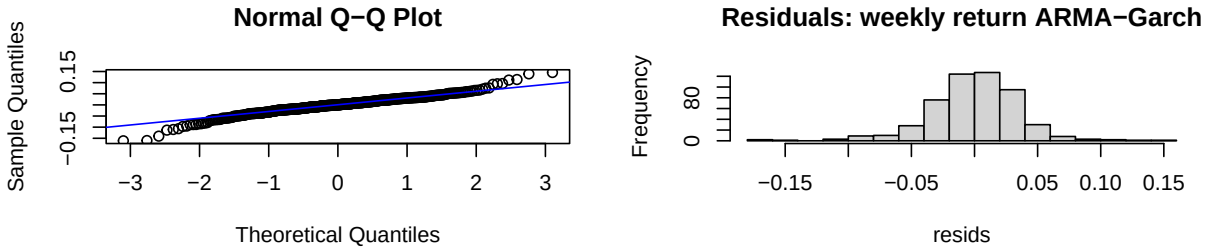


```
Box.test(garchresids.weekly,lag=10,type='Ljung')
```

```
##
##  Box-Ljung test
##
## data:  garchresids.weekly
## X-squared = 11.554, df = 10, p-value = 0.316
```

```
Box.test(garchresids.weekly^2,lag=10,type='Ljung')
```

```
##
##  Box-Ljung test
##
## data:  garchresids.weekly^2
## X-squared = 43.254, df = 10, p-value = 4.479e-06
```



Response: Question 3b

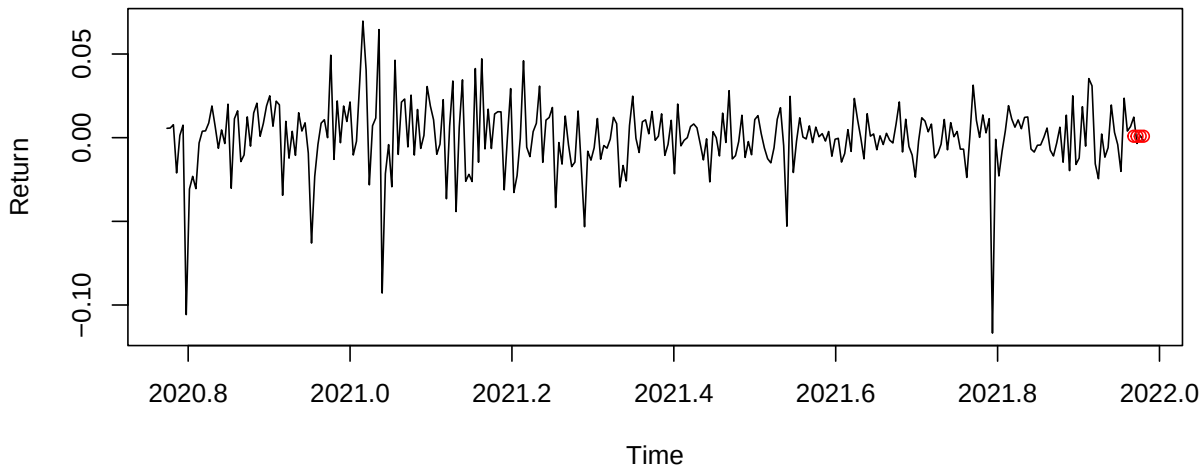
The acf plot of the residuals of daily return data looks like AR(1) process. The peak on acf plot suggested that residual squared had strong autocorrelation within the first few orders, however they dropped more quickly than arima model. On the other hand, most peaks of pacf fell within the significant range except for the first peak for both residuals and residual square, once again suggesting an AR(1) model. Nevertheless, the p-values of Box Ljung test are small, suggesting that both residual and residual square are correlated.

The acf plot of the residuals of weekly return data looks like white noise. The peak on acf plot dropped quickly, suggesting that residual square are weakly stationary. On the other hand, all peaks of weekly return residuals and most peak of weekly return residual squares were within the significant range on pacf, once again suggesting weak stationarity. The p-values of Box Ljung test showed that residual of weekly return data was indeed weakly stationary but not the residual square.

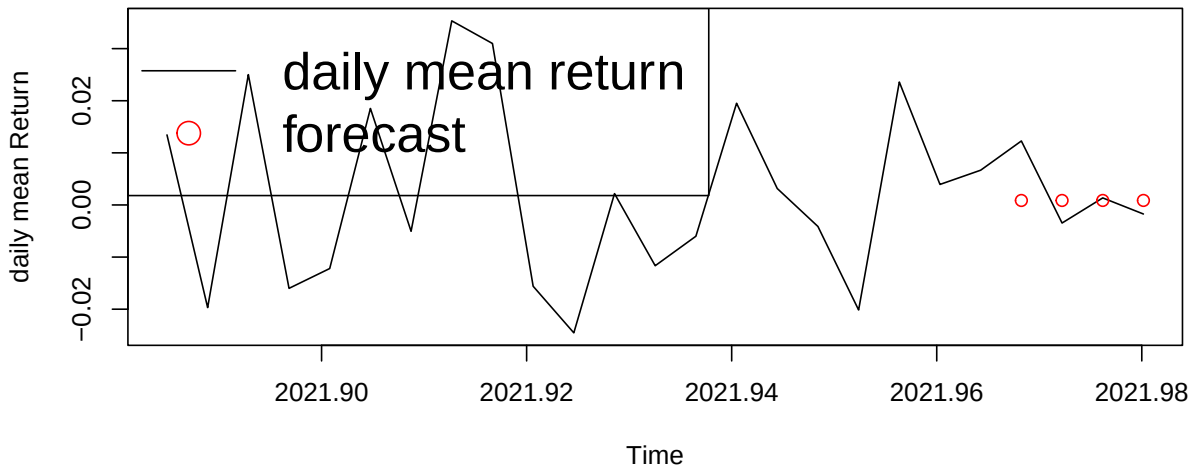
3c. Apply the model identified in (3a) and forecast the mean and the variance of the last week of data. Plot the predicted data to compare the predicted values to the actual observed ones. Interpret the results, particularly comparing forecast using daily versus weekly data.

```
train=dailyreturn[1:2511]
test=dailyreturn[2512:2515]
nfore = 4
fore.series.daily = NULL
fore.sigma.daily = NULL

spec.2 = ugarchspec(variance.model=list(garchOrder=c(1,1)),
                    mean.model=list(armaOrder=c(0, 0),
                    include.mean=T), distribution.model="std")
final.model.2.daily= ugarchfit(spec.2, train, solver = 'hybrid')

for(f in 1: nfore){
    ## Fit models
    data = train
    if(f>2)
       data = c(train,test[1:(f-1)])
    final.model.2.daily.fore = ugarchfit(spec.2, data, solver = 'hybrid')
    ## Forecast
    fore = ugarchforecast(final.model.2.daily.fore, n.ahead=1)
    fore.series.daily= c(fore.series.daily, fore@forecast$seriesFor)
    fore.sigma.daily = c(fore.sigma.daily, fore@forecast$sigmaFor)

}
```

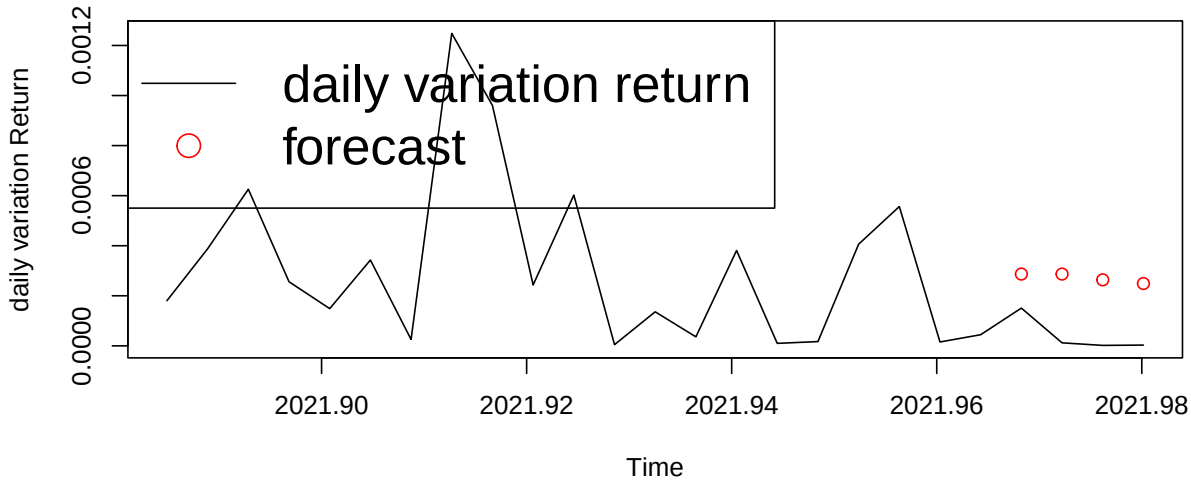
```
n_forward=4
n_backward=300

n=length(dailyreturn)
dates=index(dailyreturn)
plot(dates[(n - n_forward - n_backward):n], dailyreturn[(n - n_forward - n_backward):n],type = "l", xla
points(dates[(n-n_forward+1):n],fore.series.daily , col = "red")
```



```
returnresidplot=function(type, data,fore,foren_forward, n_backward) {



  n=length(data)
  dates=index(data)
  plot(dates[(n - n_forward - n_backward):n], data[(n - n_forward - n_backward):n],type = "l", xlab = "
  points(dates[(n-n_forward+1):n],fore , col = "red")
  legend("topleft", legend=c(paste(type,"return"),"forecast"),lty=c(1,NA),pch=c(NA,1),col=c("black","re
}

returnresidplot('daily mean',dailyreturn,fore.series.daily, 4,20)
```



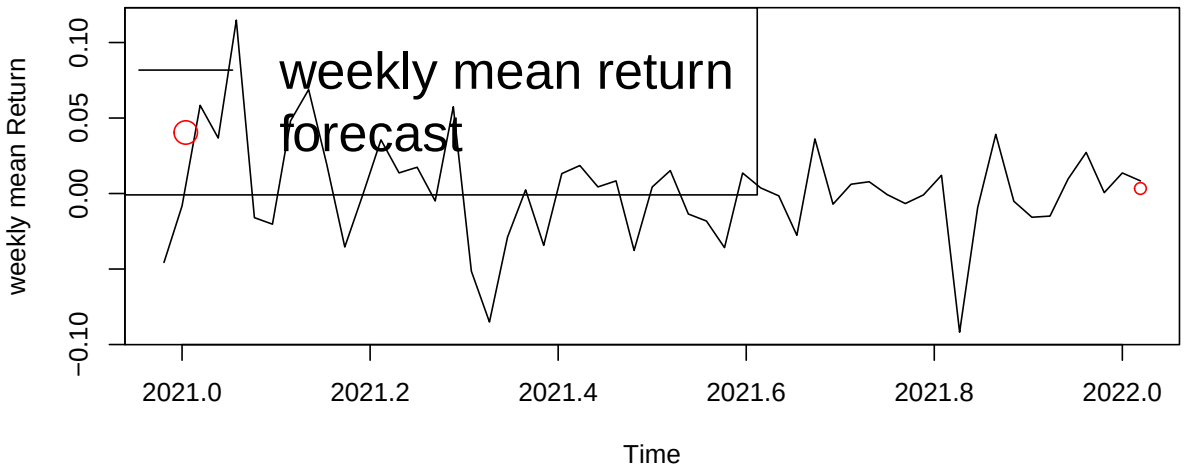

```
returnresidplot('daily variation', dailyreturn^2,fore.sigma.daily^2,4,20)
```



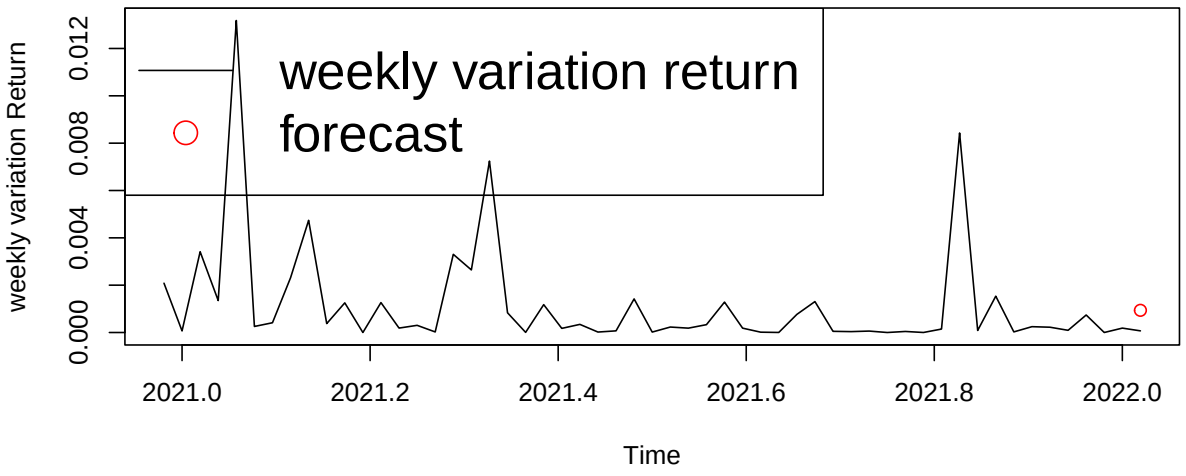

```
train=weeklyreturn[1:520]
test=weeklyreturn[521]

spec.2 = ugarchspec(variance.model=list(garchOrder=c(1,1)),
                    mean.model=list(armaOrder=c(0, 0),
                    include.mean=T), distribution.model="std")
final.model.2.weekly = ugarchfit(spec.2, train, solver = 'hybrid')
nfore = 1
fore.series.weekly = NULL
fore.sigma.weekly = NULL



for(f in 1: nfore){
    ## Fit models
    data = train

    final.model.2.weekly.fore = ugarchfit(spec.2, data, solver = 'hybrid')
    ## Forecast
    fore = ugarchforecast(final.model.2.weekly.fore, n.ahead=1)
    fore.series.weekly= c(fore.series.weekly, fore@forecast$seriesFor)
    fore.sigma.weekly = c(fore.sigma.weekly, fore@forecast$sigmaFor)

}
```

```
returnresidplot_w=function(type, data,fore,foren_forward, n_backward) {
  n=length(data)
  dates=index(data)
  plot(dates[(n - n_forward - n_backward):n], data[(n - n_forward - n_backward):n],type = "l", xlab = "
  points(dates[n],fore , col = "red")
  legend("topleft", legend=c(paste(type,"return"),"forecast"),lty=c(1,NA),pch=c(NA,1),col=c("black","re
}
```

```
returnresidplot_w("weekly mean",weeklyreturn,fore.series.weekly,1, 50)
```




```
returnresidplot_w("weekly variation",weeklyreturn^2,fore.sigma.weekly^2,1, 50)
```



Response: Question 3c

Since ARMA (0,0) is used for both models, the mean prediction is constant for both model, which didn't capture the variation in the observed data. It could be that the variation we saw are not due to the conditional mean, but the conditional variability. On the other hand, variation prediction for the daily data seemed to have captured the trend of the variation for the short test period for daily return data. Due to the small size of the prediction period, both predictions are largely inconclusive.

3d. Calculate Mean Absolute Percentage Error (MAPE) and Precision Measure (PM) for the mean forecasts

(PM should net be calculated for weekly data). Compare the accuracy of the predictions for the daily and weekly time series using these two measures.

```
train=dailyreturn[1:2511]
test=dailyreturn[2512:2515]

### Mean Absolute Percentage Error (MAPE)
paste('MAPE of daily return:',mean(abs(fore.series.daily - test)/abs(test)))
```

```
## [1] "MAPE of daily return: 1.0093977738253"
```

```
paste('pm of dairly return:',sum((fore.series.daily - test)^2)/sum((test-mean(test))^2))
```

```
## [1] "pm of dairly return: 1.04133375250791"
```

```
train=weeklyreturn[1:520]
test=weeklyreturn[521]

### Mean Absolute Percentage Error (MAPE)
paste("MAPE of weekly return",mean(abs(fore.series.weekly - test)/abs(test)))
```

```
## [1] "MAPE of weekly return 0.60139951152233"
```

Response: Question 3d

Based on the MAPE, the mean prediction are not accurate at all probably due to the small prediction size and the fact that the variance in the observed data are results from variation rather the variation of conditional mean. Nevertheless, according to PM test, the variation of the prediction is at the same level of the variation of the observed data for daily return. This suggested that ARMA-Garch model have at least captured the variability of the observed data in the prediction period.

Question 4: Reflection on the Modeling and Forecasting (10 points)

Response: Question 4 Based on the analysis above, discuss the application of ARIMA on the stock price versus the application of ARMA-GARCH on the stock return. How do the models fit the data? How well do the Overall, the prediction effectiveness of the ARIMA model is much better than the prediction of ARMA-GARCH model for the mean of the data. However, ARIMA failed to control the heteroscedasticity and the variation are constantly increasing for ARIMA. GARCH model can eliminate the heteroscedasticity for the prediction and largely captured the variation of the observed data. However, the prediction accuracy of GARCH model is not as accurate as ARIMA model. ARIMA model prediction worked slightly better for daily return data than weekly return data. ARMA-GARCH model worked slightly better for weekly return data. However, the size of the train and test samples are different for these two datasets. To predict short term stock price, it is better to use ARIMA model. To capture the long-term variability in stork return price return, ARMA-GARCH model may have its advantages. When applying these models, it is important to understand data and use proper transformation. It is also important to understand that both mean and variation should be taken into consideration when performing data analysis. Although Arima model may fit the data better, it may have correlated and increasing volatilities. The addition of GARCH model will be useful to eliminate the heteroskedasticity, model the variation and in turn generate a more accurate and sophisticated models.models predict? How do the models perform when using daily versus weekly data? Would you use one approach over another for different settings? What are some specific points of caution one would need to consider when applying those models?