In [1]:

```
# Set up the default parameters
# 1. The code block will be shown in the document
# 2. set up figure display size
# 3. turn off all the warnings and messages

knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(fig.width = 8, fig.height = 4)
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
```

Background

Individuals stock prices tend to exhibit high amounts of non-constant variance, and thus ARIMA models build 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

In [2]:

```
## 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)
library(quantmod,warn.conflicts=FALSE)
library(xts,warn.conflicts=FALSE)
options(warn=-1)
```

```
Warning message:
"package 'zoo' was built under R version 3.6.3"Warning message:
"package 'lubridate' was built under R version 3.6.3"Warning message:
"package 'mgcv' was built under R version 3.6.3"Loading required package: nlme
Warning message:
"package 'nlme' was built under R version 3.6.3"This is mgcv 1.8-35. For overview type
'help("mgcv-package")'.
Warning message:
"package 'rugarch' was built under R version 3.6.3"Loading required package: parallel
Warning message:
"package 'quantmod' was built under R version 3.6.3"Loading required package: xts
Warning message:
"package 'xts' was built under R version 3.6.3"Loading required package: TTR
Warning message:
"package 'TTR' was built under R version 3.6.3"Registered S3 method overwritten by 'quan
tmod':
  method            from
  as.zoo.data.frame zoo
```

In [3]:

```
#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?

Daily data would be more beneficial to 1) Day tradinghat look at trends and do it on a daily basis

2) Trading before and after Earniongs releases for the interested stocks

Weekly would be ideal for 1) long tradingd stack on different days (buy sell > 3 days))

2) Stock Option tradinghis will help in tdetermmining them to see stock probablity afte a period of time)

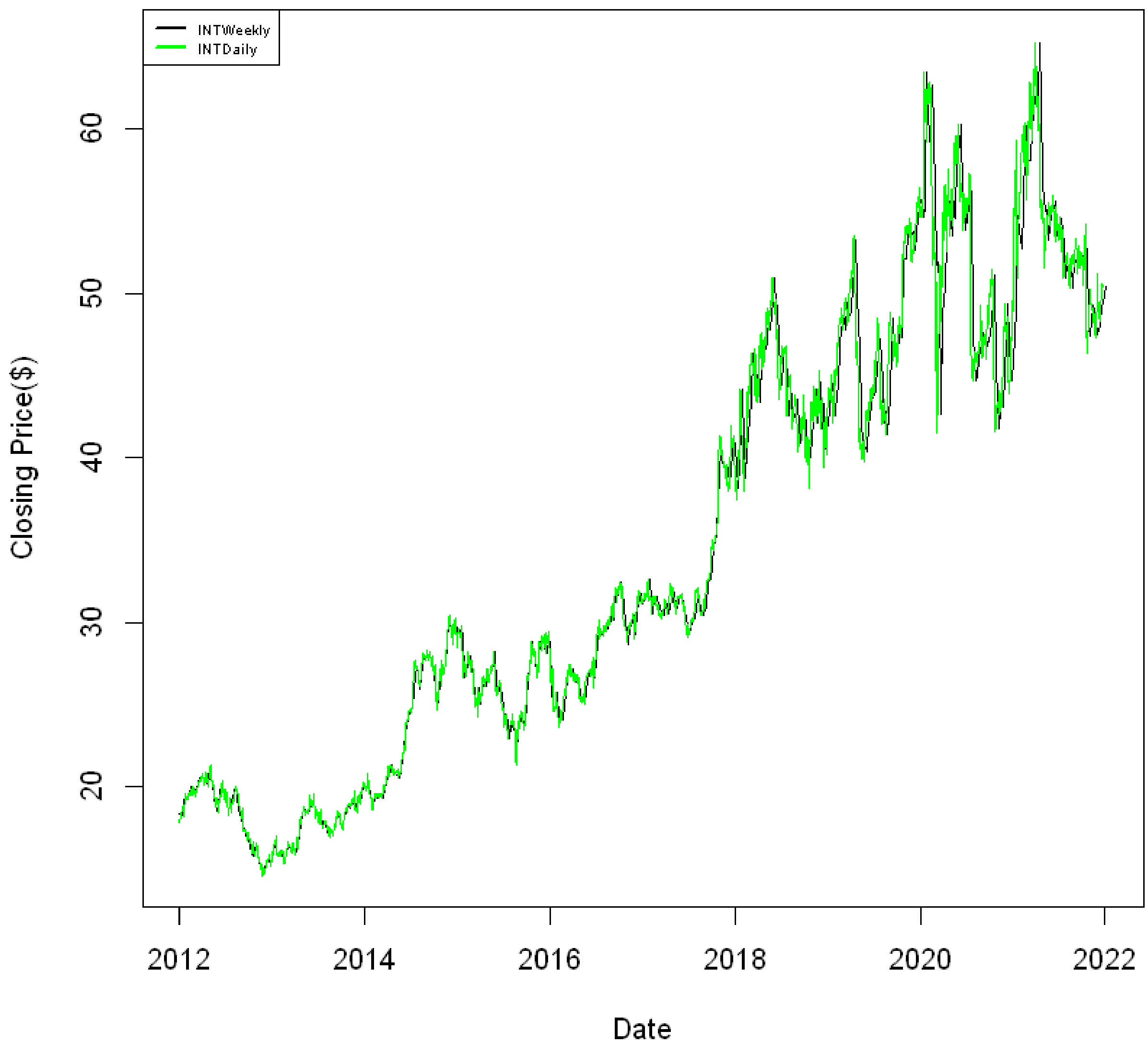
3) long term stock holders

Response: Question 1a

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

In [4]:

```
plot(INTWeekly,type = 'l', ,xlab="Date",ylab="Closing Price($)")
lines(INTDaily,col='green')
legend("topleft",legend = c("INTWeekly","INTDaily"),col = c("black","green"),lwd=2,cex
```



Response: Question 1b

Daily and Weekly prices are similar with Daily adjusted closed prices are more fluctuating than Weekly adjusted closed priced

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?

In [5]:

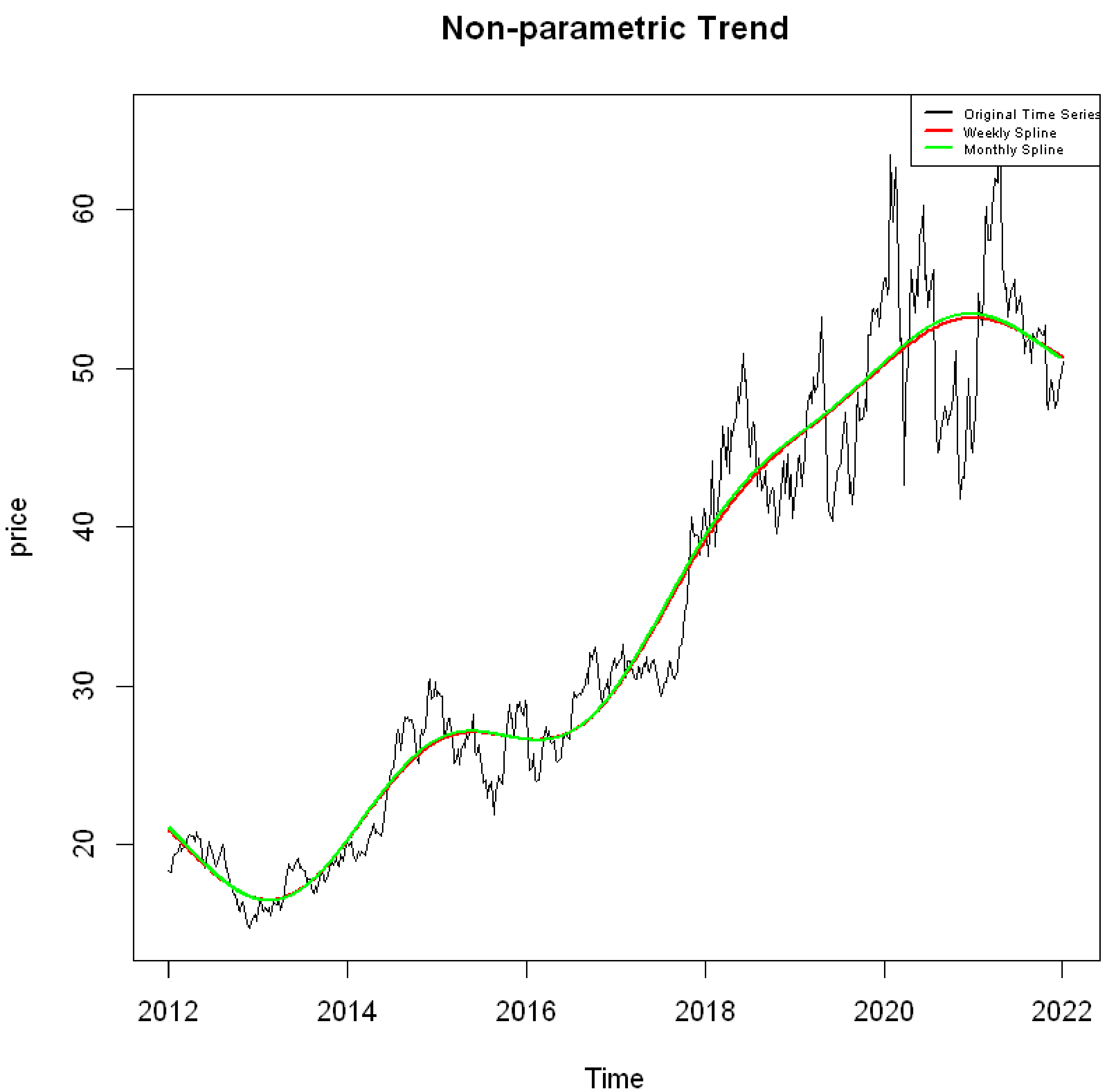
```
# Timestamp creation and Modelfitting
time.pts.weekly = c(1:length(INTWeekly))
time.pts.weekly = c(time.pts.weekly - min(time.pts.weekly))/max(time.pts.weekly)

time.pts.daily= c(1:length(INTDaily))
time.pts.daily= c(time.pts.daily- min(time.pts.daily))/max(time.pts.daily)

INTWeekly.gam <- gam(INTWeekly~ s(time.pts.weekly))
INTWeekly.fit.gam = ts(fitted(INTWeekly.gam),start=c(2012,1,1),freq=52)
```

```
INTDaily.gam <- gam(INTDaily ~ s(time.pts.daily))
INTDaily.fit.gam = ts(fitted(INTDaily.gam),start=c(2012,1,1),freq=252)


ts.plot(INTWeekly,type='l',ylab="price",main="Non-parametric Trend")
lines(INTWeekly.fit.gam,lwd=2,col="red")
lines(INTDaily.fit.gam, lwd=2, col="green")
legend("topright",legend = c("Original Time Series","Weekly Spline","Monthly Spline"),c
```



Response: Question 1c

The trend estimated using weekly and daily data are almost identical, hence the trend estimation primarily captures the overall pattern regardless whether the data are less or more granular.

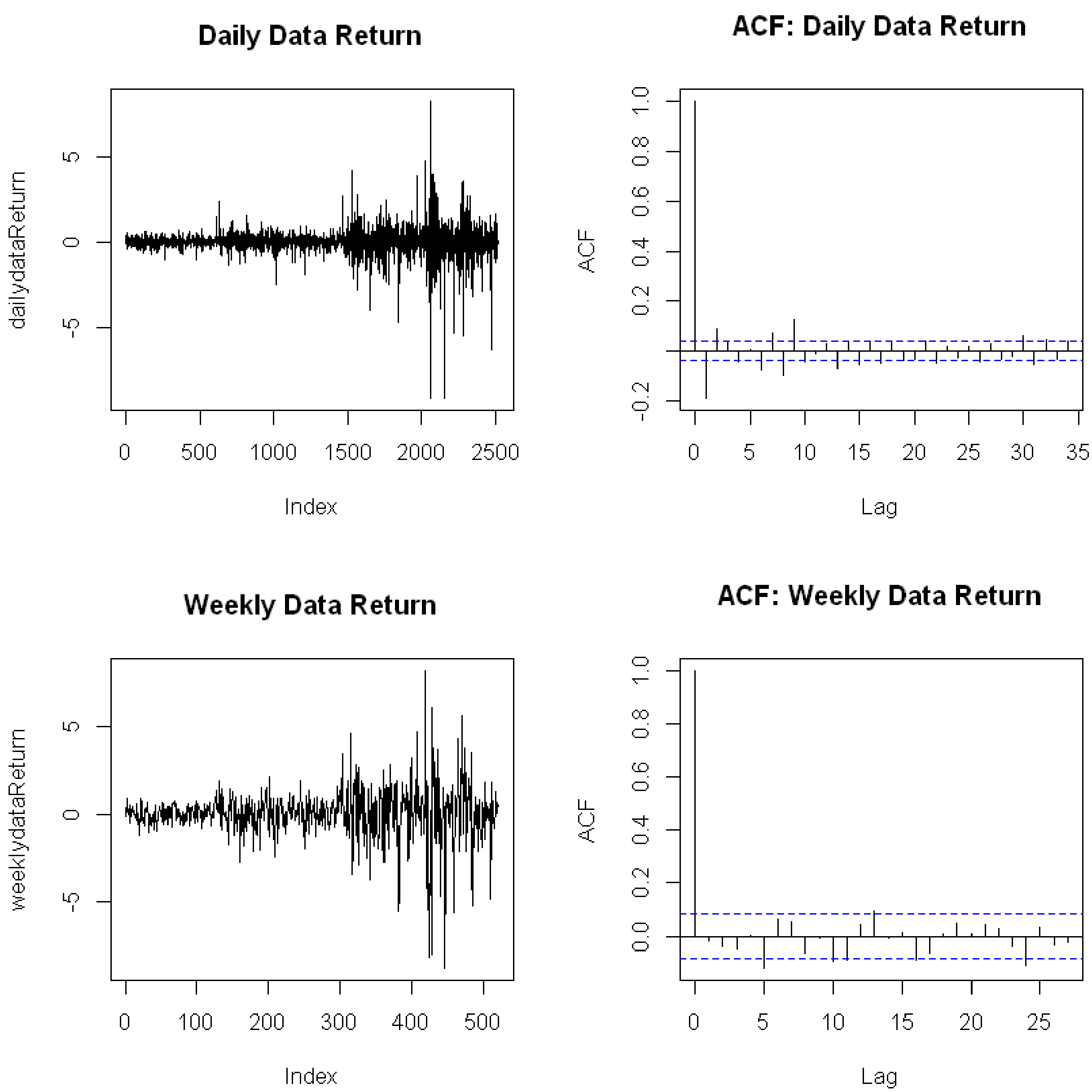
1d. Consider the return stock price computed as provided in the canvas homework assignment. Apply this formula to compute the return price based on the daily and weekly time series data. Plot the return time series and their corresponding ACF plots. How do the return time series compare in terms of stationarity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

In [6]:

```
#dailydataReturn <- diff(log(Cl(dailydata)))
#weeklydataReturn <- diff(log(Cl(weeklydata)))

dailydataReturn <- diff(Cl(dailydata))
weeklydataReturn <- diff(Cl(weeklydata))

par(mfrow=c(2,2))
plot(dailydataReturn,type="l",main="Daily Data Return")
acf(dailydataReturn,main="ACF: Daily Data Return")
plot(weeklydataReturn,type="l",main="Weekly Data Return")
acf(weeklydataReturn,main="ACF: Weekly Data Return")
```



Response: Question 1d

Daily Data Return

1) Does not indicate constant mean

2) Doesnot indicate constant variance

3) ACF plot indicates that the autocorrelation is small for all lags>1 and does not appear to have periodicity.

This indicates the Daily Return time series is non-stationarity.

Weekly Data Return

1) Does not indicate constant mean

2) Doesnot indicate constant variance

3) ACF plot indicates that the autocorrelation is small for all lags>o and does not appear to have periodicity.

This indicates the Daily Return time series is non-stationarity.

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.

In [7]:

```
weeklytestlimit = 1
dailytestlimit = 4
weeklydata.train <- weeklydata[1:(dim(weeklydata)[1]-weeklytestlimit),]
weeklydata.test <- weeklydata[(dim(weeklydata)[1]-weeklytestlimit+1):(dim(weeklydata)[1
dailydata.train <- dailydata[1:(dim(dailydata)[1]-dailytestlimit),]
dailydata.test <- dailydata[(dim(dailydata)[1]-dailytestlimit+1):(dim(dailydata)[1]),]


# Weekly Data
weekly_AIC = data.frame(p=0, d=0, q=0, aic=0)
row_num = 1
norder=8
for(p in 1:norder)
{
    for(q in 1:norder)
    {
        for (d in 1:2){
        model_pdq = arima(weeklydata.train$Adj.Close ,order = c(p,d,q), method='ML')
        weekly_AIC[row_num,] = c(p, d, q, AIC(model_pdq))
        row_num = row_num + 1
        }
    }
}
weekly_AIC_sorted = weekly_AIC[order(weekly_AIC$aic),]
print("ARIMA Model Fitting sorted in inc reasingorder of AIC values for Weekly Close pr
```

```
head(weekly_AIC_sorted)
#print(weekly_AIC_sorted)
porder_weekly = weekly_AIC_sorted$p[1]
qorder_weekly = weekly_AIC_sorted$q[1]
dorder_weekly = weekly_AIC_sorted$d[1]

weekly_AIC_model = arima(weeklydata.train$Adj.Close,order = c(porder_weekly,dorder_week
summary(weekly_AIC_model)
weekly_AIC_model$coef
```

```
[1] "ARIMA Model Fitting sorted in inc reasingorder of AIC values for Weekly Close pric
e"
```

	p	d	q	aic
57	4	1	5	1907.319
75	5	1	6	1907.638
89	6	1	5	1907.639
79	5	1	8	1907.679
71	5	1	4	1907.708
119	8	1	4	1907.913

```
          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        4    -none- call
series      1    -none- character
code        1    -none- numeric
n.cond      1    -none- numeric
nobs        1    -none- numeric
model      10    -none- list
```

ar1	-0.249901239581762
ar2	-0.521431178373267
ar3	-0.375708163650839
ar4	-0.826243899253883
ma1	0.227811200735854
ma2	0.512392719587481
ma3	0.351787612252432
ma4	0.877708379215986
ma5	-0.126408487736456

In [8]:

```
# daily Data
daily_AIC = data.frame(p=0, d=0, q=0, aic=0)
row_num = 1
norder=8
for(p in 1:norder)
{
    for(q in 1:norder)
```

```
    {
        for (d in 1:2){
        model_pdq = arima(dailydata.train$Adj.Close ,order = c(p,d,q), method='ML')
        daily_AIC[row_num,] = c(p, d, q, AIC(model_pdq))
        row_num = row_num + 1
        }
    }
}
daily_AIC_sorted = daily_AIC[order(daily_AIC$aic),]
print("ARIMA Model Fitting sorted in inc reasingorder of AIC values for daily Close pri
head(daily_AIC_sorted)
porder_daily = daily_AIC_sorted$p[1]
qorder_daily = daily_AIC_sorted$q[1]
dorder_daily = daily_AIC_sorted$d[1]

daily_AIC_model = arima(dailydata.train$Adj.Close,order = c(porder_daily,dorder_daily,q
summary(daily_AIC_model)
daily_AIC_model$coef
```

```
[1] "ARIMA Model Fitting sorted in inc reasingorder of AIC values for daily Close price"
```

	p	d	q	aic
111	7	1	8	5780.560
59	4	1	6	5780.700
125	8	1	7	5782.840
71	5	1	4	5783.459
109	7	1	7	5785.721
127	8	1	8	5786.819

```
          Length Class  Mode
coef        15   -none- numeric
sigma2       1   -none- numeric
var.coef   225   -none- numeric
mask        15   -none- logical
loglik       1   -none- numeric
aic          1   -none- numeric
arma         7   -none- numeric
residuals 2512   ts     numeric
call         4   -none- call
series       1   -none- character
code         1   -none- numeric
n.cond       1   -none- numeric
nobs         1   -none- numeric
model       10   -none- list
```

ar1	-0.568902083337216
ar2	0.344907674814606
ar3	0.554539392501215
ar4	-0.119472885640176
ar5	-0.320080781790722
ar6	0.305456449009798
ar7	0.683686763770072
ma1	0.403703285490172
ma2	-0.353799093092775
ma3	-0.411505274814559

ma4	0.149126353127209
ma5	0.211576019093588
ma6	-0.428912221203548
ma7	-0.586860943415112
ma8	0.0755200246549408

The selected models are as follows:

Weekly Arima Model: ARIMA(4,1,5) with AICc = 1907.319

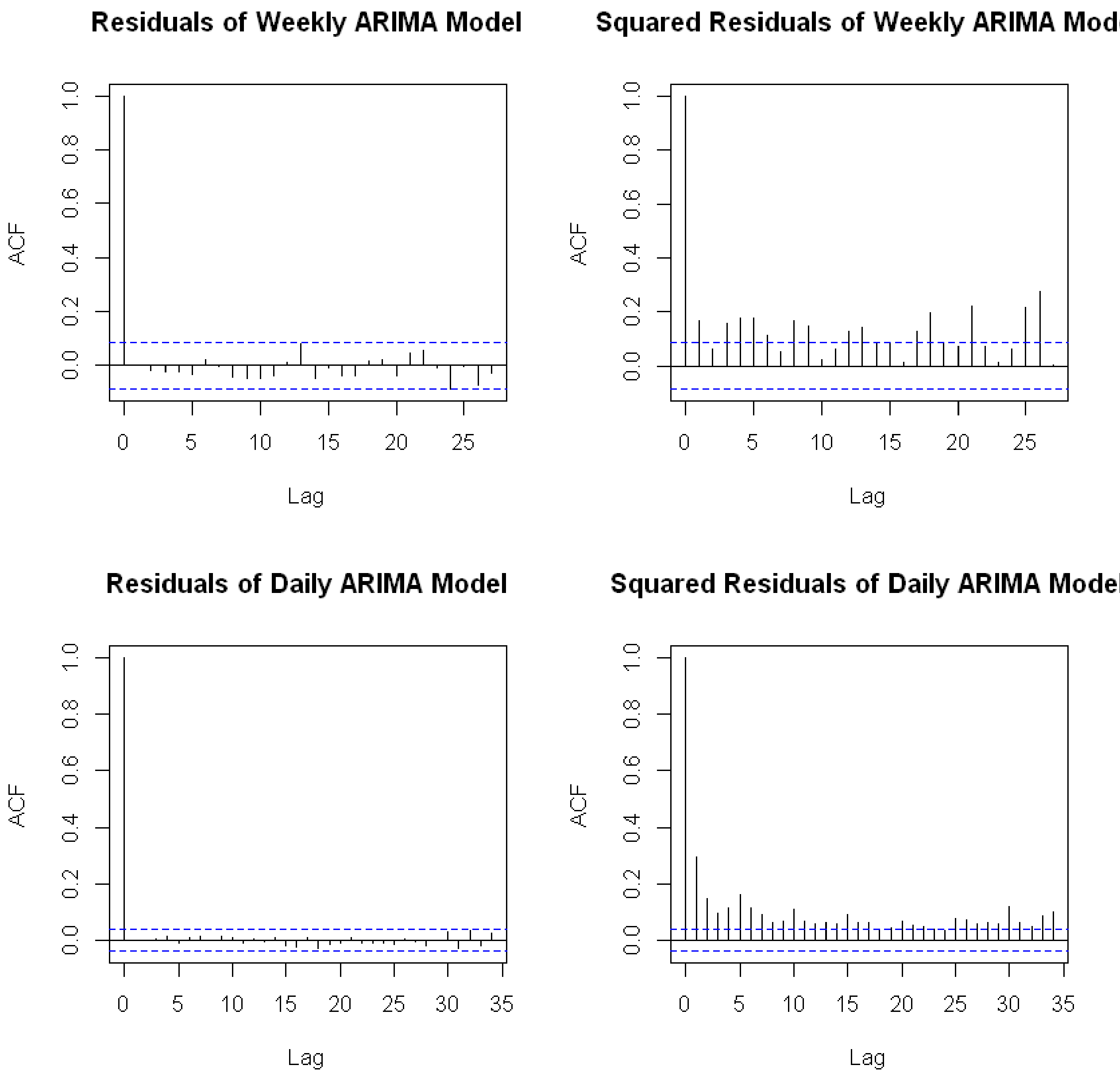
Daily Arima Model : ARIMA(7,1,8) with AICc = 5780.560

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?

In [9]:

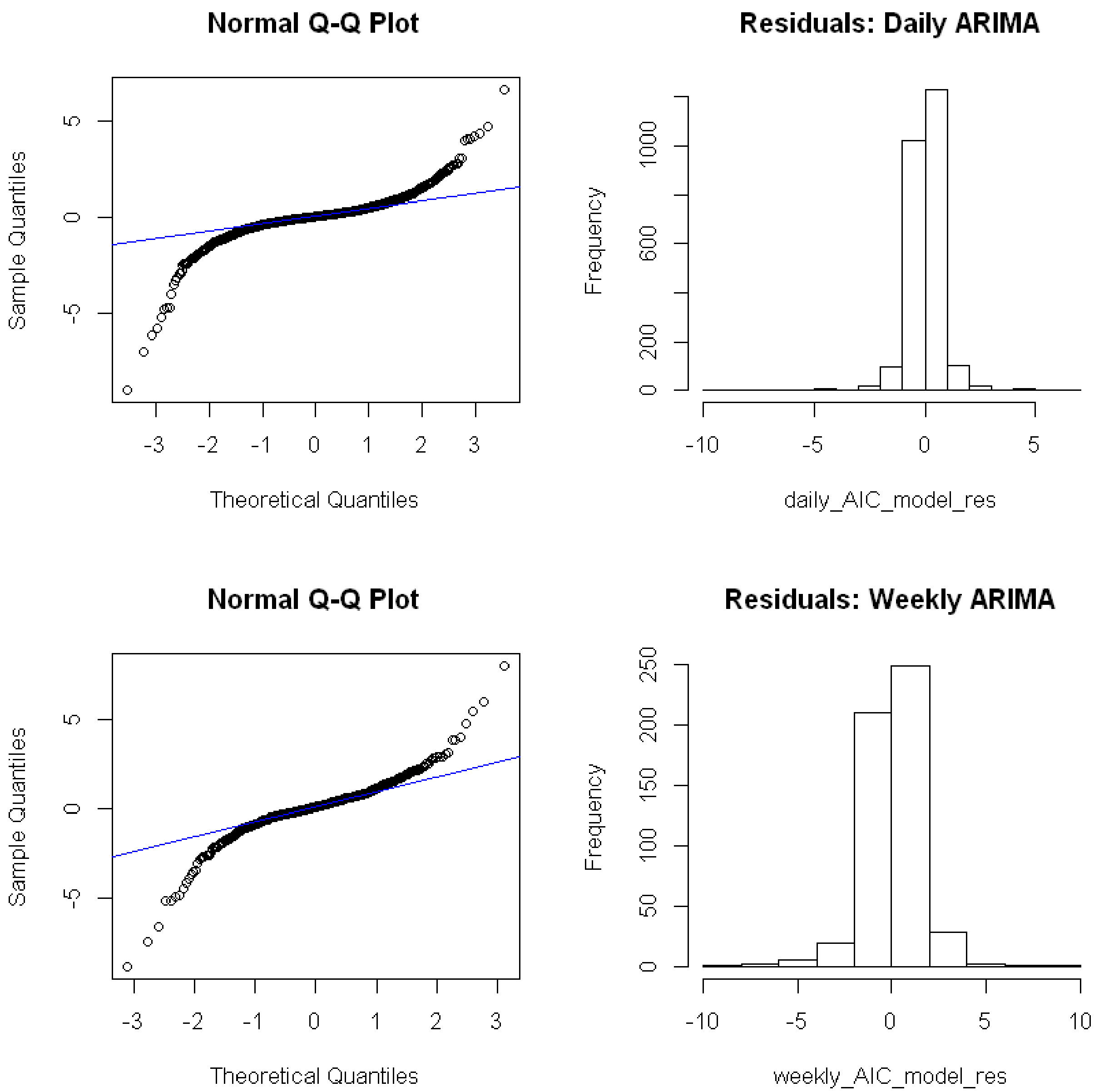
```
weekly_AIC_model_res = weekly_AIC_model$residuals
daily_AIC_model_res = daily_AIC_model$residuals

par(mfrow=c(2,2))
acf(weekly_AIC_model_res,main="Residuals of Weekly ARIMA Model")
acf(weekly_AIC_model_res^2,main="Squared Residuals of Weekly ARIMA Model")
acf(daily_AIC_model_res,main="Residuals of Daily ARIMA Model")
acf(daily_AIC_model_res^2,main="Squared Residuals of Daily ARIMA Model")
```



In [10]:

```
par(mfrow=c(2,2))
qqnorm(daily_AIC_model_res)
qqline(daily_AIC_model_res, col="blue")
hist(daily_AIC_model_res,main="Residuals: Daily ARIMA")
qqnorm(weekly_AIC_model_res)
qqline(weekly_AIC_model_res, col="blue")
hist(weekly_AIC_model_res,main="Residuals: Weekly ARIMA")
```



In [43]:

```
# test for serial correlation in residuals
Box.test(daily_AIC_model_res,lag=9,type='Ljung',fitdf=8)
Box.test(weekly_AIC_model_res,lag=3,type='Ljung',fitdf=2)

# test for serial correlation in squared residuals
Box.test((daily_AIC_model_res)^2,lag=9,type='Ljung',fitdf=8)
Box.test((weekly_AIC_model_res)^2,lag=3,type='Ljung',fitdf=2)
```

```
	Box-Ljung test

data:  daily_AIC_model_res
X-squared = 11.055, df = 1, p-value = 0.0008844
	Box-Ljung test

data:  weekly_AIC_model_res
X-squared = 0.14148, df = 1, p-value = 0.7068
	Box-Ljung test

data:  (daily_AIC_model_res)^2
X-squared = 469.15, df = 1, p-value < 2.2e-16
```

```
	Box-Ljung test

data:  (weekly_AIC_model_res)^2
X-squared = 32.389, df = 1, p-value = 1.262e-08
```

Response: Question 2b

Daily Residuals: • visually they do no appear to be normally distributed. The mean appears constant but the variance changes over time. Histogram and QQ plot appears nearly normally distributed.

Weekly Residuals: • visually they appear to be very close to being normally distributed. Mean and variance appear to be constant. Histogram and QQ plot appears nearly normally distributed.

The p-values for testing uncorrelated residuals are very small for Daily data(showing correlation) but is high for Weekly data showing non-correlation. However, the ACF plots from the previous slide look similarly to those of white noise.

The p-values for testing uncorrelated squared residuals are also very small indicating that we reject the null hypothesis, and thus, conclude that the squared residuals are correlated.

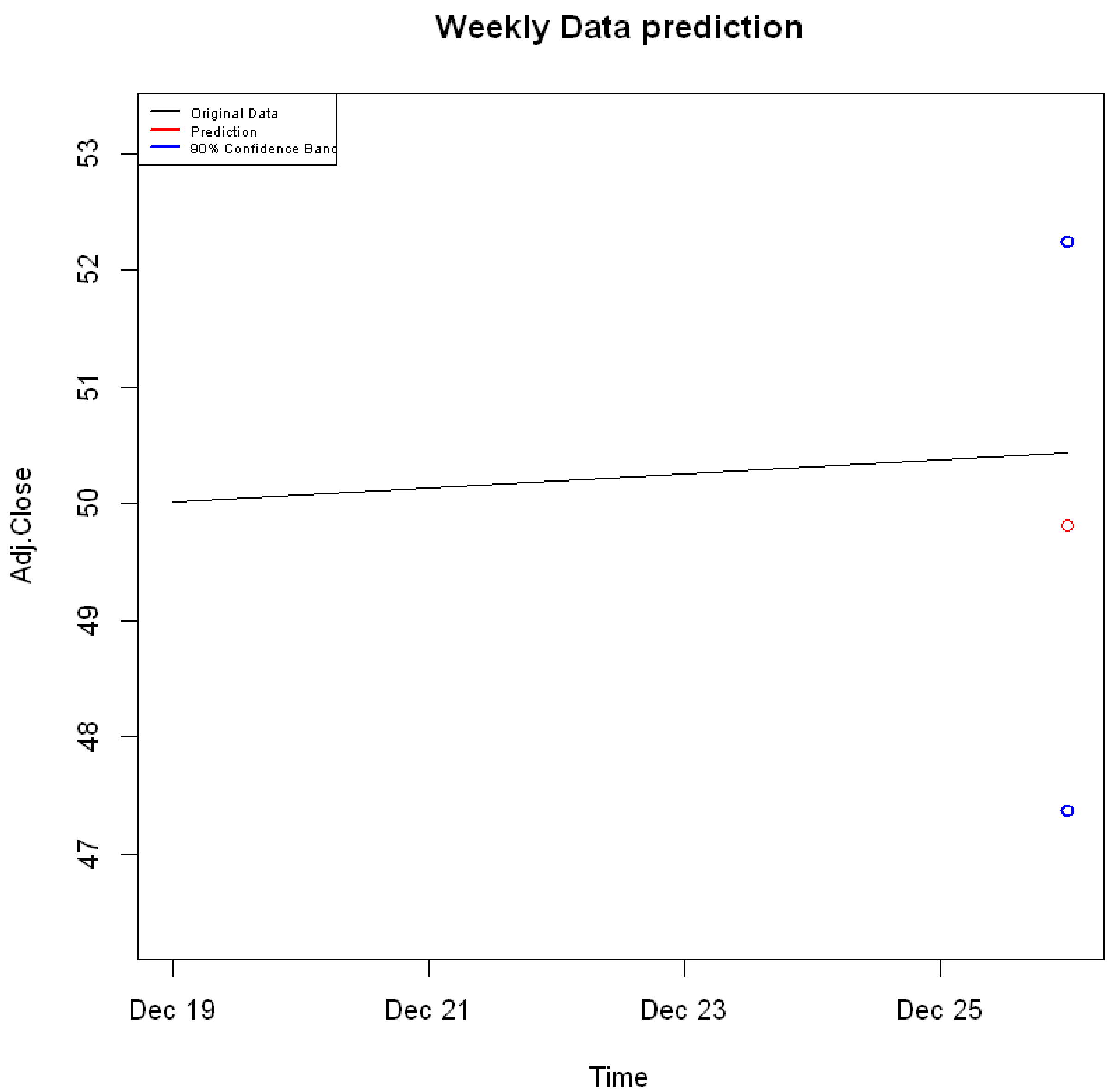
2c. Apply the model identified in (2a) and forecast the last week of data. Plot the predicted data to compare the predicted values to the actual observed ones. Include 95% confidence intervals for the forecasts in the corresponding plots.

In [12]:

```
weekly_model_pred = as.vector(predict(weekly_AIC_model,n.ahead = 1))

weekly_model_pred_ubound = weekly_model_pred$pred+1.645*weekly_model_pred$se
weekly_model_pred_lbound = weekly_model_pred$pred-1.645*weekly_model_pred$se

ymin = min(weekly_model_pred_lbound)-1
ymax = max(weekly_model_pred_ubound)+1
plot( weeklydata[(dim(weeklydata)[1]-weeklytestlimit):(dim(weeklydata)[1]),],type="l",
points(weeklydata.test$Date,weekly_model_pred$pred,col="red")
points(weeklydata.test$Date,weekly_model_pred_ubound,lty=3,lwd= 2, col="blue")
points(weeklydata.test$Date,weekly_model_pred_lbound,lty=3,lwd= 2, col="blue")
legend("topleft",legend = c("Original Data","Prediction","90% Confidence Band"),col = c
```

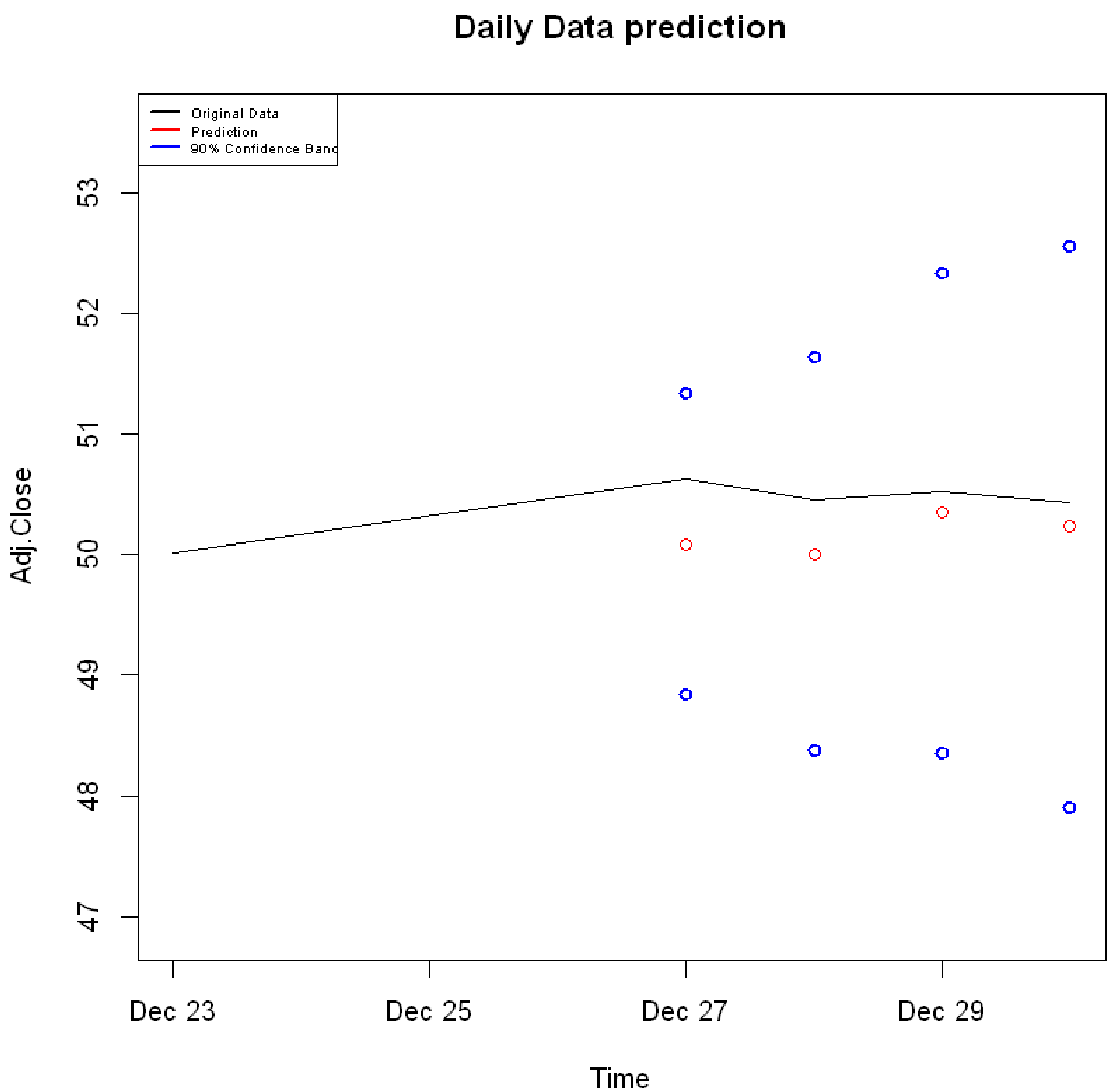


In [13]:

```
daily_model_pred = as.vector(predict(daily_AIC_model,n.ahead = 4))

daily_model_pred_ubound = daily_model_pred$pred+1.645*daily_model_pred$se
daily_model_pred_lbound = daily_model_pred$pred-1.645*daily_model_pred$se

ymin = min(daily_model_pred_lbound)-1
ymax = max(daily_model_pred_ubound)+1
plot( dailydata[(dim(dailydata)[1]-dailytestlimit):(dim(dailydata)[1]),],type="l", ylim
points(dailydata.test$Date,daily_model_pred$pred,col="red")
points(dailydata.test$Date,daily_model_pred_ubound,lty=3,lwd= 2, col="blue")
points(dailydata.test$Date,daily_model_pred_lbound,lty=3,lwd= 2, col="blue")
legend("topleft",legend = c("Original Data","Prediction","90% Confidence Band"),col = c
```



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.

In [14]:

```
### Mean Absolute Percentage Error (MAPE)
mean(abs(weekly_model_pred$pred-weeklydata.test$Adj.Close)/weeklydata.test$Adj.Close)
```

0.0125220223837661

In [15]:

```
### Mean Absolute Percentage Error (MAPE)
mean(abs(daily_model_pred$pred-dailydata.test$Adj.Close)/dailydata.test$Adj.Close)
### Precision Measure (PM)
sum((daily_model_pred$pred-dailydata.test$Adj.Close)^2)/sum((dailydata.test$Adj.Close-
```

0.00682175528041032

24.6219946989797

Response: Question 2d

All the observations are within the prediction bands for both the models. However, the model for Daily seems to be a better fit than Weekly qith lower MAPE. However, Daily model do not seem to provide good predictions since the variability in the predictions is much higher than the variability in the data (i.e. very high PM values).

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.

In [16]:

```
weeklytestlimit = 1
dailytestlimit = 4
weeklydata.train <- weeklydata[1:(dim(weeklydata)[1]-weeklytestlimit),]
weeklydata.test <- weeklydata[(dim(weeklydata)[1]-weeklytestlimit+1):(dim(weeklydata)[1
dailydata.train <- dailydata[1:(dim(dailydata)[1]-dailytestlimit),]
dailydata.test <- dailydata[(dim(dailydata)[1]-dailytestlimit+1):(dim(dailydata)[1]),]

## Step 1
# Weekly Data
weekly_AIC = data.frame(p=0, d=0, q=0, aic=0)
row_num = 1
norder=5
for(p in 1:norder)
{
    for(q in 1:norder)
    {
        possibleError <- tryCatch(
               arima(weeklydata.train$Adj.Close ,order = c(p,d,q), method='ML'),
                error=function(e) e
            )
            if(inherits(possibleError, "error"))
                {
                next
                }
            else
                {
                model_pdq = arima(weeklydata.train$Adj.Close ,order = c(p,d,q), method=
                weekly_AIC[row_num,] = c(p, d, q, AIC(model_pdq))
                row_num = row_num + 1
                }
    }
}
weekly_AIC_sorted = weekly_AIC[order(weekly_AIC$aic),]
head(weekly_AIC_sorted)
porder_weekly = weekly_AIC_sorted$p[1]
qorder_weekly = weekly_AIC_sorted$q[1]
dorder_weekly = weekly_AIC_sorted$d[1]
```

```r
weekly_AIC_model = arima(weeklydata.train$Adj.Close,order = c(porder_weekly,dorder_week
summary(weekly_AIC_model)

# daily Data
daily_AIC = data.frame(p=0, d=0, q=0, aic=0)
row_num = 1
norder=5
for(p in 1:norder)
{
    for(q in 1:norder)
    {
        possibleError <- tryCatch(
                 arima(dailydata.train$Adj.Close ,order = c(p,d,q), method='ML'),
                  error=function(e) e
            )
            if(inherits(possibleError, "error"))
                {
                next
                }
            else
                {
                model_pdq = arima(dailydata.train$Adj.Close ,order = c(p,d,q), method='
                daily_AIC[row_num,] = c(p, d, q, AIC(model_pdq))
                row_num = row_num + 1
                }
    }
}
daily_AIC_sorted = daily_AIC[order(daily_AIC$aic),]
head(daily_AIC_sorted)
porder_daily = daily_AIC_sorted$p[1]
qorder_daily = daily_AIC_sorted$q[1]
dorder_daily = daily_AIC_sorted$d[1]

daily_AIC_model = arima(dailydata.train$Adj.Close,order = c(porder_daily,dorder_daily,q
summary(daily_AIC_model)
```

	p	d	q	aic
25	5	2	5	1915.144
14	3	2	4	1917.165
15	3	2	5	1918.802
19	4	2	4	1919.818
20	4	2	5	1919.827
9	2	2	4	1920.783

```
          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 521    ts     numeric
call        4    -none- call
series      1    -none- character
```

```
code        1    -none- numeric
n.cond      1    -none- numeric
nobs        1    -none- numeric
model      10    -none- list
```

	p	d	q	aic
25	5	2	5	5799.195
20	4	2	5	5807.378
24	5	2	4	5829.088
19	4	2	4	5829.375
14	3	2	4	5832.378
12	3	2	2	5835.094

```
          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         4   -none- call
series       1   -none- character
code         1   -none- numeric
n.cond       1   -none- numeric
nobs         1   -none- numeric
model       10   -none- list
```

Model for Weekly ARMA is ARMA(4,2) for Daily is ARMA(5,5)

In [17]:

```
# Step2

#ARIMA-GARCH: Select GARCH order
test_modelAGG_weekly <- function(m,n){
    spec = ugarchspec(variance.model=list(garchOrder=c(m,n)),
                      mean.model=list(armaOrder=c(4,2),
                                      include.mean=T),
                      distribution.model="std")
    fit = ugarchfit(spec, weeklydata.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_weekly = data.frame(Inf,Inf,Inf)
names(ordersAGG_weekly) <- c("m","n","BIC")

for (m in 0:2){
    for (n in 0:2){
        possibleError <- tryCatch(
            ordersAGG_weekly<-rbind(ordersAGG_weekly,test_modelAGG_weekly(m,n)),
            error=function(e) e
        )
        if(inherits(possibleError, "error")) next
```

```
    }
}
ordersAGG_weekly <- ordersAGG_weekly[order(-ordersAGG_weekly$BIC),]
print('Weekly ARMA-GARCH model')
tail(ordersAGG_weekly)


#ARIMA-GARCH: Select GARCH order
test_modelAGG_daily <- function(m,n){
    spec = ugarchspec(variance.model=list(garchOrder=c(m,n)),
                      mean.model=list(armaOrder=c(4,2),
                                      include.mean=T),
                      distribution.model="std")
    fit = ugarchfit(spec, dailydata.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_daily = data.frame(Inf,Inf,Inf)
names(ordersAGG_daily) <- c("m","n","BIC")

for (m in 0:2){
    for (n in 0:2){
        possibleError <- tryCatch(
            ordersAGG_daily<-rbind(ordersAGG_daily,test_modelAGG_daily(m,n)),
            error=function(e) e
        )
        if(inherits(possibleError, "error")) next
    }
}
ordersAGG_daily <- ordersAGG_daily[order(-ordersAGG_daily$BIC),]
print('Daily ARMA-GARCH model')
tail(ordersAGG_daily)
```

```
[1] "Weekly ARMA-GARCH model"
```

	m	n	BIC
9	2	2	-3.73247
5	1	1	-10.30271
8	2	1	-23.02389
7	2	0	-27.05229
3	0	2	-27.42776
4	1	0	-31.18361

```
[1] "Daily ARMA-GARCH model"
```

	m	n	BIC
4	1	0	1.5515375
3	0	2	1.5478252
8	2	1	1.4973809

	m	n	BIC
5	1	1	1.3757919
6	1	2	1.3715827
7	2	0	0.9565355

order for GARCH is identified as (1,0) for Weekly GARCH moderl and (2,1) for Daily GARCH Model

In [18]:

```
#Step 3
test_modelAGA_weekly <- function(p,q){
    spec = ugarchspec(variance.model=list(garchOrder=c(1,0)),
                      mean.model=list(armaOrder=c(p,q),
                                      include.mean=T),
                      distribution.model="std")
    fit = ugarchfit(spec,weeklydata.train, solver = 'hybrid')
    current.bic = infocriteria(fit)[2]
    df = data.frame(p,q,current.bic)
    names(df) <- c("p","q","BIC")
    #print(paste(p,q,current.bic,sep=" "))
    return(df)
}

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


#ARIMA-GARCH: Select ARIMA order
test_modelAGA_daily <- function(p,q){
    spec = ugarchspec(variance.model=list(garchOrder=c(2,1)),
                      mean.model=list(armaOrder=c(p,q),
                                      include.mean=T),
                      distribution.model="std")
    fit = ugarchfit(spec,dailydata.train, solver = 'hybrid')
    current.bic = infocriteria(fit)[2]
    df = data.frame(p,q,current.bic)
    names(df) <- c("p","q","BIC")
    #print(paste(p,q,current.bic,sep=" "))
    return(df)
}

ordersAGA_daily = data.frame(Inf,Inf,Inf)
names(ordersAGA_daily) <- c("p","q","BIC")
for (p in 0:4){
    for (q in 0:4){
        possibleError <- tryCatch(
```

```
            ordersAGA_daily<-rbind(ordersAGA_daily,test_modelAGA_daily(p,q)),
            error=function(e) e
        )
        if(inherits(possibleError, "error")) next
    }
}
ordersAGA_daily <- ordersAGA_daily[order(-ordersAGA_daily$BIC),]
tail(ordersAGA_daily)
```

	p	q	BIC
23	3	3	-32.12705
29	4	3	-33.06057
21	3	1	-33.97342
32	5	0	-34.59086
20	3	0	-35.04167
26	4	0	-36.51770

	p	q	BIC
15	2	4	1.423858
12	2	0	1.418037
22	4	2	1.360754
18	3	2	1.356882
7	1	0	1.039764
10	1	3	-1.231702

Updated orders for ARMA models are identified as (1,0) for Weekly GARCH model and (2,1) for Daily GARCH Model

In [19]:

```
# Step 4
test_modelAGG_Weekly <- function(m,n){
    spec = ugarchspec(variance.model=list(garchOrder=c(m,n)),
                      mean.model=list(armaOrder=c(3,0),
                                      include.mean=T), distribution.model="std")
    fit = ugarchfit(spec, weeklydata.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_weekly = data.frame(Inf,Inf,Inf)
names(ordersAGG_weekly) <- c("m","n","BIC")

for (m in 0:2){
    for (n in 0:2){
        possibleError <- tryCatch(
            ordersAGG_weekly<-rbind(ordersAGG_weekly,test_modelAGG_Weekly(m,n)),
            error=function(e) e
```

```
        )
        if(inherits(possibleError, "error")) next
    }
}
ordersAGG_weekly <- ordersAGG_weekly[order(-ordersAGG_weekly$BIC),]
tail(ordersAGG_weekly)


test_modelAGG_daily <- function(m,n){
    spec = ugarchspec(variance.model=list(garchOrder=c(m,n)),
                      mean.model=list(armaOrder=c(3,2),
                                      include.mean=T), distribution.model="std")
    fit = ugarchfit(spec, dailydata.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_daily = data.frame(Inf,Inf,Inf)
names(ordersAGG_daily) <- c("m","n","BIC")

for (m in 0:2){
    for (n in 0:2){
        possibleError <- tryCatch(
            ordersAGG_daily<-rbind(ordersAGG_daily,test_modelAGG_daily(m,n)),
            error=function(e) e
        )
        if(inherits(possibleError, "error")) next
    }
}
ordersAGG_daily <- ordersAGG_daily[order(-ordersAGG_daily$BIC),]
tail(ordersAGG_daily)
```

	m	n	BIC
4	1	0	-13.04820
9	2	2	-35.13181
6	1	2	-36.61734
7	2	0	-36.62937
3	0	2	-36.63199
2	0	1	-36.64400

	m	n	BIC
2	0	1	1.385517
6	1	2	1.377236
3	0	2	1.371882
5	1	1	1.371041
9	2	2	1.359985
8	2	1	1.356315

In [35]:
```
weekly_fit = ugarchspec(variance.model=list(garchOrder=c(1,1)),
                 mean.model=list(armaOrder=c(0, 1),
                 include.mean=T), distribution.model="std")
weekly_AIC_model_res = ugarchfit(weekly_fit, weeklydata.train, solver = 'hybrid')

daily_fit = ugarchspec(variance.model=list(garchOrder=c(2,1)),
                 mean.model=list(armaOrder=c(3, 2),
                 include.mean=T), distribution.model="std")
daily_AIC_model_res = ugarchfit(daily_fit, dailydata.train, solver = 'hybrid')

print('Weekly ARMA-GARCH MODEL')
weekly_fit
print('Daily ARMA-GARCH MODEL')
daily_fit
```

```
[1] "Weekly ARMA-GARCH MODEL"
*---------------------------------*
*       GARCH Model Spec          *
*---------------------------------*

Conditional Variance Dynamics
------------------------------------
GARCH Model             : sGARCH(1,1)
Variance Targeting      : FALSE

Conditional Mean Dynamics
------------------------------------
Mean Model              : ARFIMA(0,0,1)
Include Mean            : TRUE
GARCH-in-Mean           : FALSE

Conditional Distribution
------------------------------------
Distribution    :  std
Includes Skew   :  FALSE
Includes Shape  :  TRUE
Includes Lambda :  FALSE
[1] "Daily ARMA-GARCH MODEL"
*---------------------------------*
*       GARCH Model Spec          *
*---------------------------------*

Conditional Variance Dynamics
------------------------------------
GARCH Model             : sGARCH(2,1)
Variance Targeting      : FALSE

Conditional Mean Dynamics
------------------------------------
Mean Model              : ARFIMA(3,0,2)
Include Mean            : TRUE
GARCH-in-Mean           : FALSE

Conditional Distribution
------------------------------------
Distribution    :  std
Includes Skew   :  FALSE
Includes Shape  :  TRUE
Includes Lambda :  FALSE
```

Response: Question 3a

Final ARMA GARCH Models identified as

Weekly Model ARMA(3,0) - GARCH(0,2)

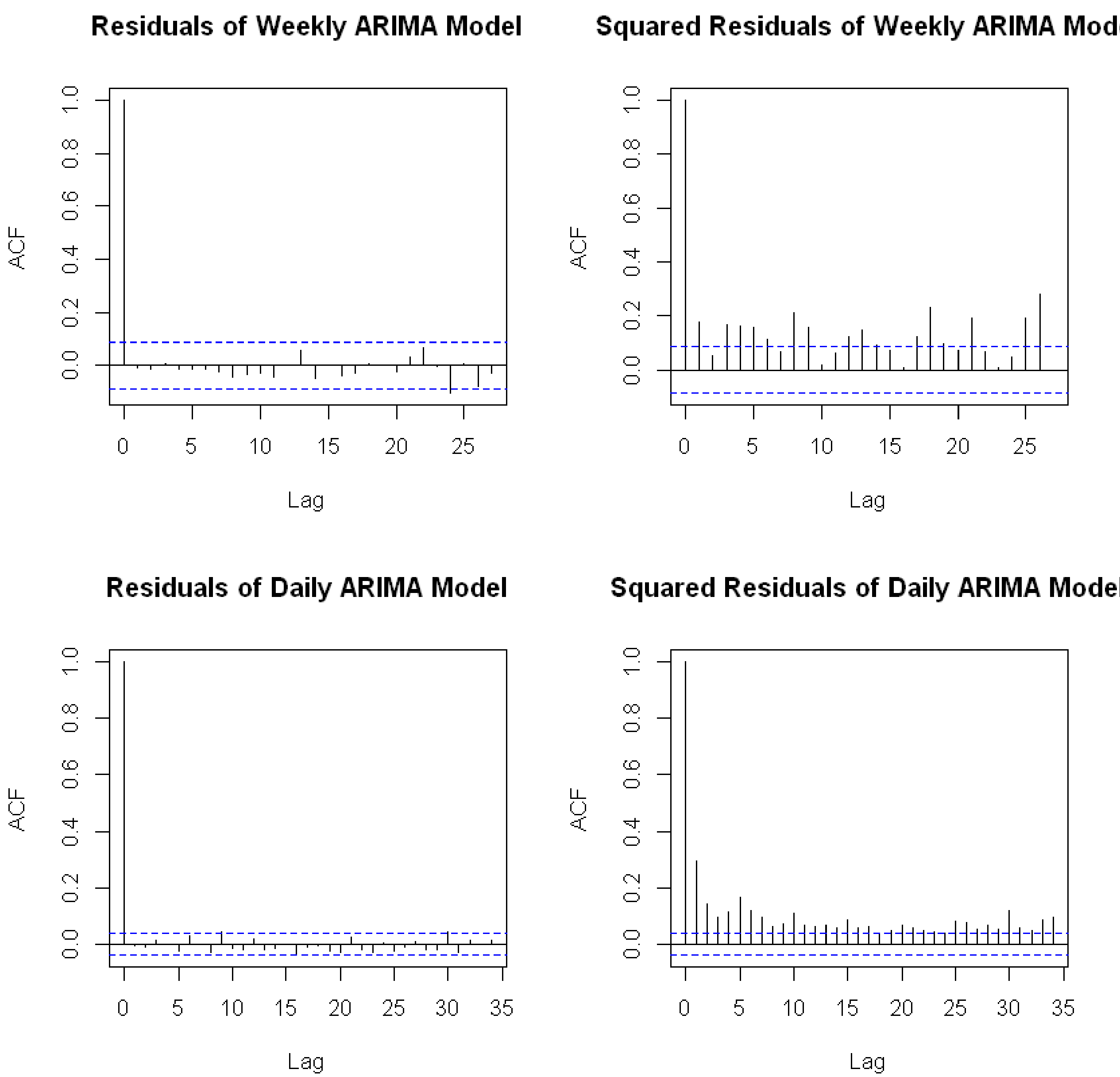
Daily Model ARMA(3,2) - GARCH(2,1)

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?

In [36]:

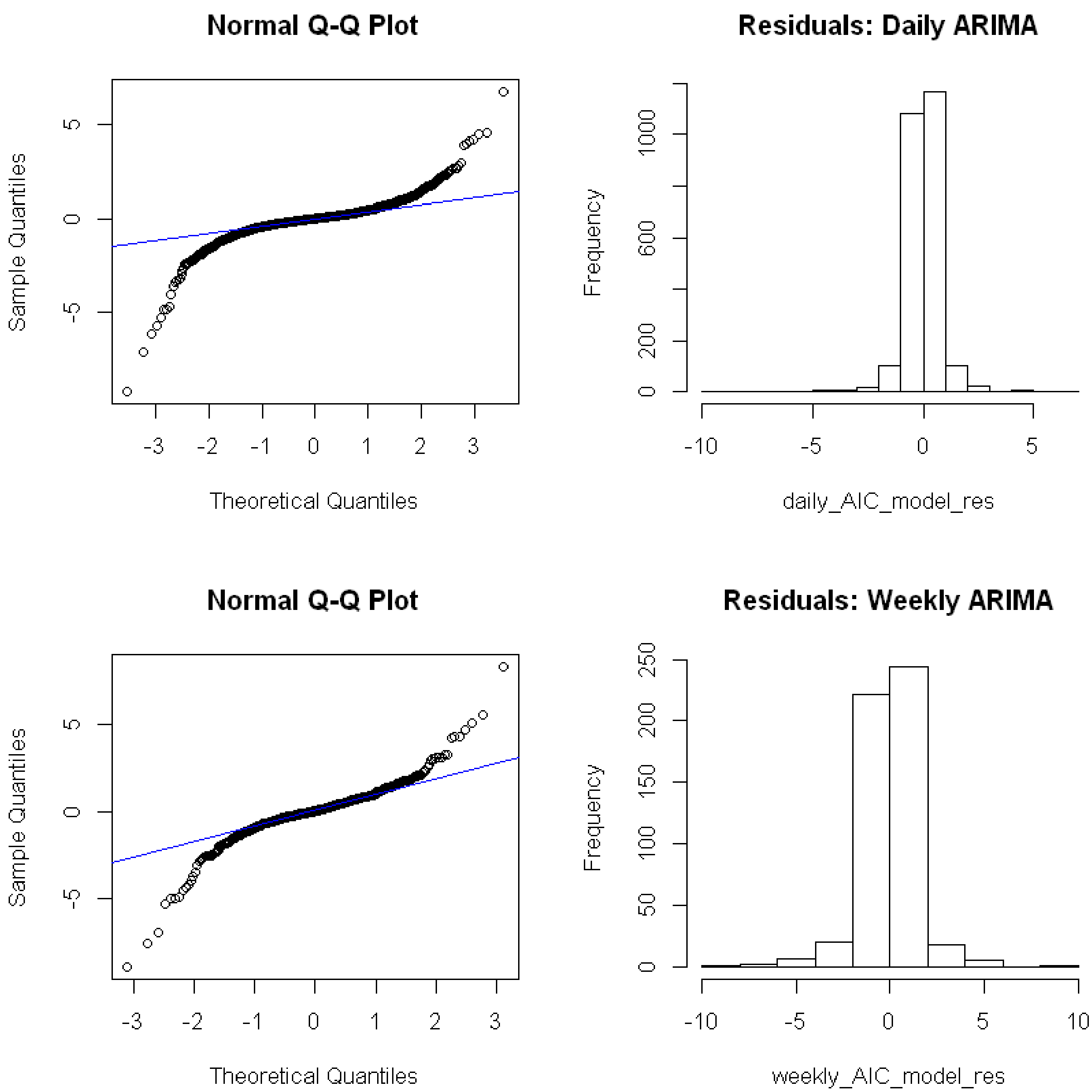
```
weekly_AIC_model_res = weekly_AIC_model$residuals
daily_AIC_model_res = daily_AIC_model$residuals

par(mfrow=c(2,2))
acf(weekly_AIC_model_res,main="Residuals of Weekly ARIMA Model")
acf(weekly_AIC_model_res^2,main="Squared Residuals of Weekly ARIMA Model")
acf(daily_AIC_model_res,main="Residuals of Daily ARIMA Model")
acf(daily_AIC_model_res^2,main="Squared Residuals of Daily ARIMA Model")
```



In [37]:

```
par(mfrow=c(2,2))
qqnorm(daily_AIC_model_res)
qqline(daily_AIC_model_res, col="blue")
hist(daily_AIC_model_res,main="Residuals: Daily ARIMA")
qqnorm(weekly_AIC_model_res)
qqline(weekly_AIC_model_res, col="blue")
hist(weekly_AIC_model_res,main="Residuals: Weekly ARIMA")
```



In [44]:

```
# test for serial correlation in residuals
Box.test(daily_AIC_model_res,lag=9,type='Ljung',fitdf=8)
Box.test(weekly_AIC_model_res,lag=3,type='Ljung',fitdf=2)

# test for serial correlation in squared residuals
Box.test((daily_AIC_model_res)^2,lag=9,type='Ljung',fitdf=8)
Box.test((weekly_AIC_model_res)^2,lag=3,type='Ljung',fitdf=2)
```

```
	Box-Ljung test
```

```
data:  daily_AIC_model_res
X-squared = 11.055, df = 1, p-value = 0.0008844
        Box-Ljung test

data:  weekly_AIC_model_res
X-squared = 0.14148, df = 1, p-value = 0.7068
        Box-Ljung test

data:  (daily_AIC_model_res)^2
X-squared = 469.15, df = 1, p-value < 2.2e-16
        Box-Ljung test

data:  (weekly_AIC_model_res)^2
X-squared = 32.389, df = 1, p-value = 1.262e-08
```

Response: Question 3b

Daily Residuals: • visually they do no appear to be normally distributed. The mean appears constant but the variance changes over time. Histogram and QQ plot doesnot appear nearly normally distributed.

Weekly Residuals: • visually they appear to be very close to being normally distributed. Mean and variance appear to be constant. Histogram and QQ plot appears nearly normally distributed.

The p-values for testing uncorrelated residuals are very small for Daily data(showing correlation) but is high for Weekly data showing non-correlation. However, the ACF plots from the previous slide look similarly to those of white noise.

The p-values for testing uncorrelated squared residuals are also very small indicating that we reject the null hypothesis, and thus, conclude that the squared residuals are correlated.

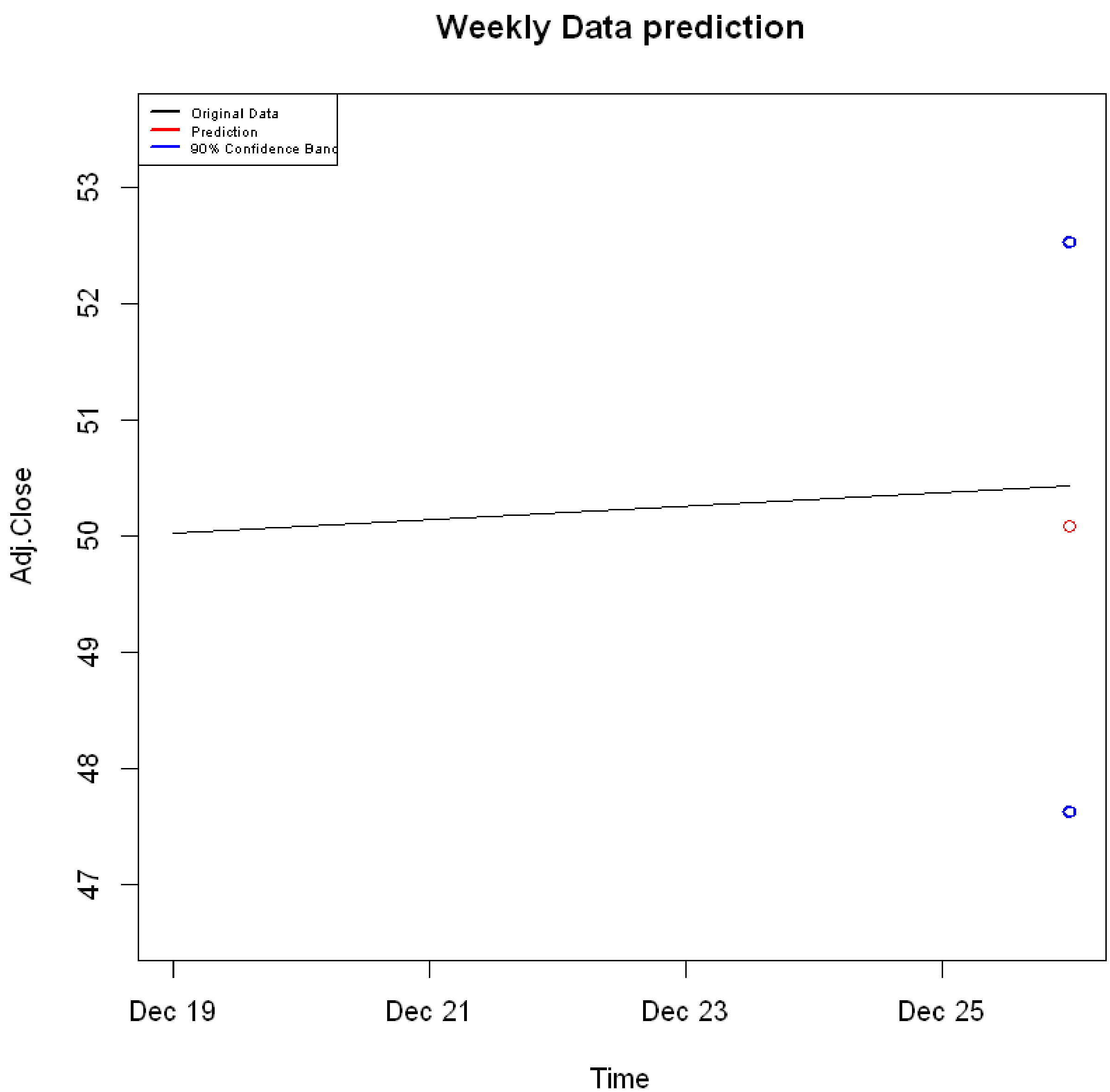
3c. Apply the model identified in (3a) and forecast the mean and the variance of the last week of data. Plot the predicted data to compare the predicted values to the actual observed ones. Interpret the results, particularly comparing forecast using daily versus weekly data.

In [38]:

```
weekly_model_pred = as.vector(predict(weekly_AIC_model,n.ahead = 1))

weekly_model_pred_ubound = weekly_model_pred$pred+1.645*weekly_model_pred$se
weekly_model_pred_lbound = weekly_model_pred$pred-1.645*weekly_model_pred$se

ymin = min(weekly_model_pred_lbound)-1
ymax = max(weekly_model_pred_ubound)+1
plot( weeklydata[(dim(weeklydata)[1]-weeklytestlimit):(dim(weeklydata)[1]),],type="l",
points(weeklydata.test$Date,weekly_model_pred$pred,col="red")
points(weeklydata.test$Date,weekly_model_pred_ubound,lty=3,lwd= 2, col="blue")
points(weeklydata.test$Date,weekly_model_pred_lbound,lty=3,lwd= 2, col="blue")
legend("topleft",legend = c("Original Data","Prediction","90% Confidence Band"),col = c
```

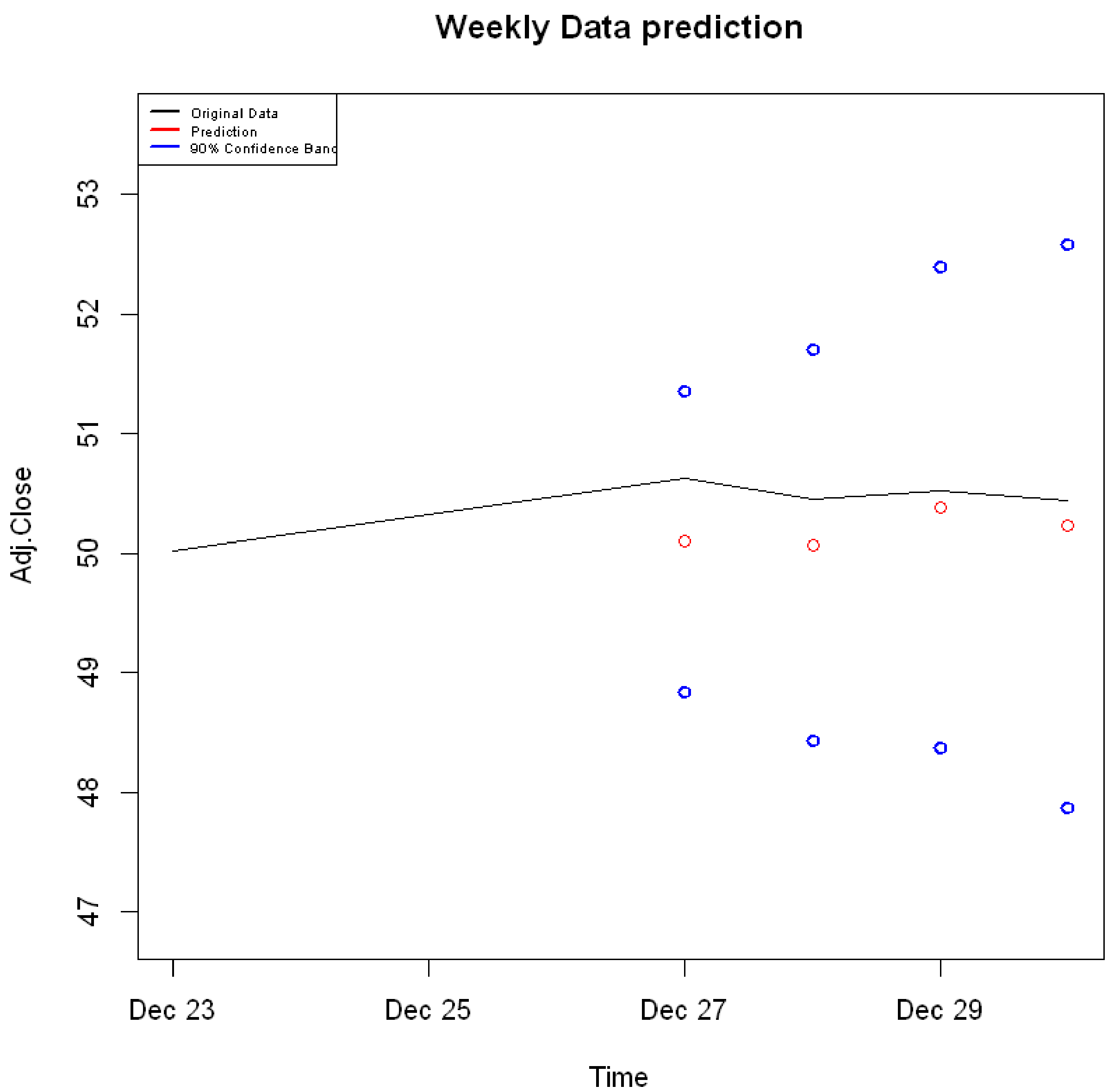


In [39]:

```
daily_model_pred = as.vector(predict(daily_AIC_model,n.ahead = 4))

daily_model_pred_ubound = daily_model_pred$pred+1.645*daily_model_pred$se
daily_model_pred_lbound = daily_model_pred$pred-1.645*daily_model_pred$se

ymin = min(daily_model_pred_lbound)-1
ymax = max(daily_model_pred_ubound)+1
plot( dailydata[(dim(dailydata)[1]-dailytestlimit):(dim(dailydata)[1]),],type="l", ylim
points(dailydata.test$Date,daily_model_pred$pred,col="red")
points(dailydata.test$Date,daily_model_pred_ubound,lty=3,lwd= 2, col="blue")
points(dailydata.test$Date,daily_model_pred_lbound,lty=3,lwd= 2, col="blue")
legend("topleft",legend = c("Original Data","Prediction","90% Confidence Band"),col = c
```



In [40]:

```
# test for serial correlation in residuals
Box.test(daily_AIC_model_res,lag=9,type='Ljung',fitdf=8)
Box.test(weekly_AIC_model_res,lag=3,type='Ljung',fitdf=2)

# test for serial correlation in squared residuals
Box.test((daily_AIC_model_res)^2,lag=9,type='Ljung',fitdf=8)
Box.test((weekly_AIC_model_res)^2,lag=3,type='Ljung',fitdf=2)
```

```
        Box-Ljung test

data:  daily_AIC_model_res
X-squared = 11.055, df = 1, p-value = 0.0008844
        Box-Ljung test

data:  weekly_AIC_model_res
X-squared = 0.14148, df = 1, p-value = 0.7068
        Box-Ljung test

data:  (daily_AIC_model_res)^2
X-squared = 469.15, df = 1, p-value < 2.2e-16
```

```
	Box-Ljung test

data:  (weekly_AIC_model_res)^2
X-squared = 32.389, df = 1, p-value = 1.262e-08
```

Response: Question 3c

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.

In [41]:

```
### Mean Absolute Percentage Error (MAPE)
mean(abs(weekly_model_pred$pred-weeklydata.test$Adj.Close)/weeklydata.test$Adj.Close)
### Precision Measure (PM)
sum((weekly_model_pred$pred-weeklydata.test$Adj.Close)^2)
```

0.00716167579029646

0.130474876351845

In [42]:

```
### Mean Absolute Percentage Error (MAPE)
mean(abs(daily_model_pred$pred-dailydata.test$Adj.Close)/dailydata.test$Adj.Close)
```

0.00630466934286482

Response: Question 3d

All the observations are within the prediction bands for both the models. However, the model for Daily seems to be a better fit than Weekly qith lower MAPE. Also, Daily model do not seem to provide good predictions since little variability in the predictions is observed variability in the data (i.e. low PM values).

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

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 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?

Response: Question 4

Here are my view points ont he analysis done so far

How do the models fit the data? How well do the models predict?

As per models all 4 are relatively good predictions but could be made better. I think ARMA(p,q)-GARCH(m,n) models did better than higher order ARIMA(p,d,q) as we can wee the MAPE of Weekly ARMA_GARCH model increased by 100 times better than weekly ARIMA model. Also PM for Daily Data is reduced by 100 time with usage of ARMA-GARCH Method

How do the models perform when using daily versus weekly data?

After detail analysis done so far DAILY ARMA-GARCH is the better of all. Also we could see Daily models of both ARIMA and ARMA-GARCCH performing better than weekly model

Would you use one approach over another for different settings?

Definately, as discussed in my initial section I would definately use weekly trends for putting long call option or trading a stock by holding it for long time.

What are some specific points of caution one would need to consider when applying those models?

One caution is that even by applying ARMA-GARCH model for weekly trend I still was not convinced with residuals. So a better or different model(EGARCH,APARCH,iGARCH) would likely solve it.

Also compution time is major as we could see there is not much better prediction(based on MAPE values) for ARIMA vs AMA-GARCH models. which makes me think that we need to take all to consideration while modelling a solution

Also, considering outlier events like pandemic, Geopolitical events, Severe Wweather are important to model the solution along nature of data itself

In [ ]:
```

```