Quantitative Finance Project

José Thiéry M. Hagbe

15 février, 2024

Contents

1. Download APPLE daily data from Yahoo Finance with prices for one stock from 2019-01-01, to 2024-01-01.

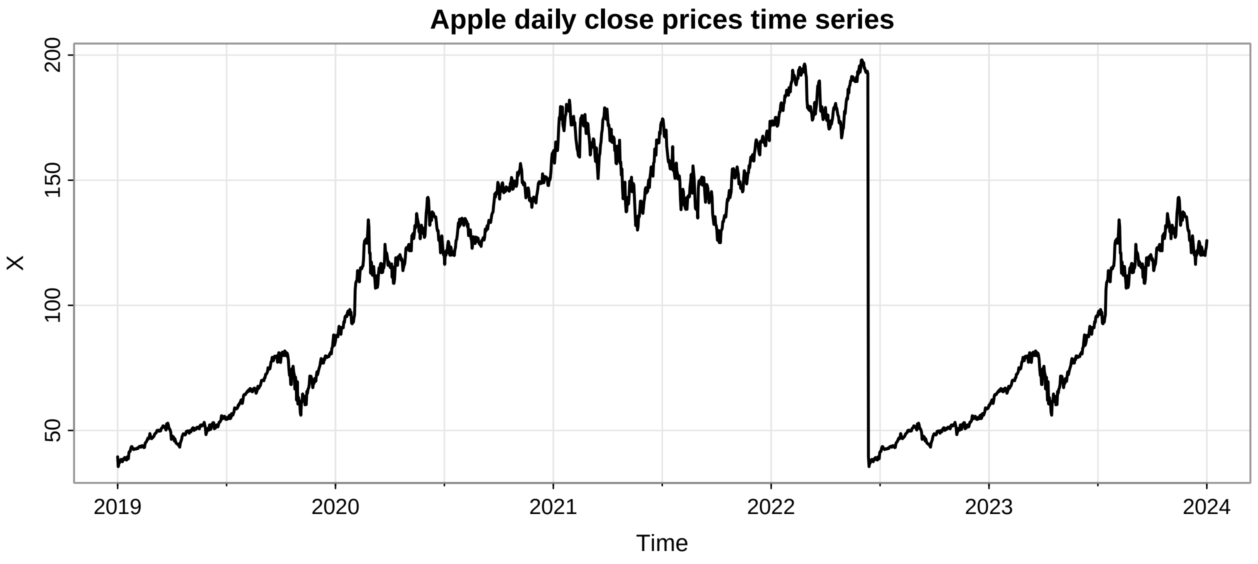
```r
library(quantmod)
# Download APPLE daily data from Yahoo Finance
getSymbols("AAPL", src = "yahoo", from = "2019-01-01", to = "2024-01-01")
```

```
## [1] "AAPL"
```

```r
Daily <- `AAPL`
colnames(Daily) <- c("Open", "High", "Low", "Close", "Volume", "Adjusted")
```

2. Display the time series plot of the daily close data. Based on this information, do these data appear to come from a stationary or nonstationary process?

```r
# Create daily close prices time series
X <- ts(data = Daily$Close, start = c(2019, 1, 1), end =  c(2024, 1, 1), frequency = 365)

# Create daily close prices time series plot
tsplot(X, main = "Apple daily close prices time series", lwd = 2)
```



Comment: It appeared that these data are coming from nonstationary process

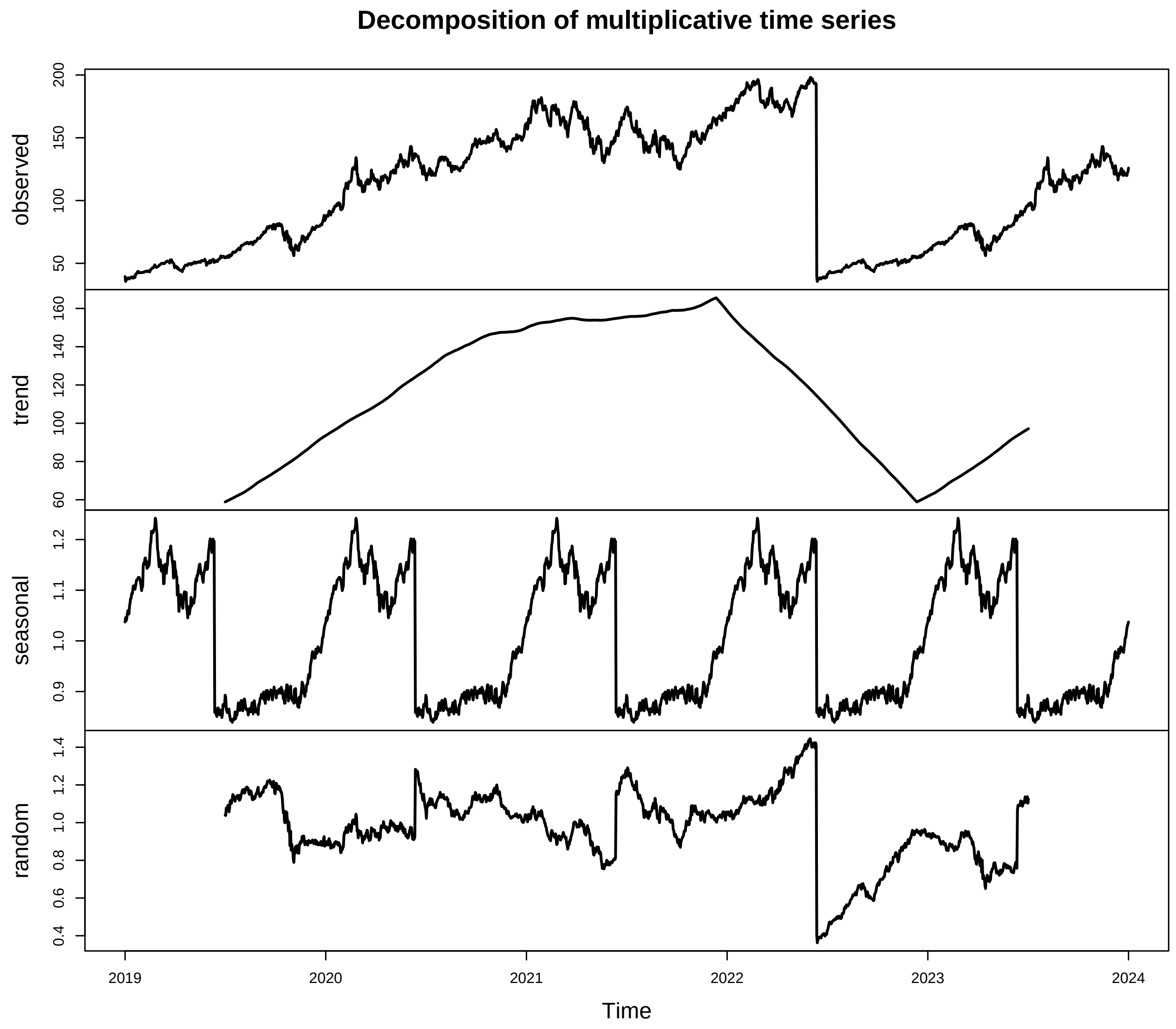
3. Use the best subsets ARIMA approach to specify a model for the daily close data.

Decomposing the Data

Decomposing the data into its trend, seasonal, and random error components will give some idea how these components relate to the observed dataset.

```r
# Decomposition using multiplicative type
X.decomp <- decompose(X, type = "multiplicative")
```

```r
plot(X.decomp, lwd=2)
```



Summary Statistics of the Data

```r
summary(X)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   35.55   64.31  117.51  109.58  148.48  198.11
```

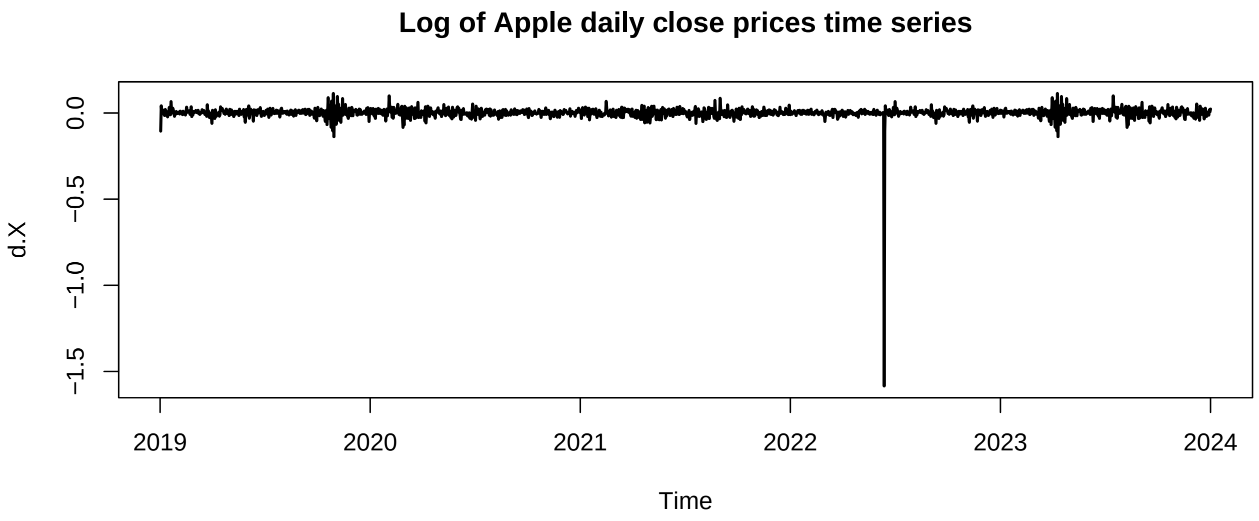
```r
describe(X)
```

```
##    vars    n   mean   sd median trimmed   mad   min    max  range skew kurtosis
## X1    1 1826 109.58 46.8 117.51  108.65 63.62 35.55 198.11 162.56 0.03    -1.34
##     se
## X1 1.1
```

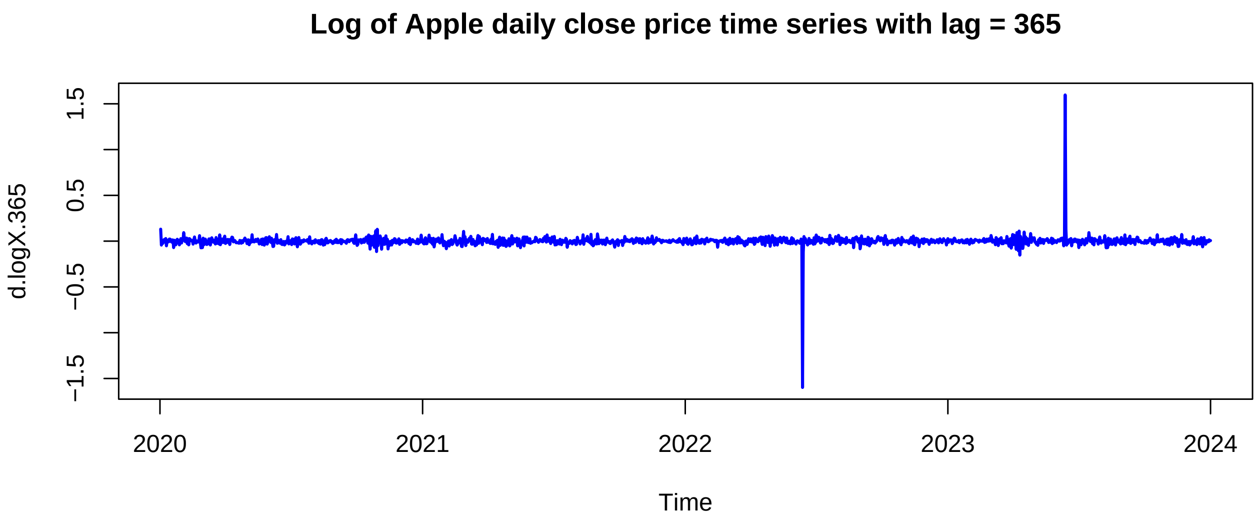
Data Transformation To Achieve Stationarity

Now, we will have to perform some data transformation to achieve Stationarity.

```r
# Remove the trend from the data
# First log Difference
d.X = diff(log(X)) # to remove trend
plot(d.X, main = "Log of Apple daily close prices time series", lwd=2)
```



```r
d.logX.365 = diff(d.X, lag = 365)  # remove seasonality
plot(d.logX.365, main =" Log of Apple daily close price time series with lag = 365 ", lwd=2, col="blue"
```



```r
# test for stationarity
adf.test(d.logX.365)
```

```
## Warning in adf.test(d.logX.365): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  d.logX.365
## Dickey-Fuller = -10.636, Lag order = 11, p-value = 0.01
## alternative hypothesis: stationary
```

Comment: We see that the series is stationary enough to do any kind of time series modelling.

Identification of the best model

```r
# Model Selection
set.seed(123)
auto.arima(d.logX.365, stationary = TRUE, ic = c("aic"), trace = TRUE)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,0,2)(1,0,1)[365] with non-zero mean : Inf
##  ARIMA(0,0,0)             with non-zero mean : -3800.64
##  ARIMA(1,0,0)(1,0,0)[365] with non-zero mean : Inf
##  ARIMA(0,0,1)(0,0,1)[365] with non-zero mean : Inf
##  ARIMA(0,0,0)             with zero mean     : -3802.639
##  ARIMA(0,0,0)(1,0,0)[365] with non-zero mean : Inf
##  ARIMA(0,0,0)(0,0,1)[365] with non-zero mean : Inf
##  ARIMA(0,0,0)(1,0,1)[365] with non-zero mean : Inf
##  ARIMA(1,0,0)             with non-zero mean : -3804.81
##  ARIMA(1,0,0)(0,0,1)[365] with non-zero mean : Inf
##  ARIMA(1,0,0)(1,0,1)[365] with non-zero mean : Inf
##  ARIMA(2,0,0)             with non-zero mean : -3805.562
##  ARIMA(2,0,0)(1,0,0)[365] with non-zero mean : Inf
##  ARIMA(2,0,0)(0,0,1)[365] with non-zero mean : Inf
##  ARIMA(2,0,0)(1,0,1)[365] with non-zero mean : Inf
##  ARIMA(3,0,0)             with non-zero mean : -3802.814
##  ARIMA(2,0,1)             with non-zero mean : -3804.158
##  ARIMA(1,0,1)             with non-zero mean : -3806.506
##  ARIMA(1,0,1)(1,0,0)[365] with non-zero mean : Inf
##  ARIMA(1,0,1)(0,0,1)[365] with non-zero mean : Inf
##  ARIMA(1,0,1)(1,0,1)[365] with non-zero mean : Inf
##  ARIMA(0,0,1)             with non-zero mean : -3802.139
##  ARIMA(1,0,2)             with non-zero mean : -3805.091
##  ARIMA(0,0,2)             with non-zero mean : -3803.142
##  ARIMA(2,0,2)             with non-zero mean : -3802.123
##  ARIMA(1,0,1)             with zero mean     : -3808.5
##  ARIMA(1,0,1)(1,0,0)[365] with zero mean     : Inf
##  ARIMA(1,0,1)(0,0,1)[365] with zero mean     : Inf
##  ARIMA(1,0,1)(1,0,1)[365] with zero mean     : Inf
##  ARIMA(0,0,1)             with zero mean     : -3804.139
##  ARIMA(1,0,0)             with zero mean     : -3806.806
##  ARIMA(2,0,1)             with zero mean     : -3806.112
##  ARIMA(1,0,2)             with zero mean     : -3807.087
##  ARIMA(0,0,2)             with zero mean     : -3805.142
##  ARIMA(2,0,0)             with zero mean     : -3807.56
##  ARIMA(2,0,2)             with zero mean     : -3804.178
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(1,0,1)             with zero mean     : -3805.768
##
##  Best model: ARIMA(1,0,1)             with zero mean

## Series: d.logX.365
## ARIMA(1,0,1) with zero mean
##
## Coefficients:
##           ar1     ma1
##       -0.6791  0.7307
## s.e.   0.1993  0.1859
##
## sigma^2 = 0.004308:  log likelihood = 1905.88
## AIC=-3805.77   AICc=-3805.75   BIC=-3789.91
```

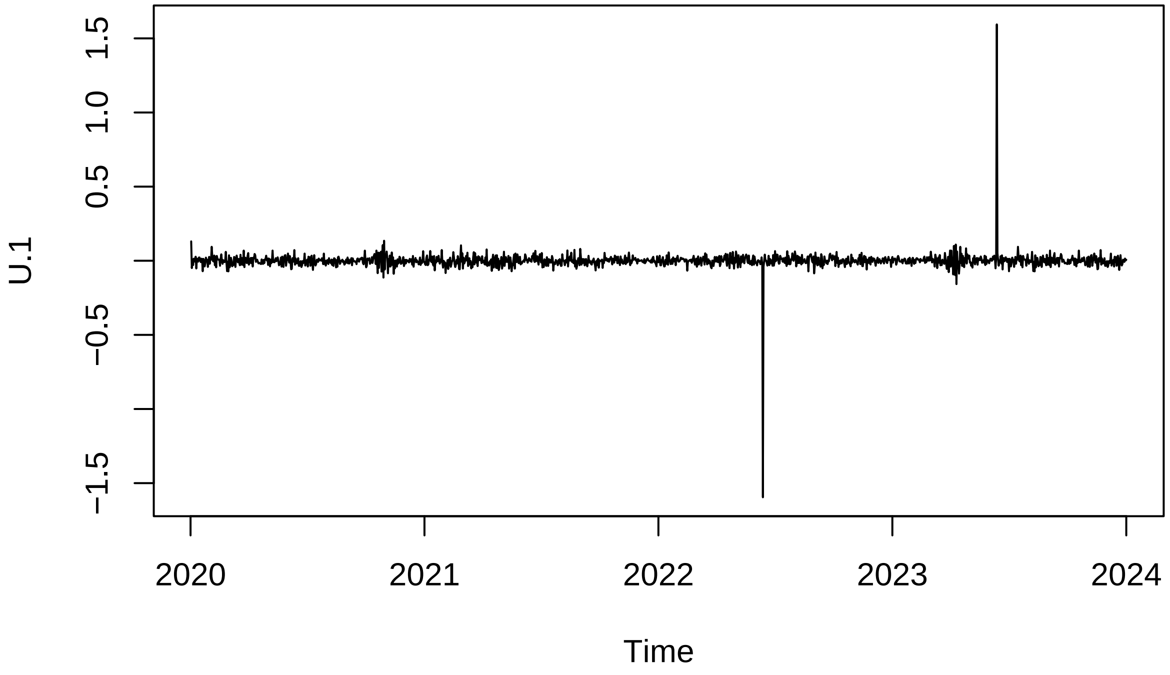
Comment: auto arima indicates us that the best arima model is ARIMA(1,0,1)

Daily Model

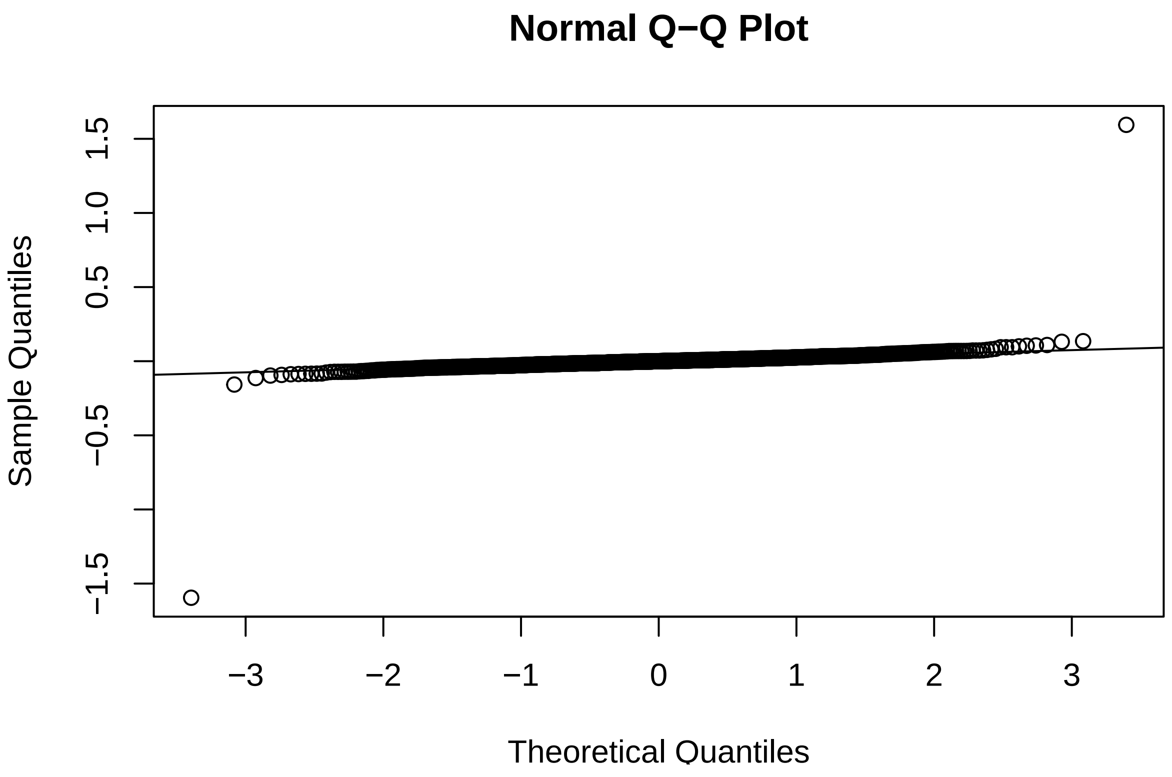
```
## Setting up the Model
Daily.model <- arima(d.logX.365, order = c(1,0,1))
```

Model evaluation

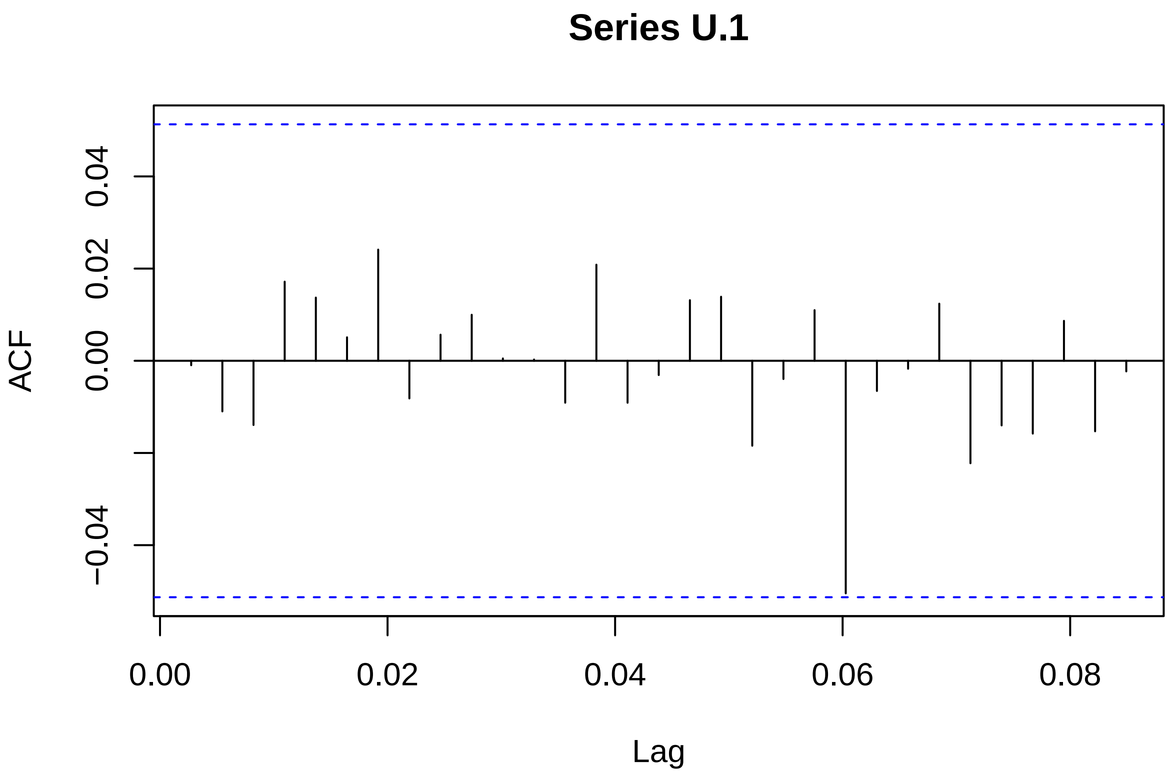
```
## Diagnostic Checking
U.1 <- Daily.model$residuals
plot(U.1)
```



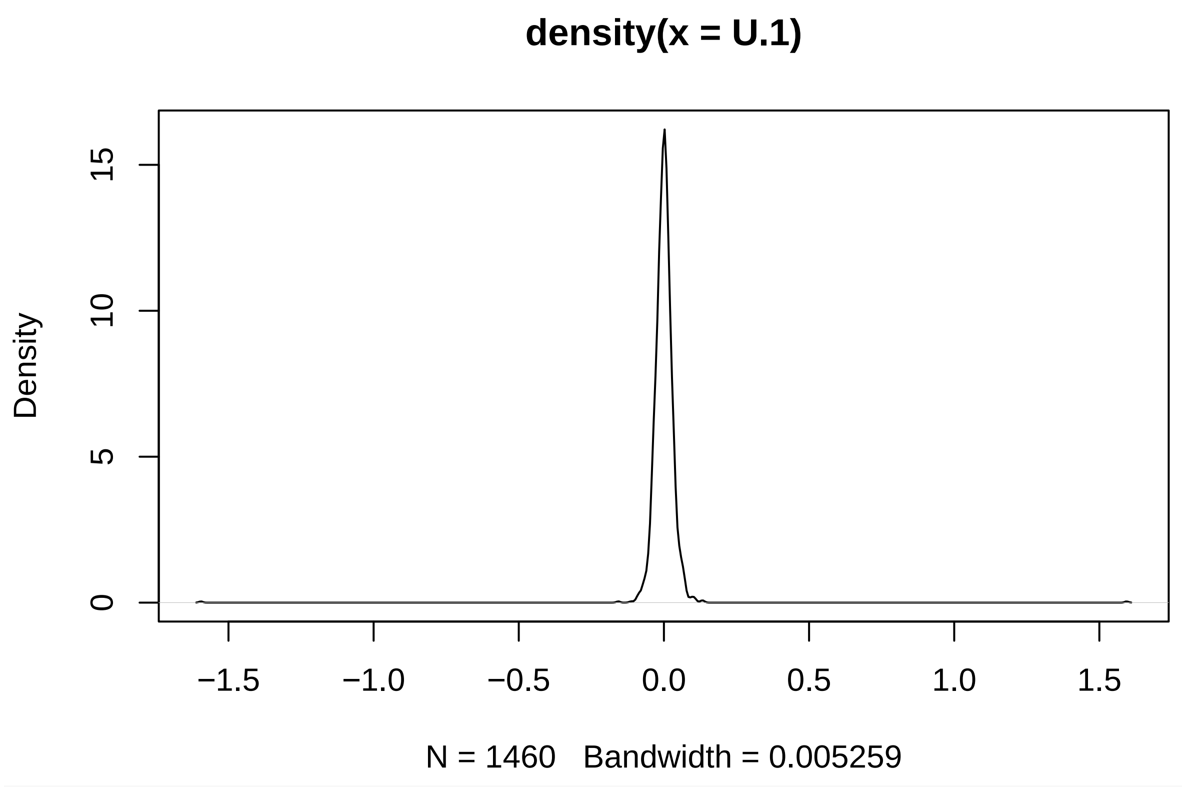
```
# Normality of the Residuals
qqnorm(U.1); qqline(U.1)
```



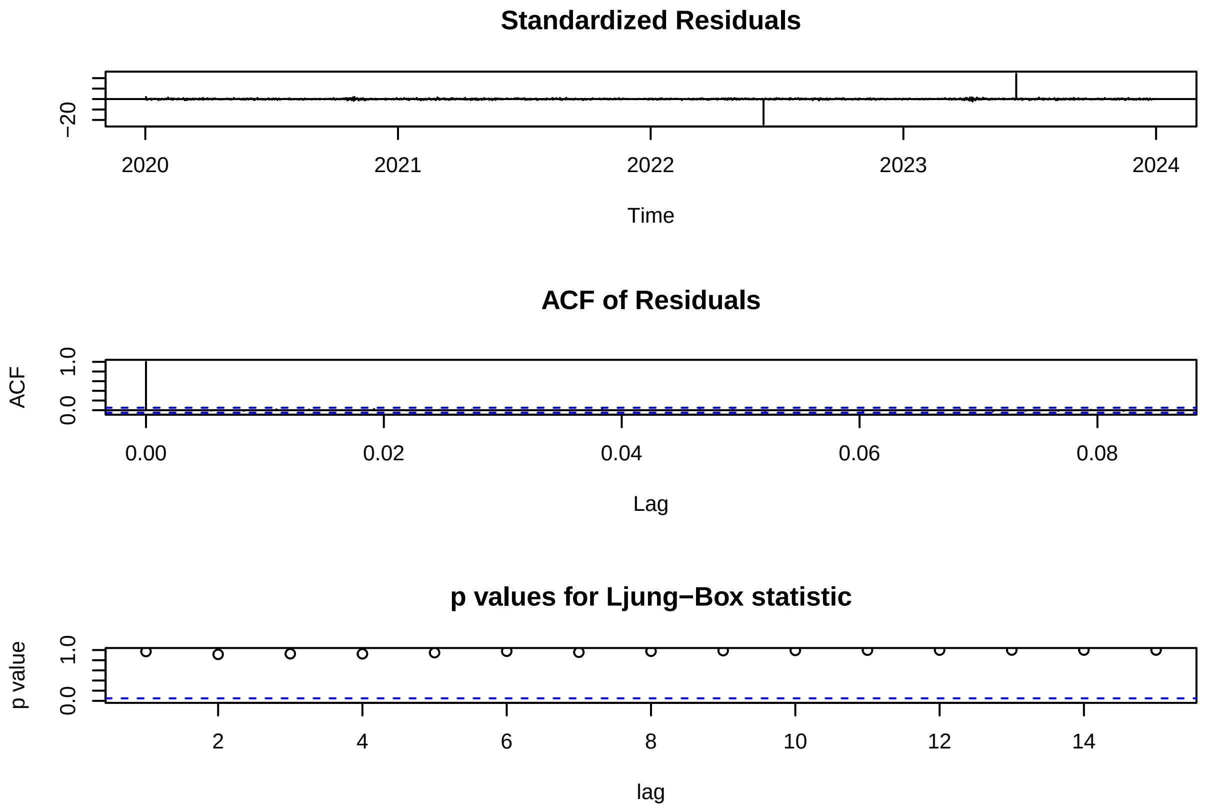
```r
# ACF of residuals
acf(U.1)
```



```r
plot(density(U.1))
```



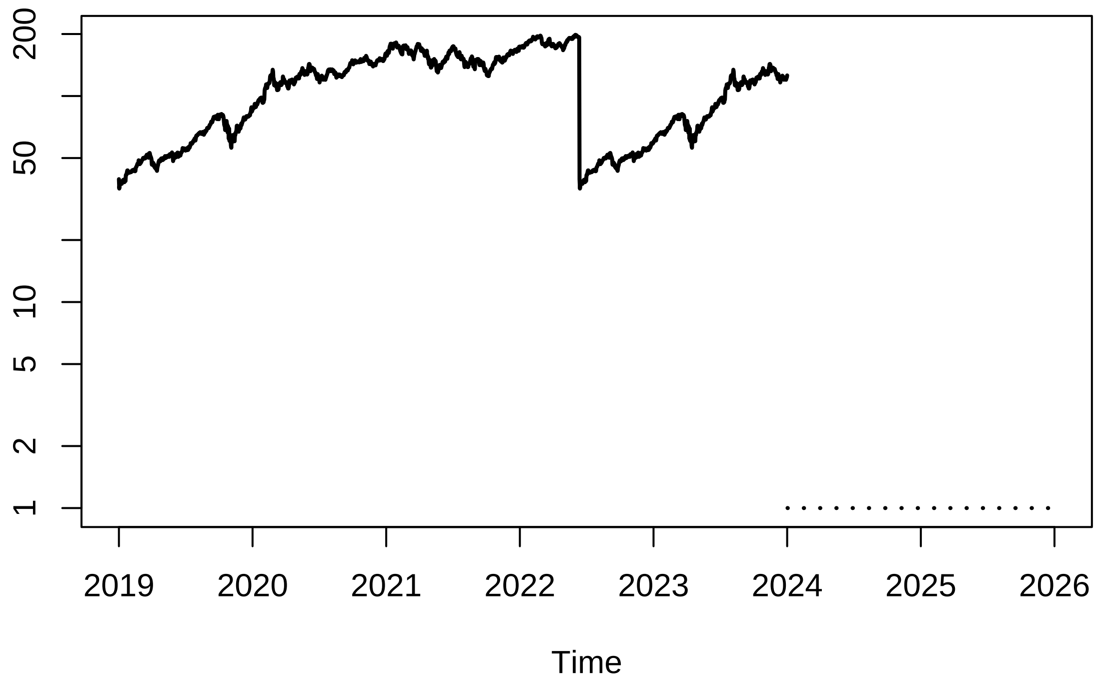
```r
#Using tsdiag tools
tsdiag(Daily.model, gof.lag = 15)
```



Forecasting

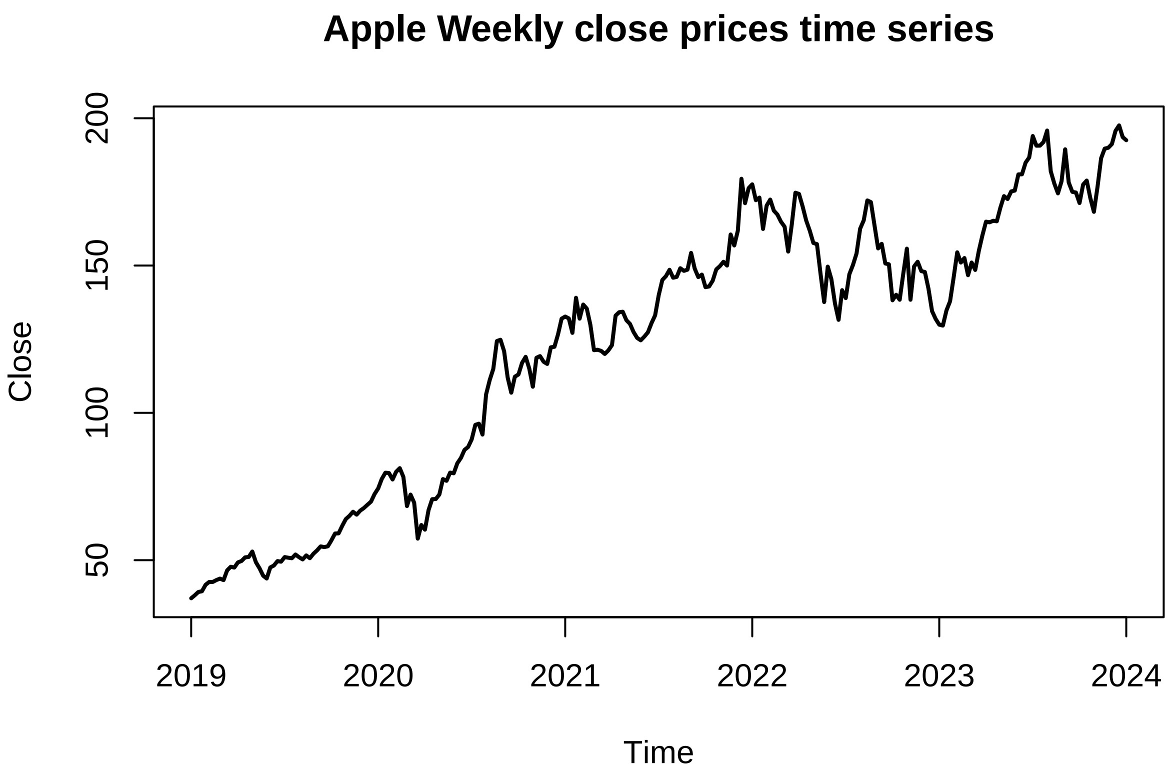
Using our best model, we can forecast the observations for the next 2 years.

```
Daily.pred <- predict(Daily.model, n.ahead = 2*365)
ts.plot(X, exp(Daily.pred$pred), log = "y", lty =c(1,3), lwd = 2)
```



4. Use the best subsets ARIMA approach to specify a model for the weekly close data.

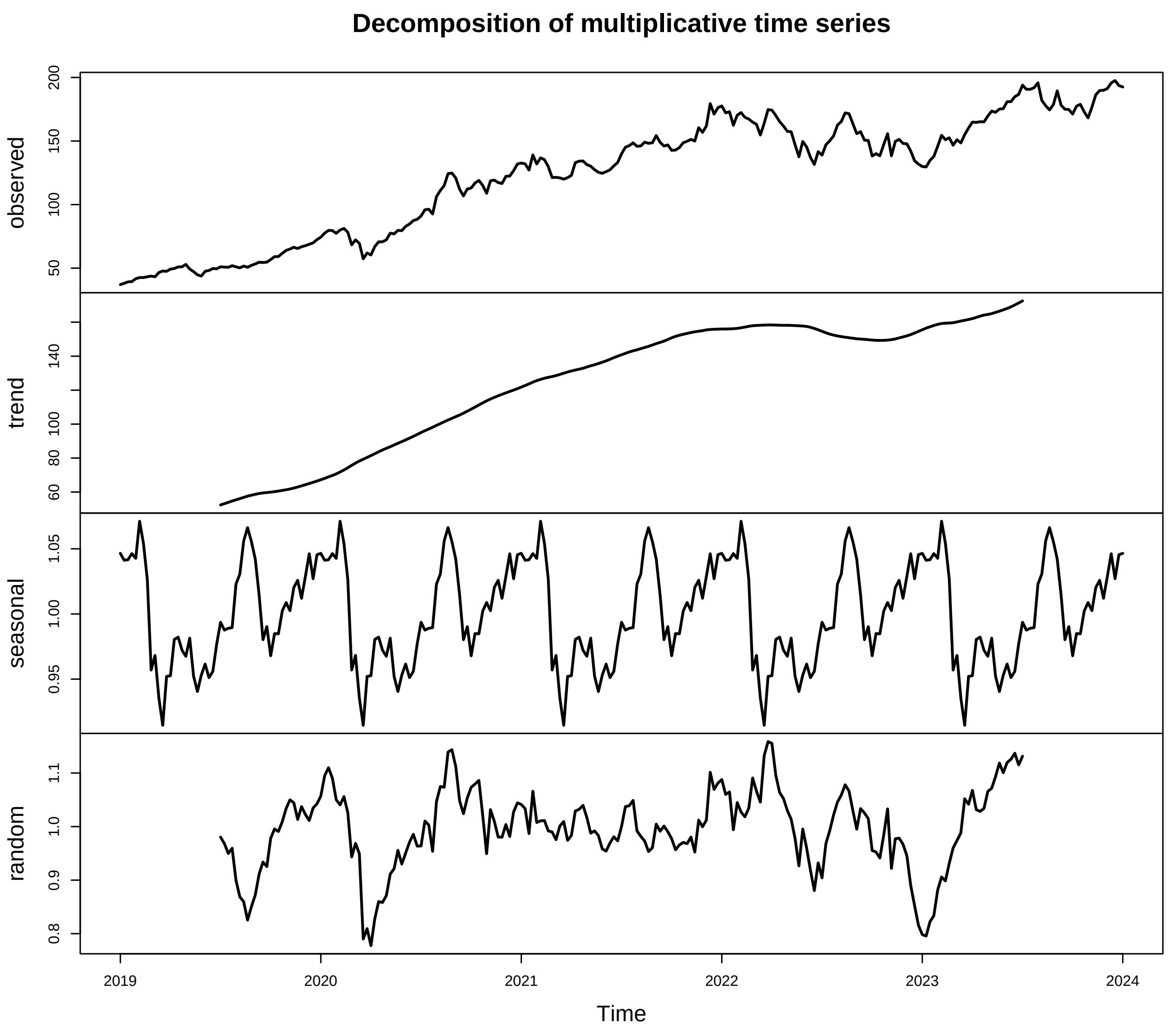
```r
# Aggregate daily data to weekly
Weekly <- to.weekly(Daily)
colnames(Weekly) <- c("Open", "High", "Low", "Close", "Volume", "Adjusted")

# Create weekly close prices time series
Y <- ts(data = Weekly$Close, start = c(2019, 1, 1), end =  c(2024, 1, 1), frequency = 52)

# Create weekly close prices time series
plot(Y, main="Apple Weekly close prices time series", lwd = 2)
```



Decomposing the Data

Decomposing the data into its trend, seasonal, and random error components will give some idea how these components relate to the observed dataset.

```r
# Decomposition using multiplicative type
Y.decomp <- decompose(Y, type = "multiplicative")
plot(Y.decomp, lwd=2)
```



Summary Statistics of the Data

```
summary(Y)
```

```
##      Close       
##  Min.   : 37.06  
##  1st Qu.: 77.53  
##  Median :134.32  
##  Mean   :123.06  
##  3rd Qu.:160.55  
##  Max.   :197.57
```

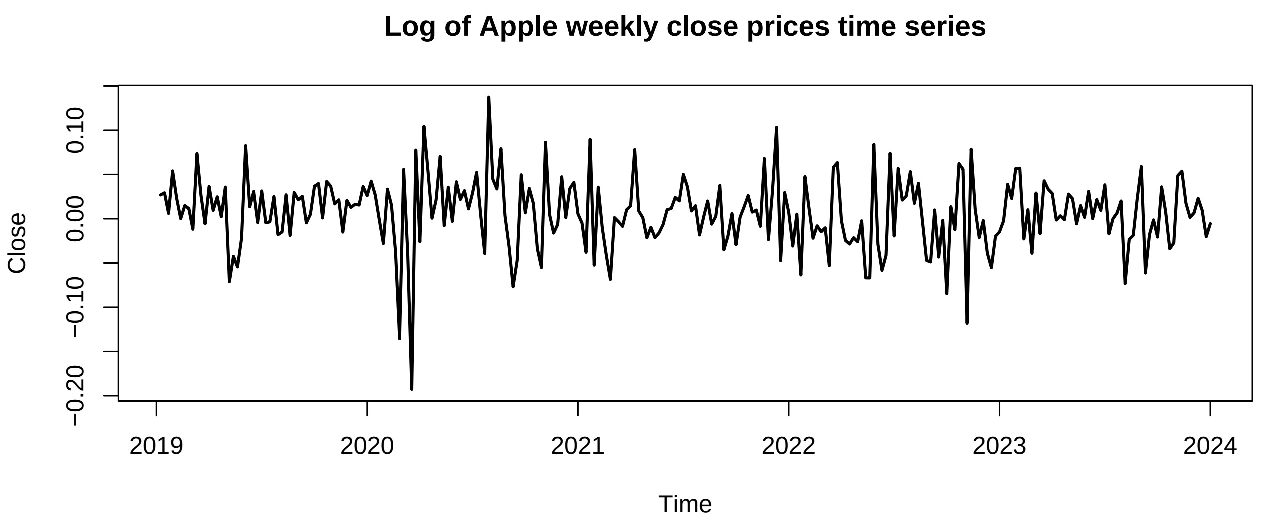
```
describe(Y)
```

```
##    vars   n   mean    sd median trimmed   mad   min    max  range skew kurtosis
## X1    1 261 123.06 46.64 134.32   124.8 50.88 37.06 197.57 160.51 -0.4    -1.16
##      se
## X1 2.89
```

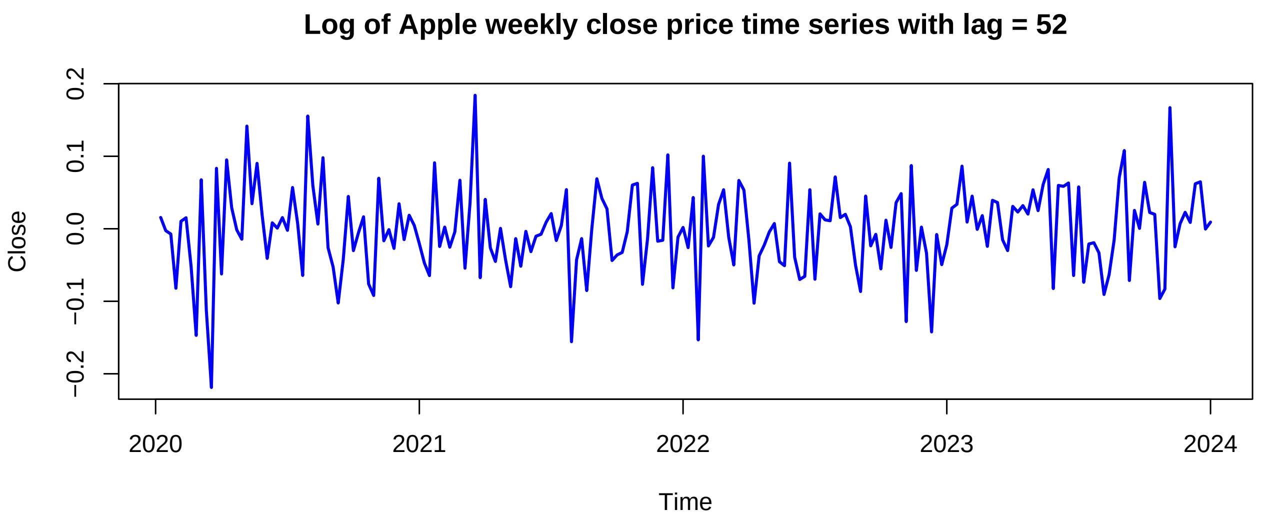
Data Transformation To Achieve Stationarity

Now, we will have to perform some data transformation to achieve Stationarity.

```
# Remove the trend from the data
# First log Difference
d.Y = diff(log(Y)) # to remove trend
plot(d.Y, main = "Log of Apple weekly close prices time series", lwd=2)
```



```
d.logY.52 = diff(d.Y, lag = 52)  # remove seasonality
plot(d.logY.52, main =" Log of Apple weekly close price time series with lag = 52 ", lwd=2, col="blue")
```



```
# test for stationarity
adf.test(d.logY.52)
```

```
## Warning in adf.test(d.logY.52): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  d.logY.52
## Dickey-Fuller = -5.5748, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

Comment: We see that the series is stationary enough to do any kind of time series modelling.

Identification of the best model

```
# Model Selection
set.seed(123)
auto.arima(d.logY.52, stationary = TRUE, ic = c("aic"), trace = TRUE)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,0,2)(1,0,1)[52] with non-zero mean : Inf
##  ARIMA(0,0,0)            with non-zero mean : -580.0943
##  ARIMA(1,0,0)(1,0,0)[52] with non-zero mean : -674.1046
##  ARIMA(0,0,1)(0,0,1)[52] with non-zero mean : Inf
##  ARIMA(0,0,0)            with zero mean     : -581.9694
##  ARIMA(1,0,0)            with non-zero mean : -578.5567
##  ARIMA(1,0,0)(1,0,1)[52] with non-zero mean : -678.1467
##  ARIMA(1,0,0)(0,0,1)[52] with non-zero mean : Inf
##  ARIMA(0,0,0)(1,0,1)[52] with non-zero mean : -678.5859
##  ARIMA(0,0,0)(0,0,1)[52] with non-zero mean : Inf
##  ARIMA(0,0,0)(1,0,0)[52] with non-zero mean : -674.561
##  ARIMA(0,0,1)(1,0,1)[52] with non-zero mean : -678.4059
##  ARIMA(1,0,1)(1,0,1)[52] with non-zero mean : -676.8726
##  ARIMA(0,0,0)(1,0,1)[52] with zero mean     : -677.6064
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(0,0,0)(1,0,1)[52] with non-zero mean : Inf
##  ARIMA(0,0,1)(1,0,1)[52] with non-zero mean : Inf
##  ARIMA(1,0,0)(1,0,1)[52] with non-zero mean : Inf
##  ARIMA(0,0,0)(1,0,1)[52] with zero mean     : Inf
##  ARIMA(1,0,1)(1,0,1)[52] with non-zero mean : Inf
##  ARIMA(0,0,0)(1,0,0)[52] with non-zero mean : -634.7971
##
##  Best model: ARIMA(0,0,0)(1,0,0)[52] with non-zero mean
```

```
## Series: d.logY.52
## ARIMA(0,0,0)(1,0,0)[52] with non-zero mean
##
## Coefficients:
##          sar1     mean
##       -0.5661  -0.0030
## s.e.   0.0612   0.0024
##
## sigma^2 = 0.002465:  log likelihood = 320.4
## AIC=-634.8   AICc=-634.68   BIC=-624.78
```

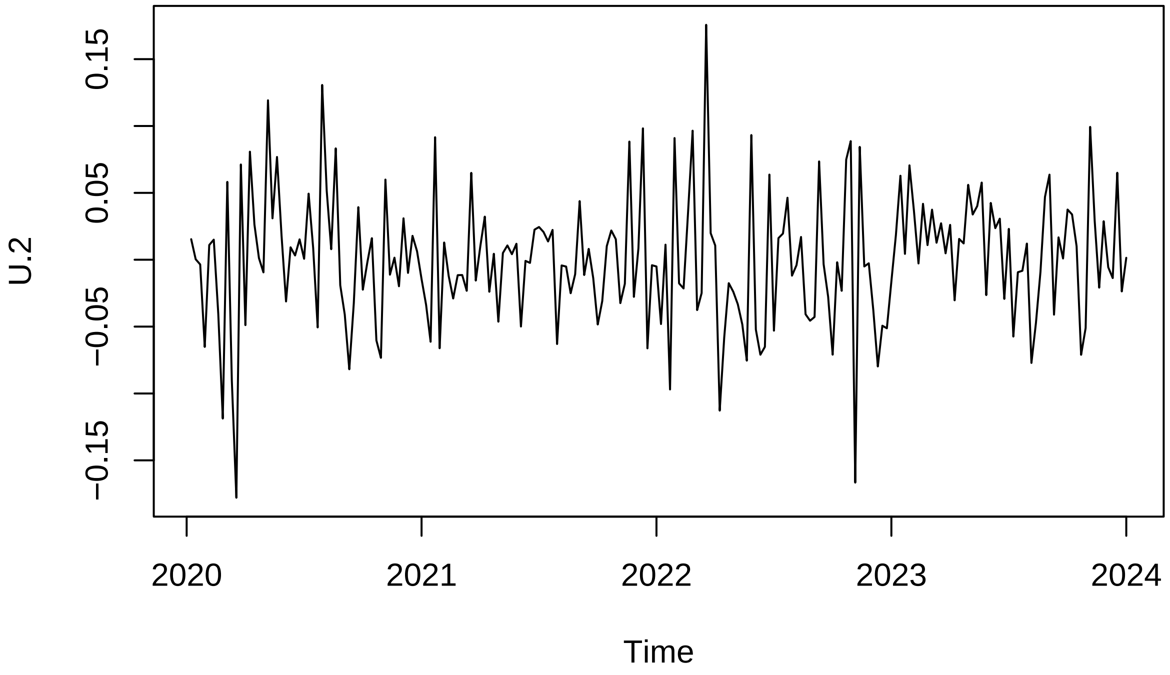
Comment: auto arima indicates us that the best arima model is ARIMA(0,0,0)(1,0,0)[52]

Weekly Model

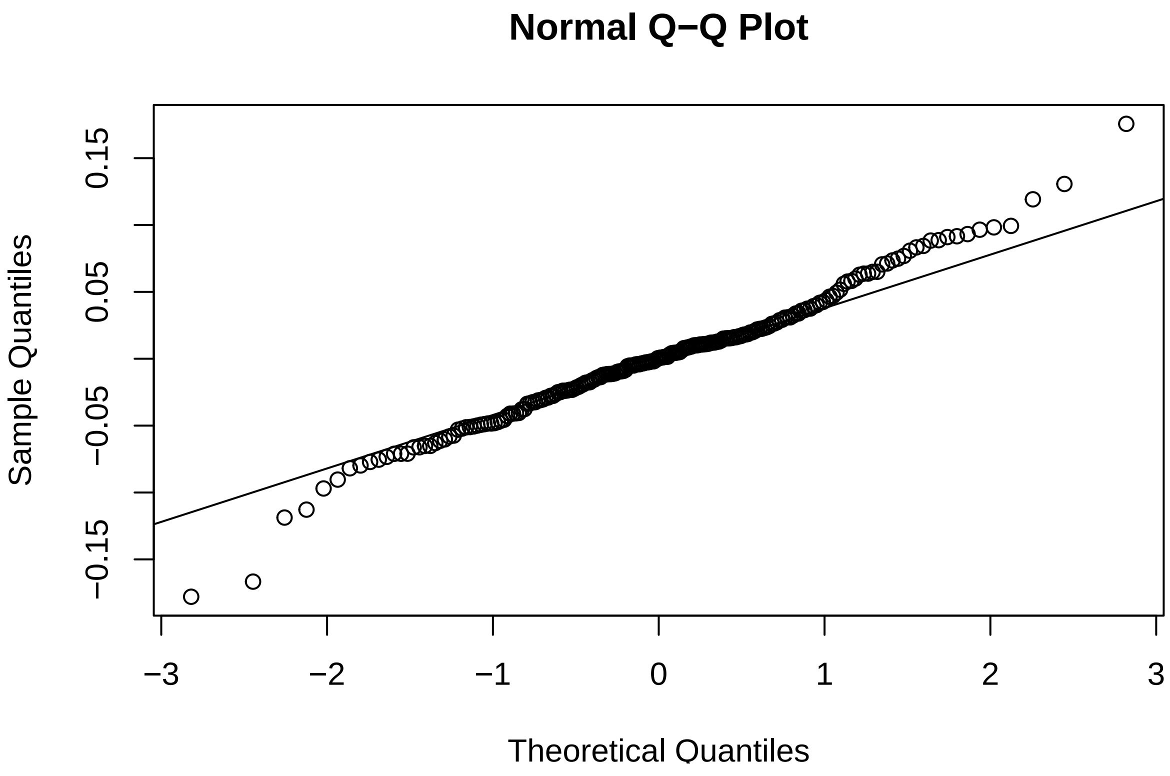
```
## Setting up the Model
Weekly.model <- arima(d.logY.52, c(0, 0, 0) , seasonal = list(order = c(1, 0, 0), period = 52))
```

Model evaluation

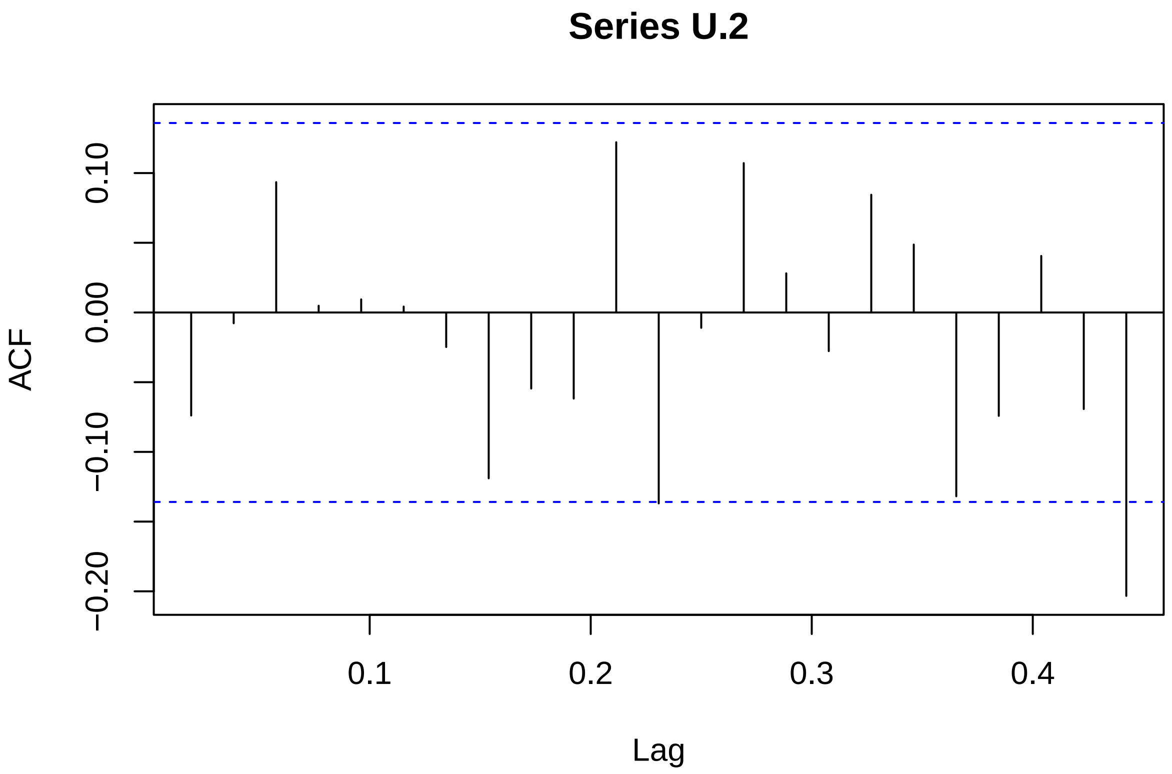
```r
## Diagnostic Checking
U.2 <- Weekly.model$residuals
plot(U.2)
```



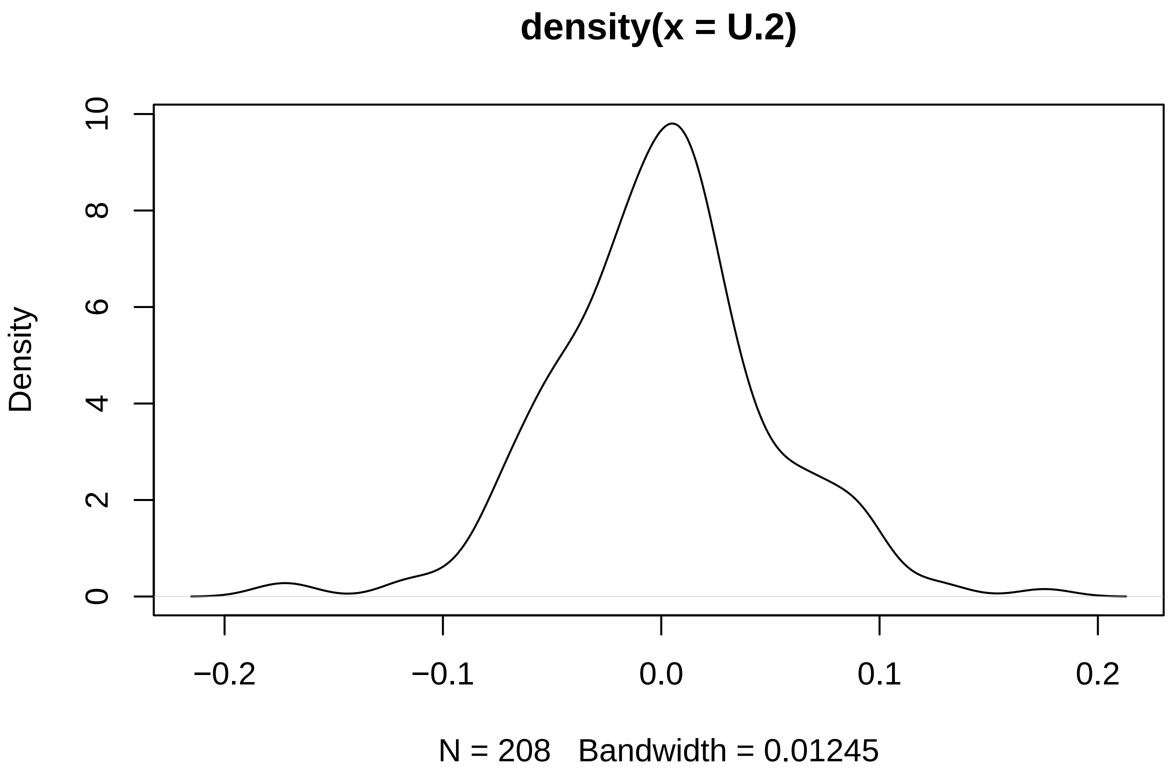
```r
# Normality of the Residuals
qqnorm(U.2); qqline(U.2)
```



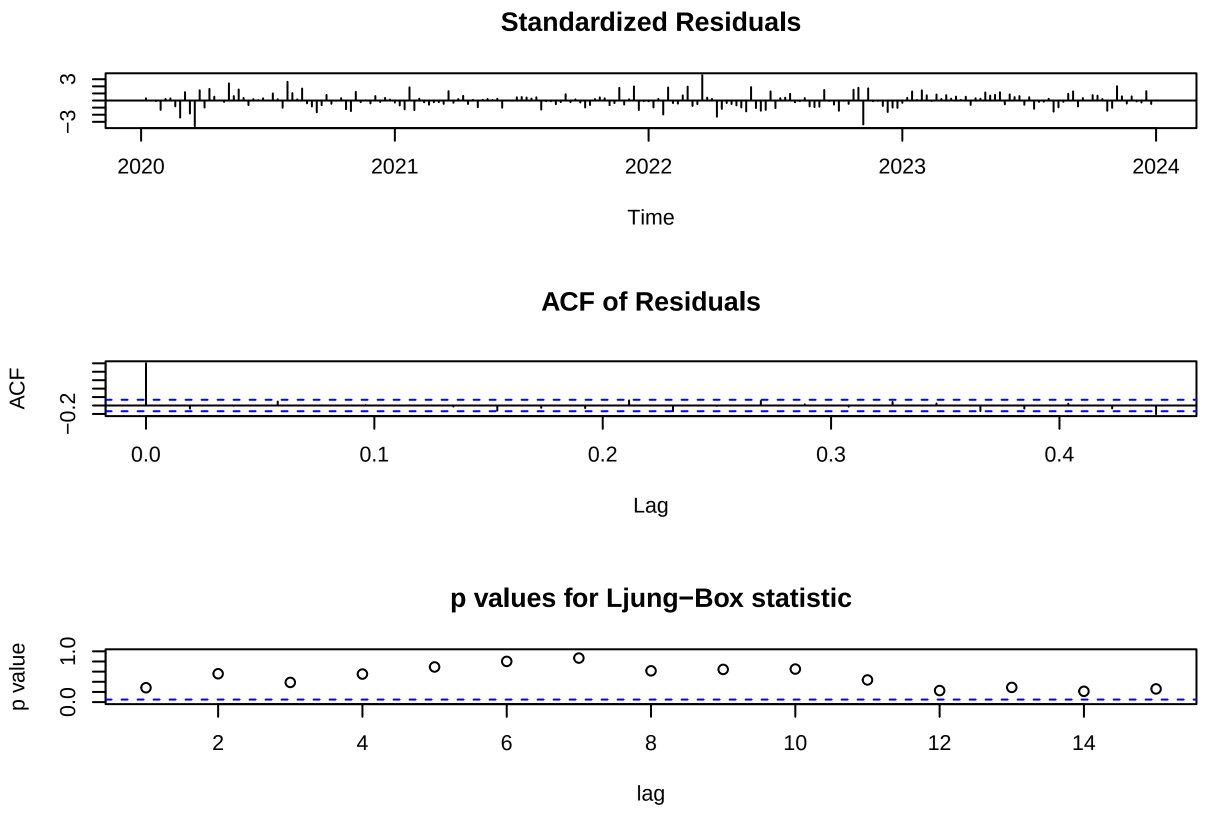
```r
# ACF of residuals
acf(U.2)
```



```r
plot(density(U.2))
```



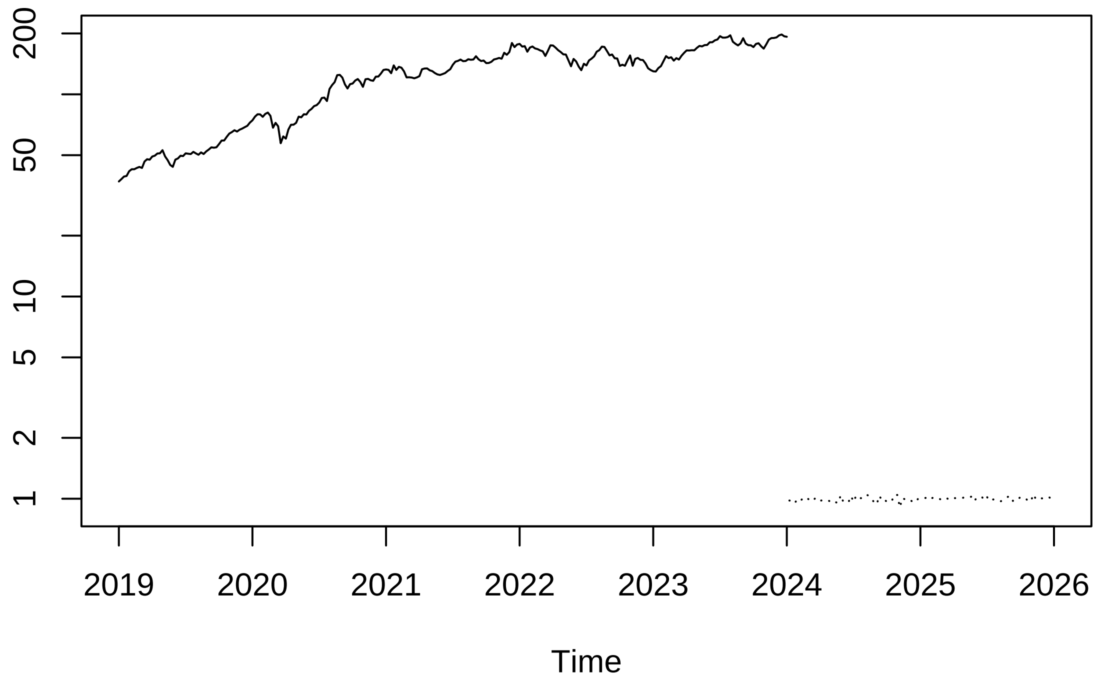
```r
#Using tsdiag tools
tsdiag(Weekly.model, gof.lag = 15)
```



Forecasting

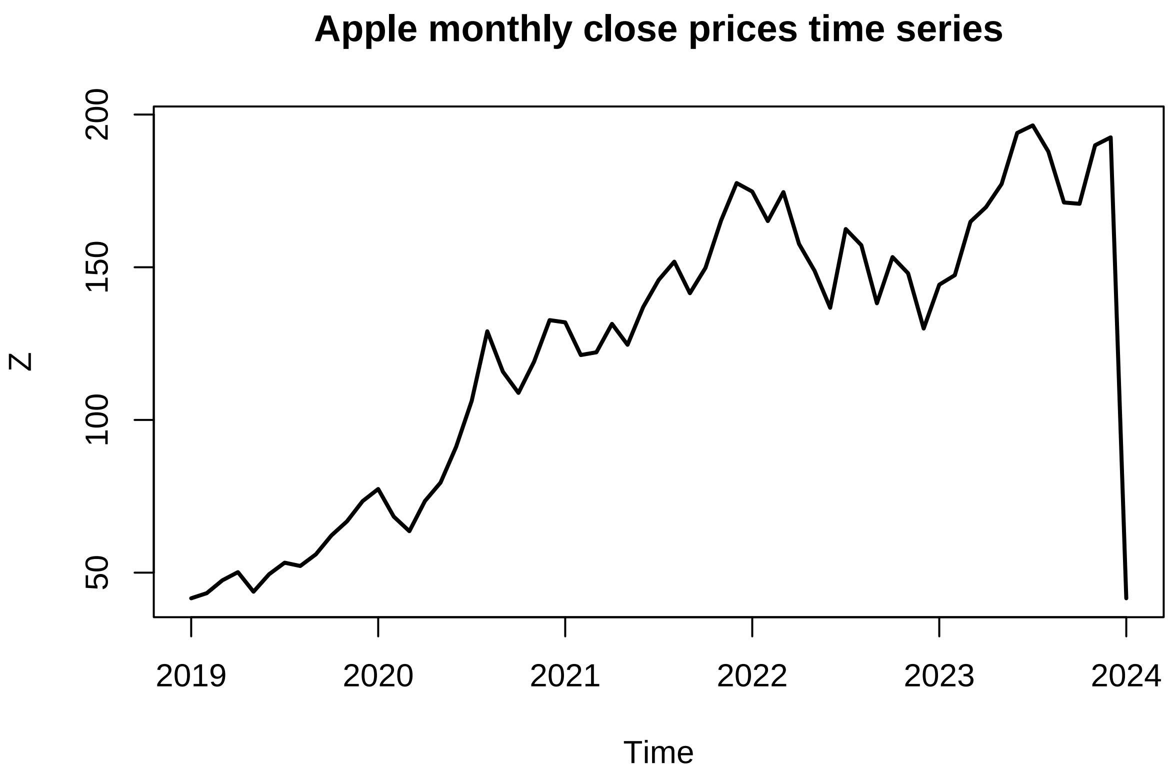
Using our best model, we can forecast the observations for the next 2 years.

```r
Weekly.pred <- predict(Weekly.model, n.ahead = 2*52)
ts.plot(Y, exp(Weekly.pred$pred), log = "y", lty =c(1,3), lwd = 1)
```


200
50
10
5
2
1
2019
2020
2021
2022
2023
2024
2025
2026
Time

5. Use the best subsets ARIMA approach to specify a model for the monthly close data.

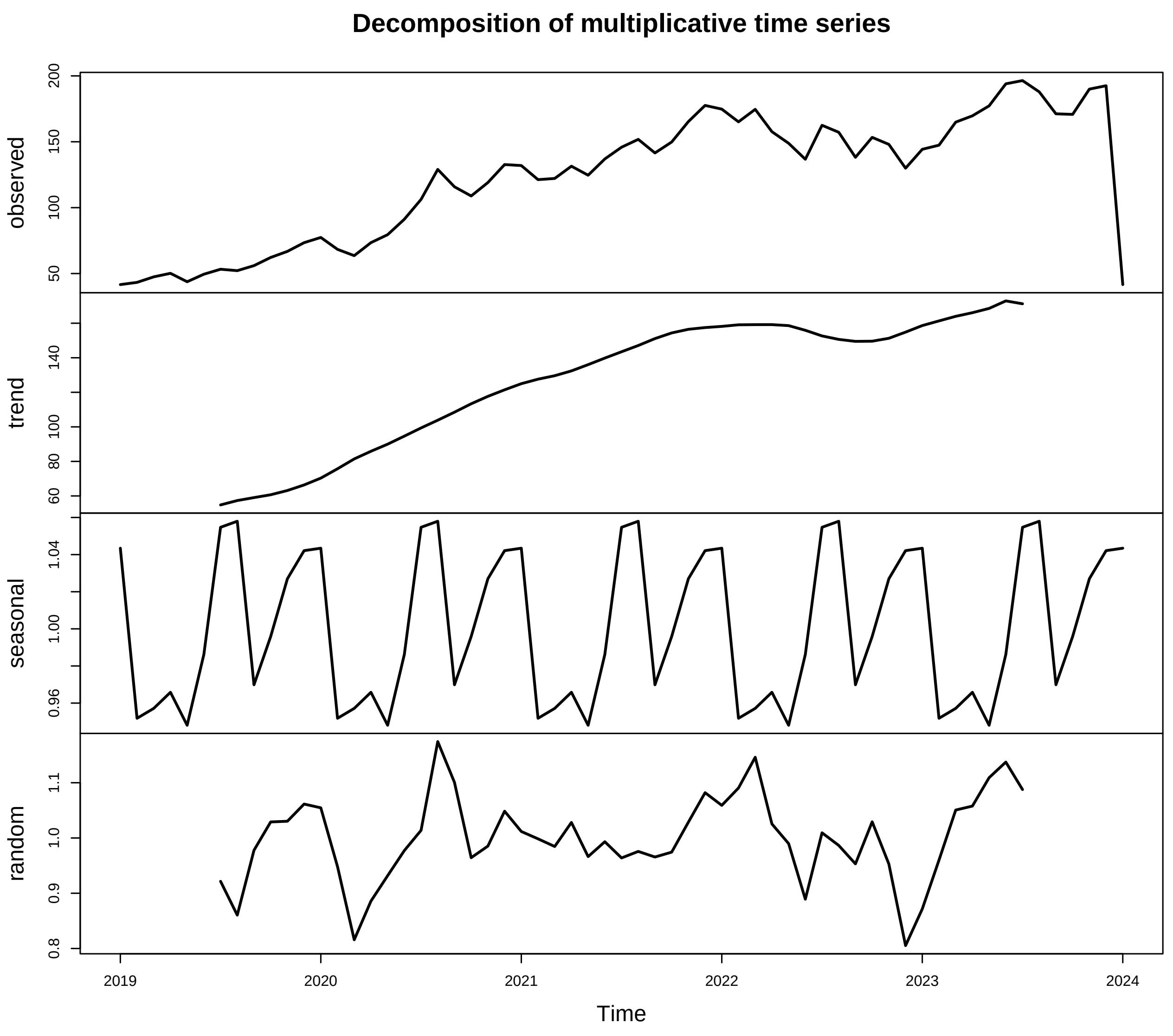
```r
# Aggregate daily data to monthly
Monthly <- to.monthly(Daily)
colnames(Monthly) <- c("Open", "High", "Low", "Close", "Volume", "Adjusted")

# Create monthly close prices time series
Z <- ts(data = Monthly$Close, start = c(2019, 1, 1), end =  c(2024, 1, 1), frequency = 12)

# Create monthly close prices time series
plot(Z, main="Apple monthly close prices time series", lwd = 2)
```



Decomposing the Data

Decomposing the data into its trend, seasonal, and random error components will give some idea how these components relate to the observed dataset.

```r
# Decomposition using multiplicative type
Z.decomp <- decompose(Z, type = "multiplicative")

plot(Z.decomp, lwd=2)
```



Summary Statistics of the Data

```r
summary(Z)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   41.61   73.45  132.69  122.95  162.51  196.45
```

```r
describe(Z)
```

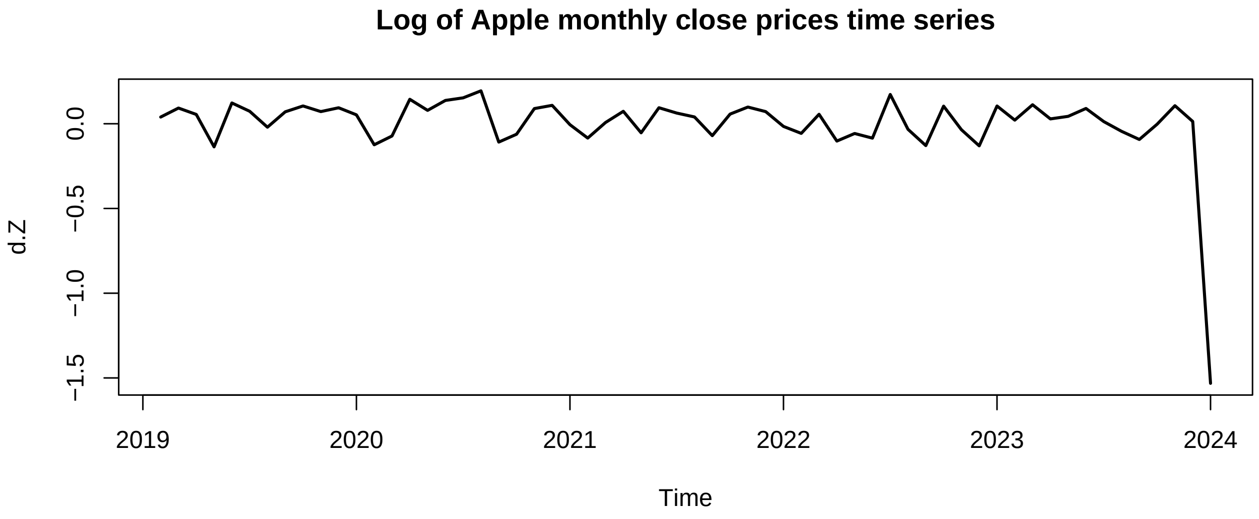
```
##    vars  n   mean    sd median trimmed   mad   min    max  range  skew kurtosis
## X1    1 61 122.95 48.02 132.69  124.37 54.84 41.61 196.45 154.84 -0.35    -1.23
##      se
## X1 6.15
```

Data Transformation To Achieve Stationarity

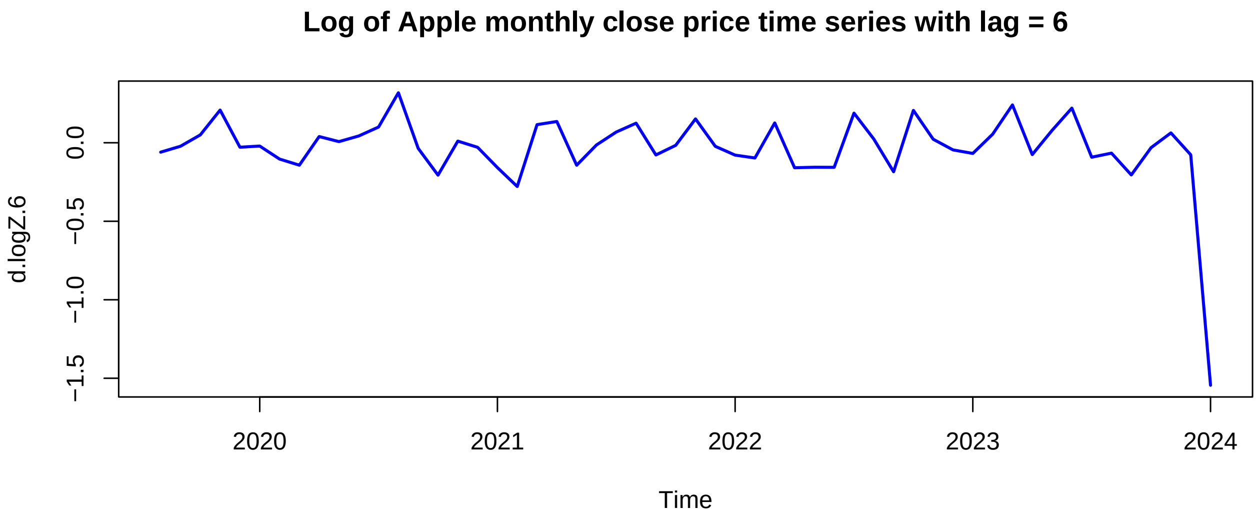
Now, we will have to perform some data transformation to achieve Stationarity.

```r
# Remove the trend from the data
# First log Difference
d.Z = diff(log(Z)) # to remove trend
```

```
plot(d.Z, main = "Log of Apple monthly close prices time series", lwd=2)
```



```
d.logZ.6 = diff(d.Z, lag = 6)  # remove seasonality
plot(d.logZ.6, main =" Log of Apple monthly close price time series with lag = 6 ", lwd=2, col="blue")
```



```
# test for stationarity
adf.test(d.logZ.6)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  d.logZ.6
## Dickey-Fuller = -0.91431, Lag order = 3, p-value = 0.9434
## alternative hypothesis: stationary
```

Comment: We see that the series is stationary enough to do any kind of time series modelling.

Identification of the best model

```
# Model Selection
set.seed(123)
```

```r
auto.arima(d.logZ.6, stationary = TRUE, ic = c("aic"), trace = TRUE)
```

```
##
##  ARIMA(2,0,2)(1,0,1)[12] with non-zero mean : Inf
##  ARIMA(0,0,0)            with non-zero mean : 4.418034
##  ARIMA(1,0,0)(1,0,0)[12] with non-zero mean : 7.33503
##  ARIMA(0,0,1)(0,0,1)[12] with non-zero mean : 6.529795
##  ARIMA(0,0,0)            with zero mean     : 3.424238
##  ARIMA(0,0,0)(1,0,0)[12] with non-zero mean : 5.730565
##  ARIMA(0,0,0)(0,0,1)[12] with non-zero mean : 5.782058
##  ARIMA(0,0,0)(1,0,1)[12] with non-zero mean : 7.573207
##  ARIMA(1,0,0)            with non-zero mean : 6.09048
##  ARIMA(0,0,1)            with non-zero mean : 5.347148
##  ARIMA(1,0,1)            with non-zero mean : 6.949911
##
##  Best model: ARIMA(0,0,0)            with zero mean
```

```
## Series: d.logZ.6
## ARIMA(0,0,0) with zero mean
##
## sigma^2 = 0.06011:  log likelihood = -0.71
## AIC=3.42   AICc=3.5   BIC=5.41
```

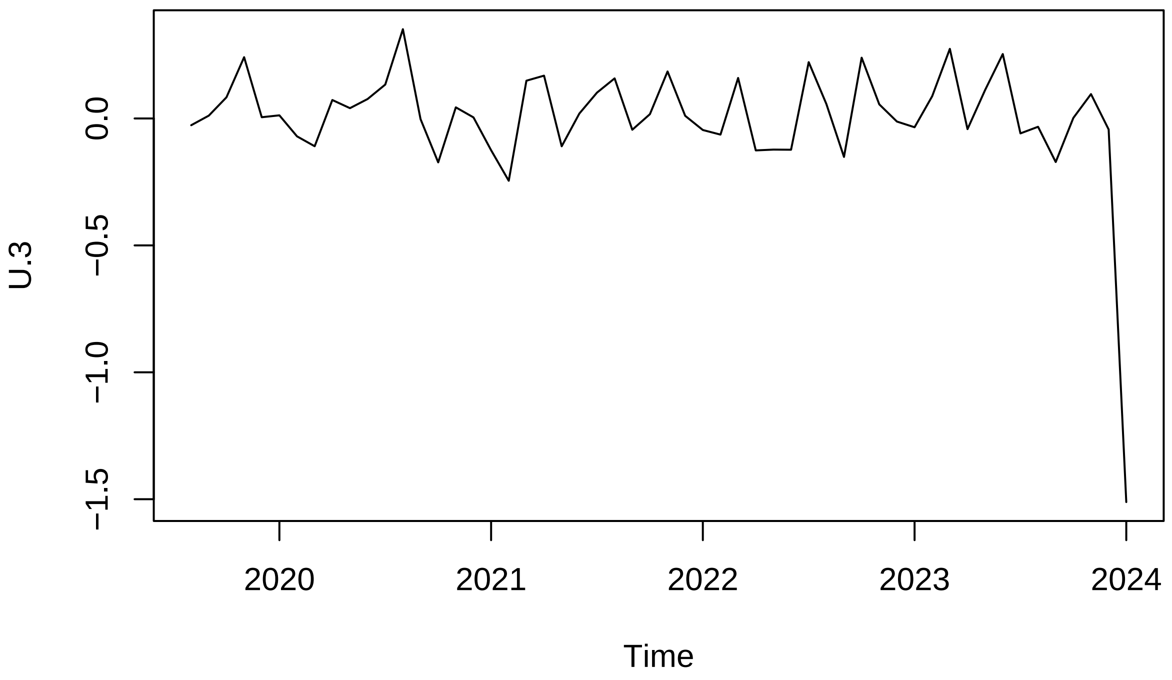
Comment: auto arima indicates us that the best arima model is ARIMA(0,0,0)

Weekly Model

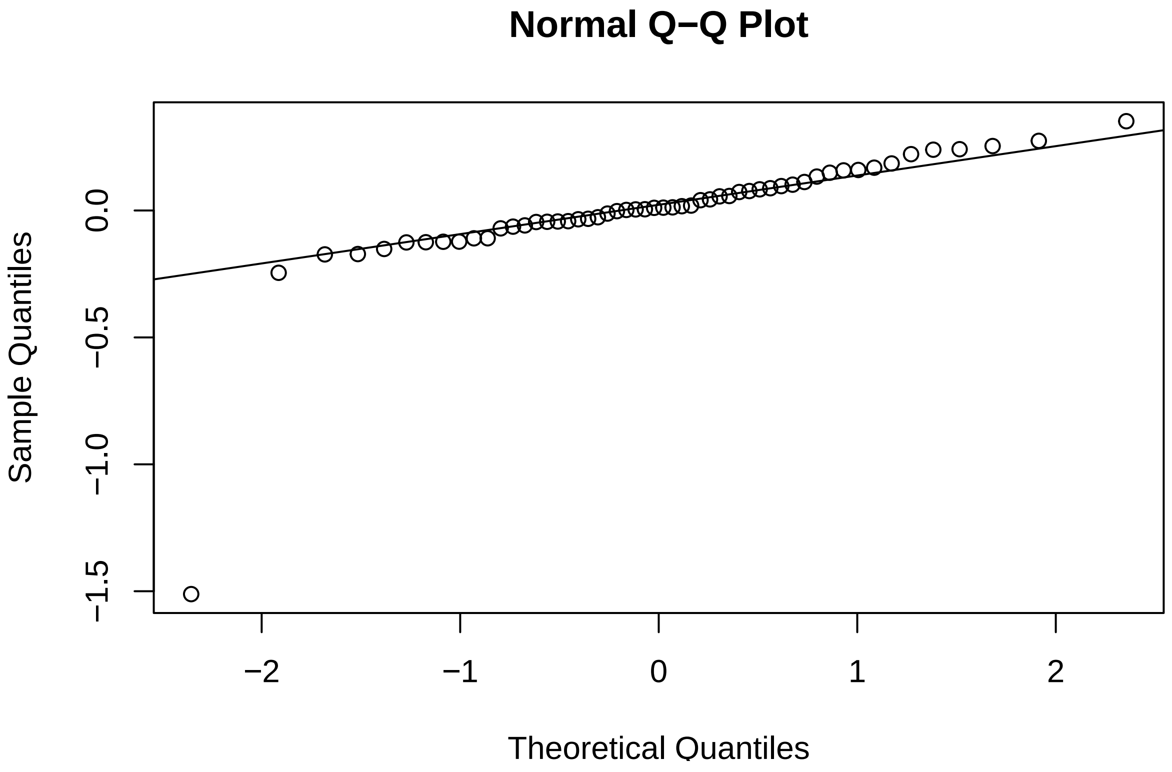
```r
## Setting up the Model
Monthly.model <- arima(d.logZ.6, c(0, 0, 0) )
```

Model evaluation

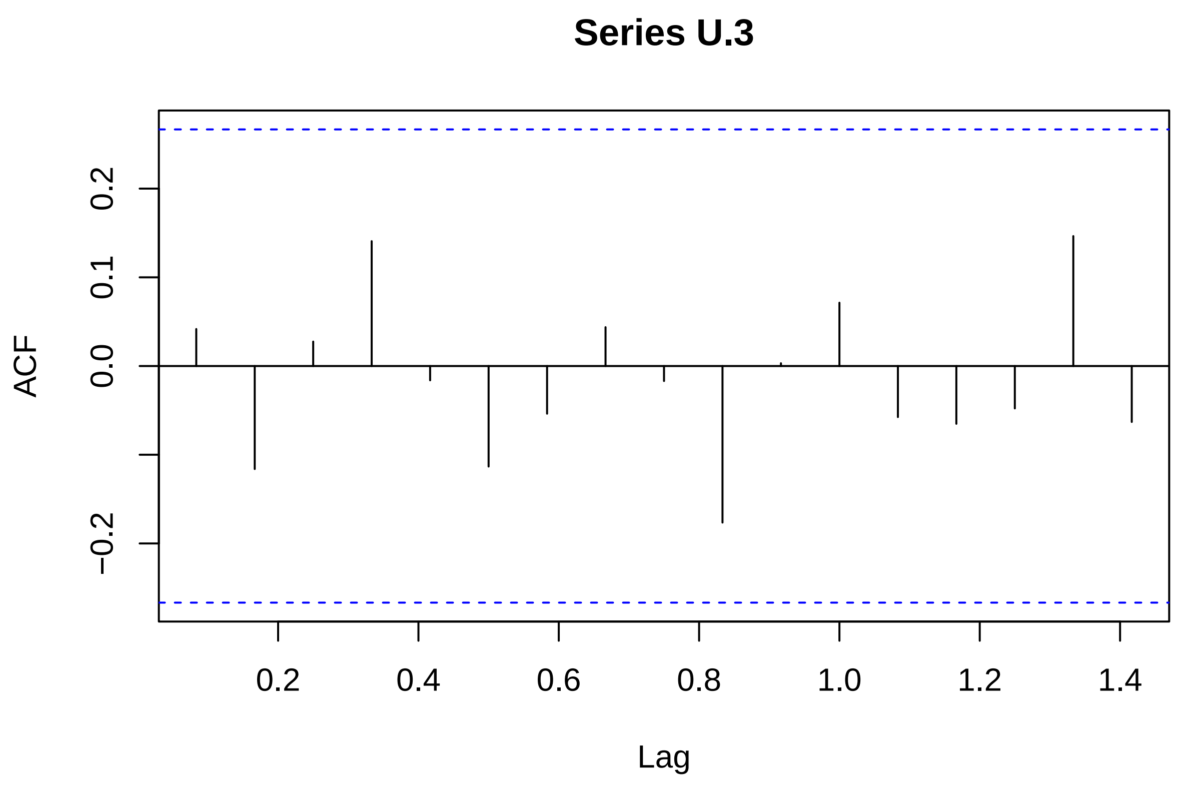
```r
## Diagnostic Checking
U.3 <- Monthly.model$residuals
plot(U.3)
```



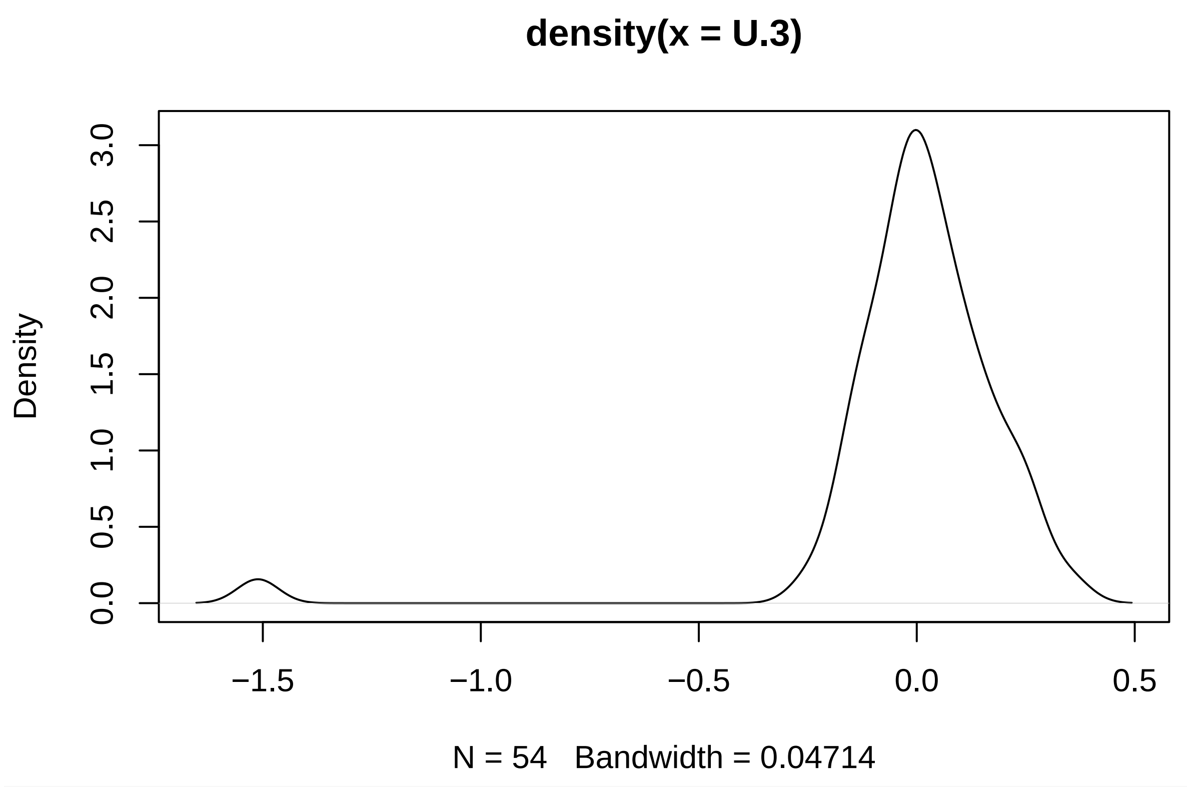
```r
# Normality of the Residuals
qqnorm(U.3); qqline(U.3)
```



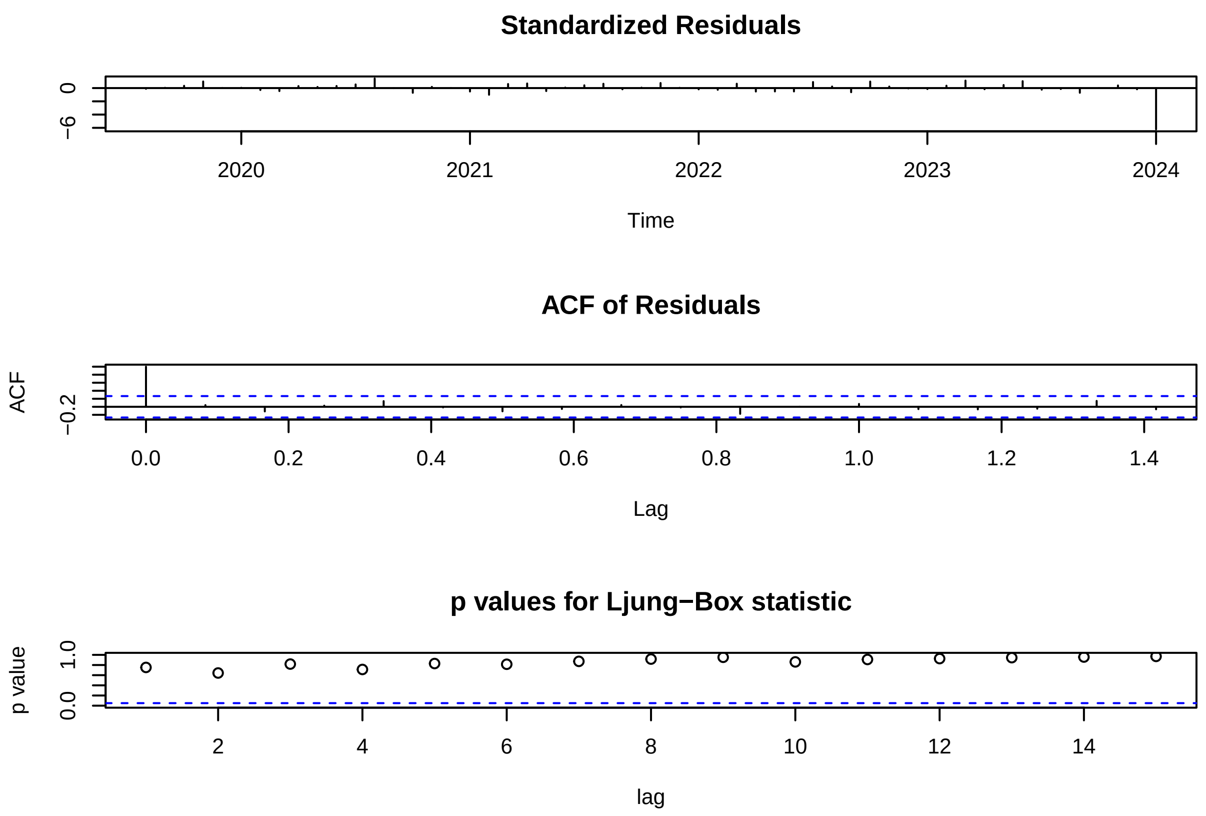
```r
# ACF of residuals
acf(U.3)
```



```r
plot(density(U.3))
```



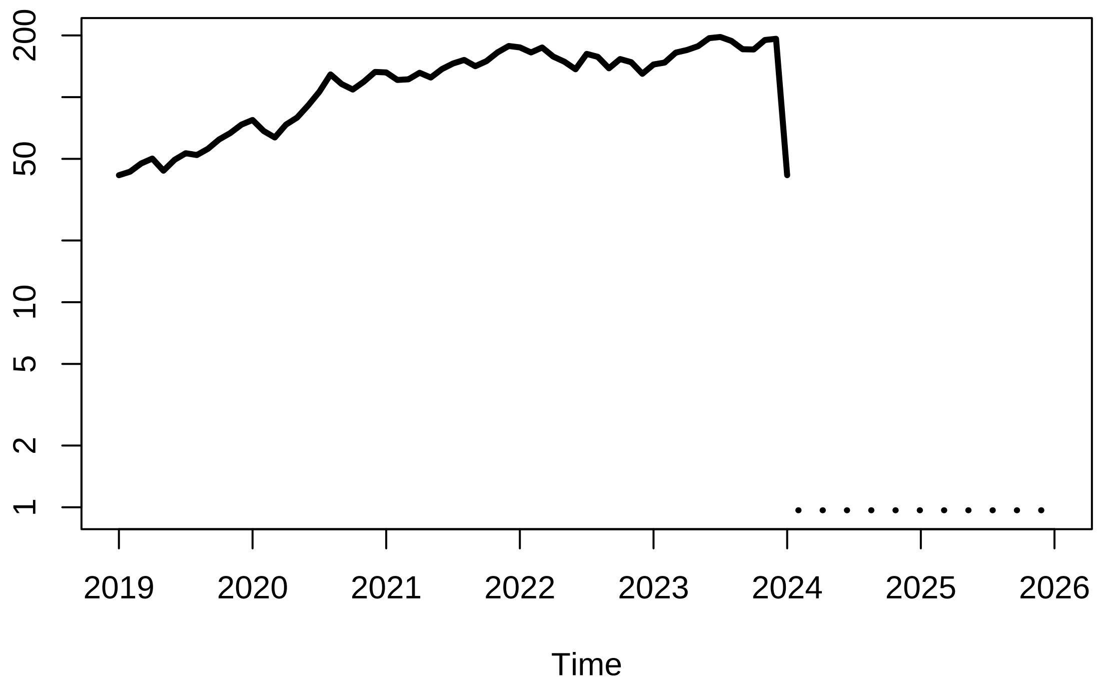
```r
#Using tsdiag tools
tsdiag(Monthly.model, gof.lag = 15)
```



Forecasting

Using our best model, we can forecast the observations for the next 2 years.

```
Monthly.pred <- predict(Monthly.model, n.ahead = 2*12)
ts.plot(Z, exp(Monthly.pred$pred), log = "y", lty =c(1,3), lwd = 3)
```



6. Briefly comment on the differences between the best models for daily, weekly and monthly data.

Comparing Models

Looking at the AIC/BIC values, coefficients, and diagnostic statistics of each of those three models, the Daily model in term of performance is better than Weekly one which its turn is better than the Monthly one.