PSTAT 174 Final Project

Nathan Ho

2025-03-10

Load in and Explore Data

Original dataset contains Nvidia stock closing prices from 2018 - 2024 New dataset includes Nvidia stock closing prices from January 2022 to September 2024.

```
library(astsa)
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(timeSeries)
```

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

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

```
## The following objects are masked from 'package:graphics':
##
##     lines, points
```

```
library(forecast)
```

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

```
## The following object is masked from 'package:astsa':
##
##     gas
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:timeSeries':
##
##     filter, lag

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(fGarch)
```

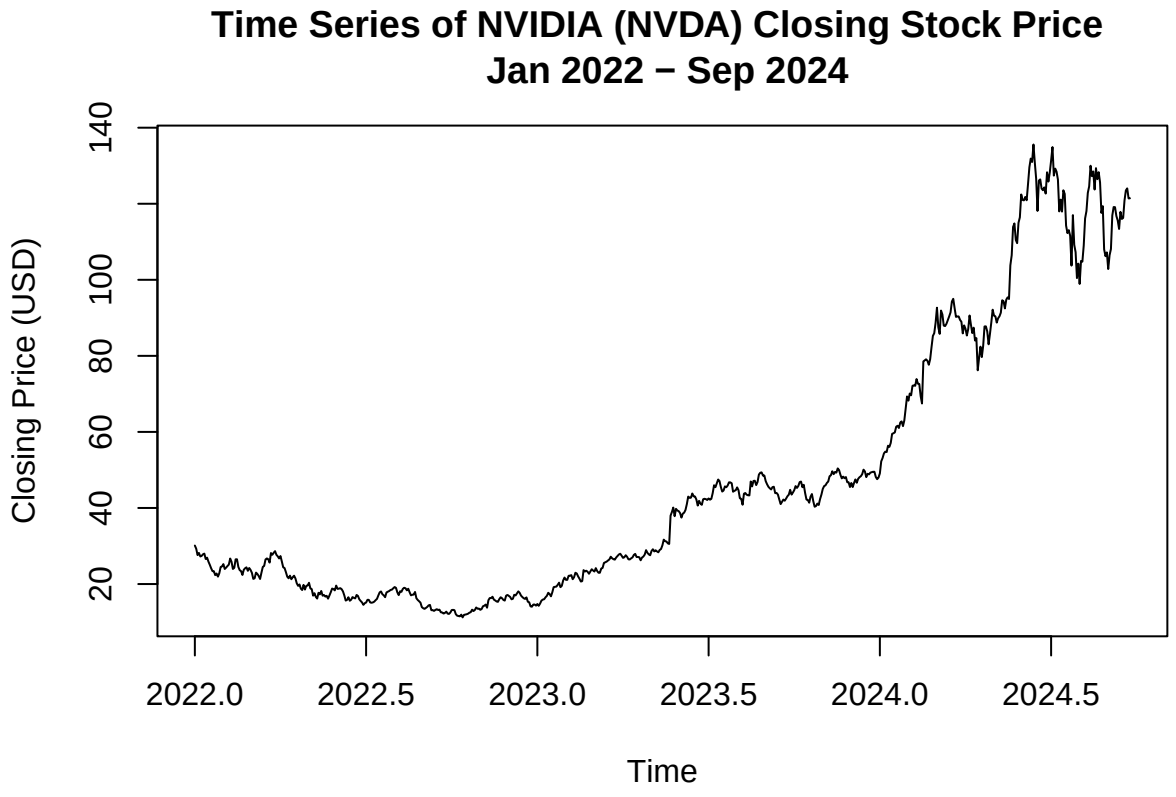
```
## NOTE: Packages 'fBasics', 'timeDate', and 'timeSeries' are no longer
## attached to the search() path when 'fGarch' is attached.
##
## If needed attach them yourself in your R script by e.g.,
##         require("timeSeries")
```

```r
library(rugarch)
```

```
## Loading required package: parallel

##
## Attaching package: 'rugarch'

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

```r
library(pander)

NVDA <- read.csv("~/Downloads/NVIDIA_STOCK.csv") %>%
  rename(Date = Price)

NVDA <- NVDA %>%
  mutate(Date = as.Date(Date, format="%Y-%m-%d")) %>%
  filter(Date >= as.Date("2022-01-03") & Date <= as.Date("2024-09-30")) %>%
  arrange(Date)

ClosingPrice <- NVDA$Close

NVDAts <- ts(as.numeric(ClosingPrice), start = c(2022, 1), frequency = 252)
```

```r
plot(NVDAts,
     main="Time Series of NVIDIA (NVDA) Closing Stock Price \n Jan 2022 - Sep 2024",
     ylab="Closing Price (USD)",
     xlab="Time")
```




```r
NVDAts <- as.numeric(NVDAts)
```

Interpretation: 1. Strong Upward Trend: - Stock price has been increasing over time - Suggests a long-term growth in Nvidia's stock

2. Exponential Growth Pattern:

- Growth accelerates sharply after mid 2023, suggesting possible non-linear
- Suggests that a log transformation is needed

3. Non-Stationary Data:

- The trend suggests that the data is not stationary, meaning that differencing would be required before fitting an ARIMA model.
- Exponential increase suggests log transformation is necessary to stabilize variance.

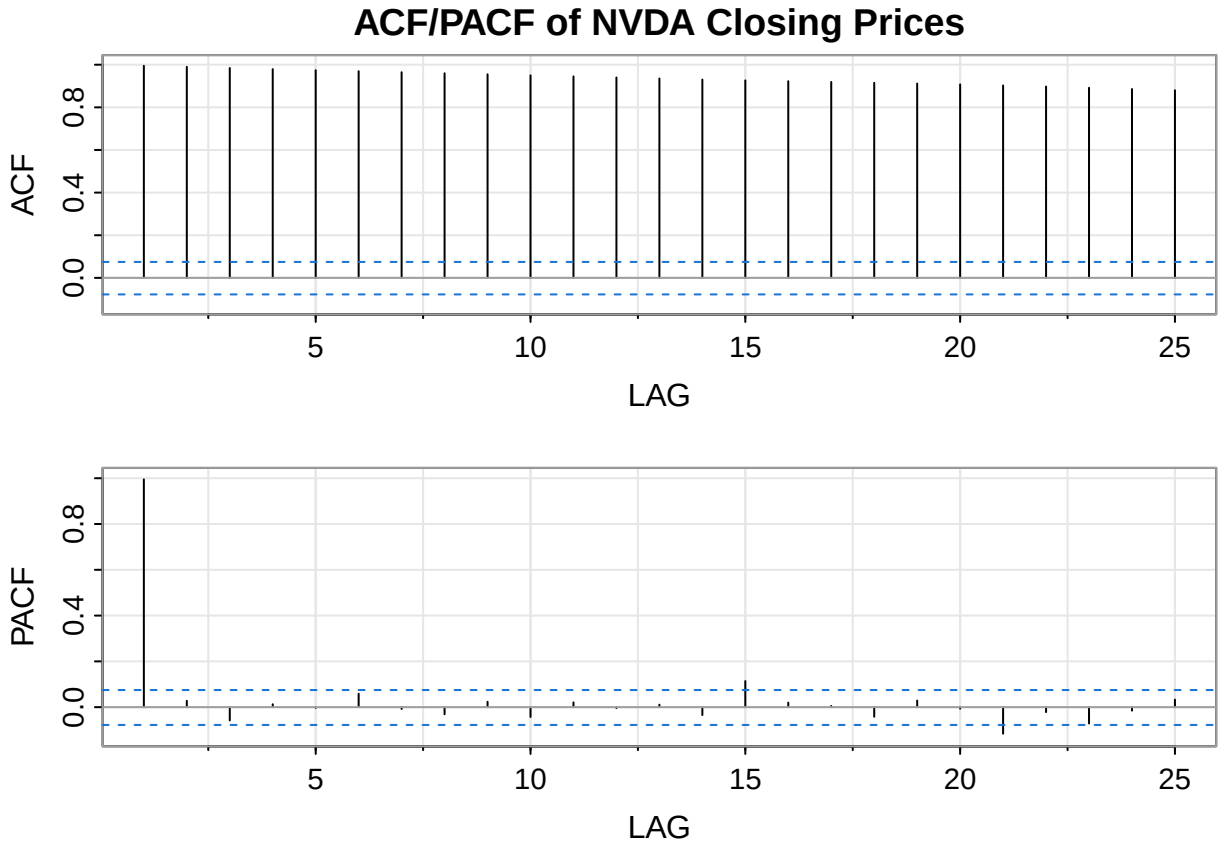
Check for Stationarity

Use Augmented Dickey-Fuller (ADF) test to test stationarity

```
adf.test(NVDAts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  NVDAts
## Dickey-Fuller = -1.9029, Lag order = 8, p-value = 0.6194
## alternative hypothesis: stationary
```

```
acf2(NVDAts, 25, main = "ACF/PACF of NVDA Closing Prices")
```



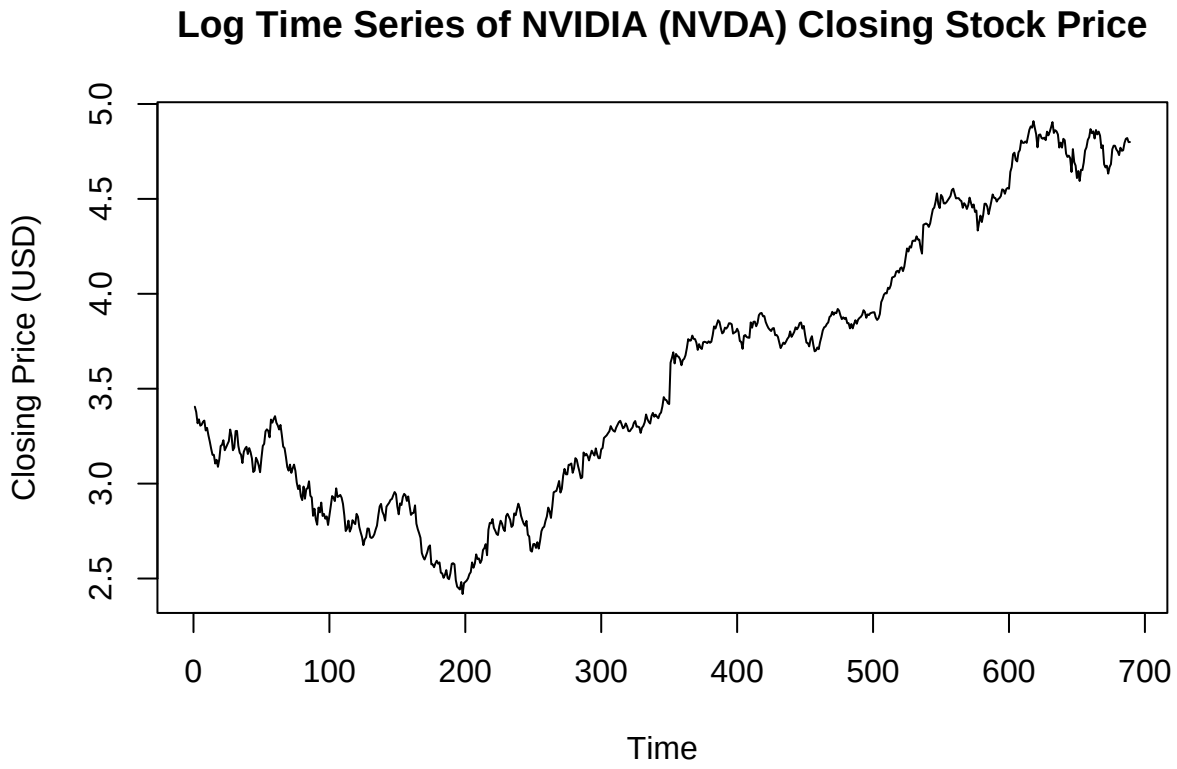

```
##      [,1] [,2]  [,3] [,4] [,5] [,6]  [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF  0.99 0.99  0.98 0.98 0.97 0.97  0.96  0.96 0.96  0.95  0.95  0.94  0.94
## PACF 0.99 0.03 -0.06 0.01 0.00 0.06 -0.01 -0.03 0.02 -0.04  0.02  0.00  0.01
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF   0.93  0.93  0.92  0.92  0.92  0.91  0.91  0.90  0.90  0.89  0.89  0.88
## PACF -0.03  0.11  0.02  0.01 -0.04  0.03 -0.01 -0.12 -0.02 -0.07 -0.02  0.03
```

Interpretation: 1. Non-Stationary Data: The ADF test p-value = 0.6194 is greater than 0.05, which means that we fail to reject the null hypothesis of stationarity Meaning: Stock prices exhibit a trend and require differencing before fitting an ARIMA model. First-order differencing (d=1) is required to remove the trend and make the data stationary.

Transform

Log Transformation: reduces the heteroskedasticity and helps stabilize variance. Making trends more linear, improving model performance.

```
NVDALog <- log(NVDAts)
plot(NVDALog,
     main="Log Time Series of NVIDIA (NVDA) Closing Stock Price",
     ylab="Closing Price (USD)",
     xlab="Time",
     type="l")
```

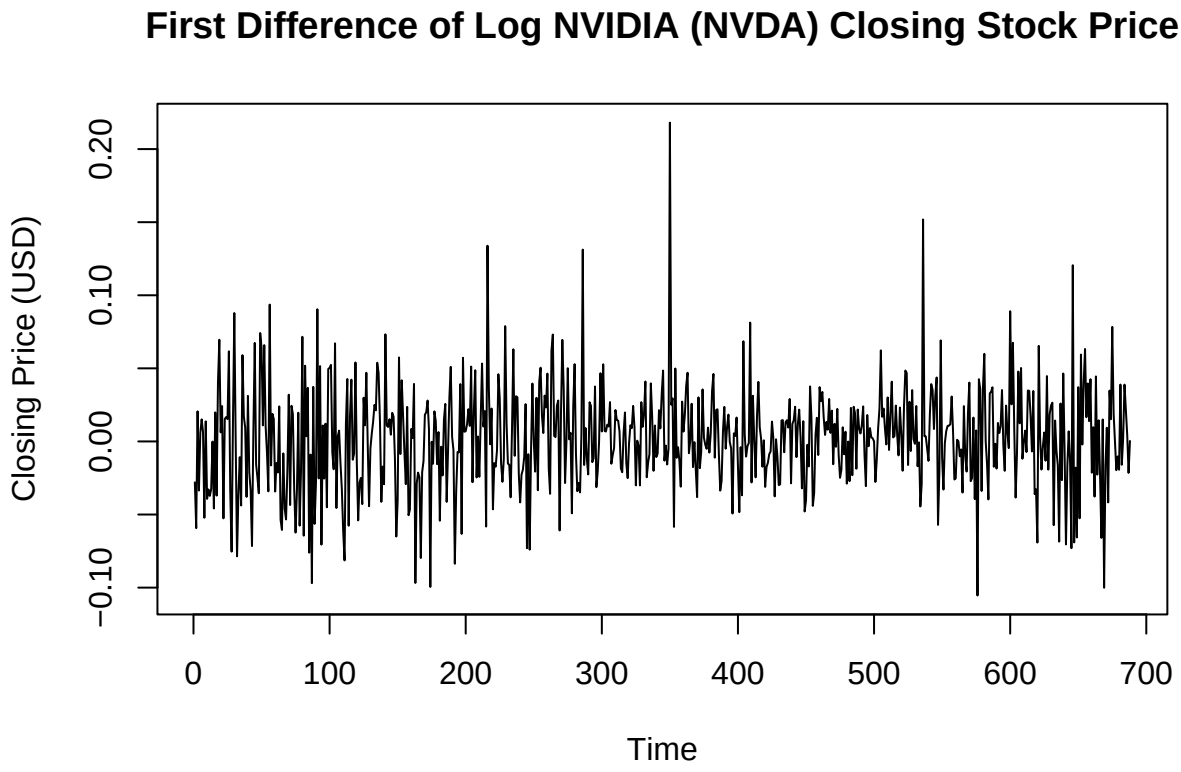


First order differencing is used to remove trends and make data stationary. NVidia's stock price has a strong long-term trend and differencing is needed to model short term fluctuations effectively.

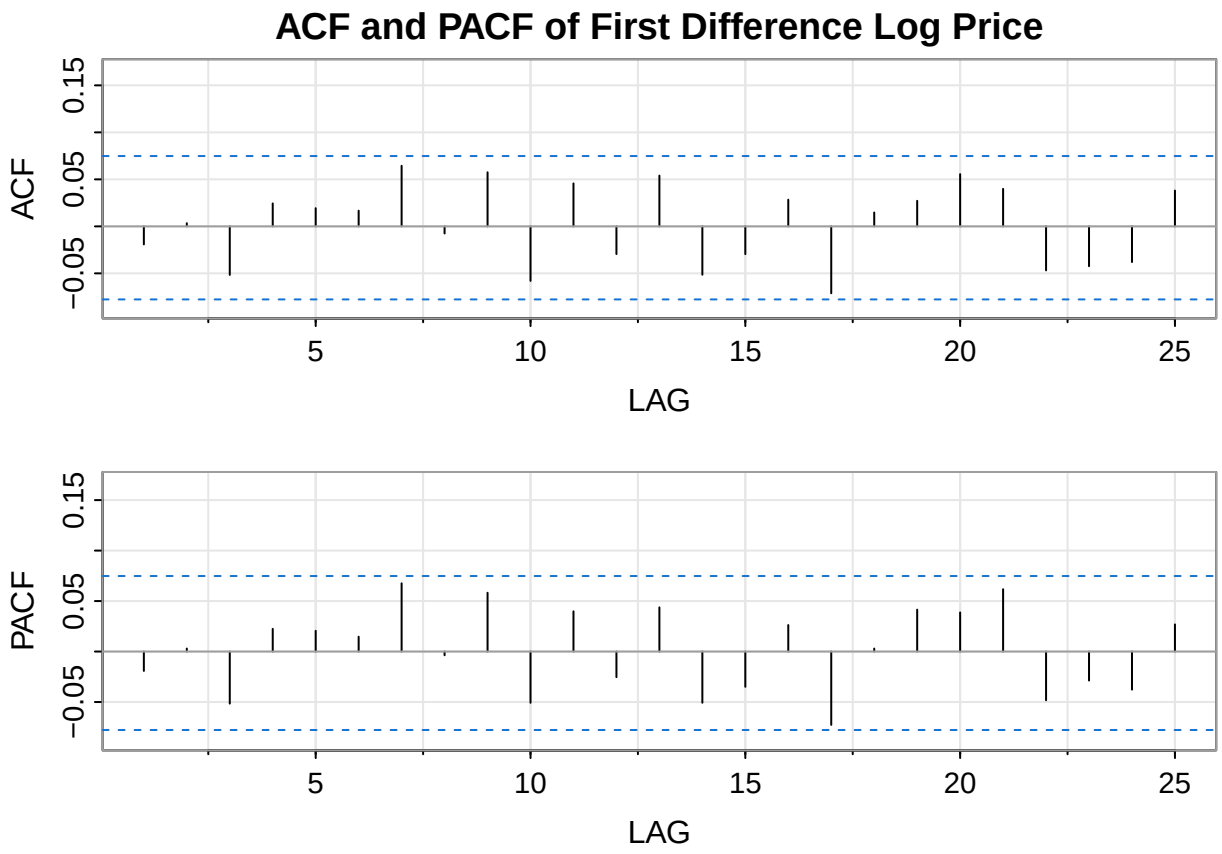
```
NVDALogDiff = diff(log(NVDAts))
adf.test(NVDALogDiff)
```

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

##
##  Augmented Dickey-Fuller Test
##
## data:  NVDALogDiff
## Dickey-Fuller = -7.8345, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```r
plot(NVDALogDiff,
     main="First Difference of Log NVIDIA (NVDA) Closing Stock Price",
     ylab="Closing Price (USD)",
     xlab="Time",
     type="l")
```




```r
acf2(NVDALogDiff, 25, main = "ACF and PACF of First Difference Log Price")
```




```
##       [,1] [,2]  [,3] [,4] [,5] [,6] [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF  -0.02    0 -0.05 0.02 0.02 0.02 0.06 -0.01 0.06 -0.06  0.05 -0.03  0.05
## PACF -0.02    0 -0.05 0.02 0.02 0.01 0.07  0.00 0.06 -0.05  0.04 -0.03  0.04
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF  -0.05 -0.03  0.03 -0.07  0.01  0.03  0.06  0.04 -0.05 -0.04 -0.04  0.04
## PACF -0.05 -0.03  0.03 -0.07  0.00  0.04  0.04  0.06 -0.05 -0.03 -0.04  0.03
```

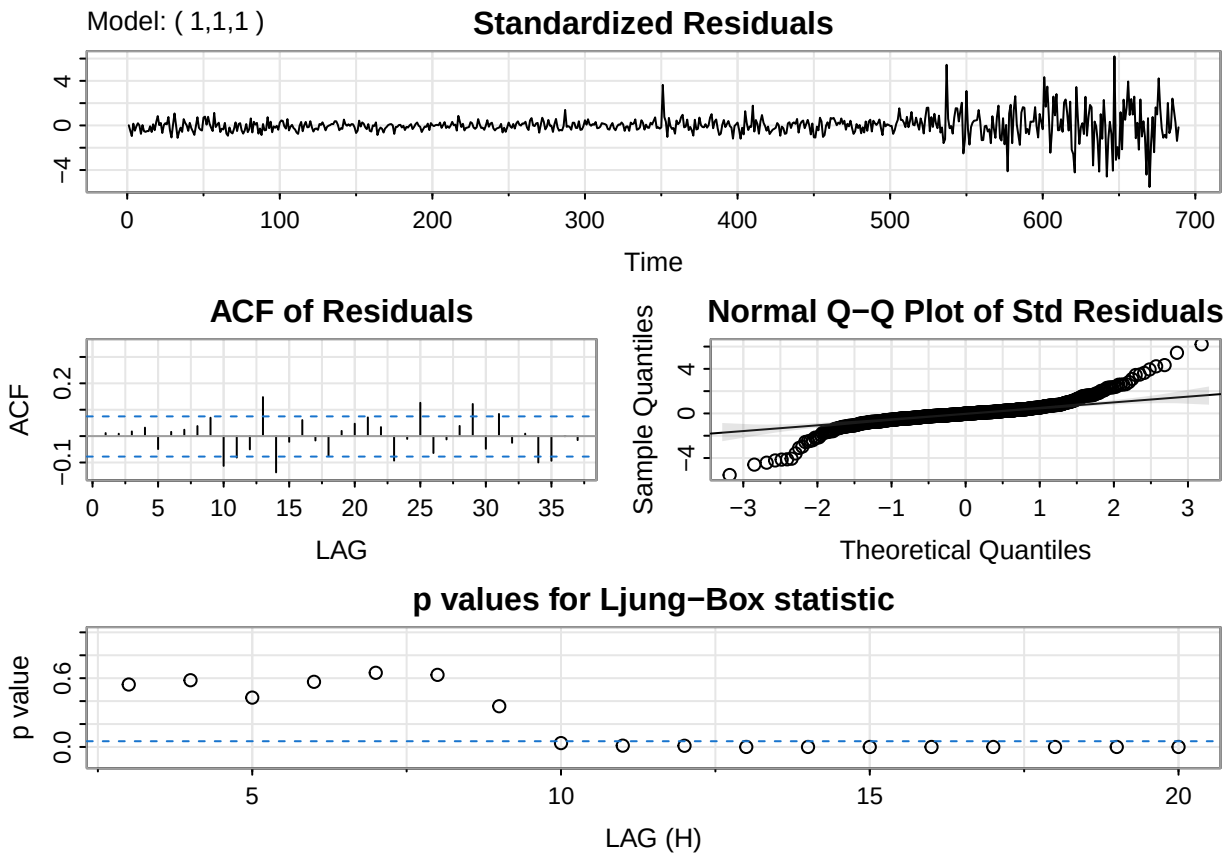
p-value = 0.01 reject the null hypothesis (now stationary) ACF and PACF analysis Autocorrelation Function (ACF): - Strong correlation at multiple lags, confirming that there is dependence between stock prices over time - Gradual decay suggests a moving average (MA) component is necessary Partial Autocorrelation Function (PACF): - First lag is significant, indicating the need for autoregressive (AR) terms Key Insight: Both ACF and PACF indicate the presence of short-term memory effects, meaning SARIMA modeling is appropriate

Testing different models for ACF/PACFs

```
Sarima1 <- sarima(NVDAts, p=1, d=1, q=1)
```

```
## initial  value 0.711360
## iter   2 value 0.708138
## iter   3 value 0.706565
## iter   4 value 0.706540
```

```
## iter   5 value 0.705533
## iter   6 value 0.703526
## iter   7 value 0.703155
## iter   8 value 0.703063
## iter   9 value 0.702962
## iter  10 value 0.702432
## iter  11 value 0.701960
## iter  12 value 0.701945
## iter  13 value 0.701932
## iter  14 value 0.701916
## iter  15 value 0.701905
## iter  16 value 0.701902
## iter  17 value 0.701902
## iter  18 value 0.701902
## iter  19 value 0.701901
## iter  20 value 0.701901
## iter  21 value 0.701901
## iter  22 value 0.701901
## iter  22 value 0.701901
## final  value 0.701901
## converged
## initial  value 0.700745
## iter   2 value 0.700744
## iter   3 value 0.700742
## iter   4 value 0.700741
## iter   5 value 0.700738
## iter   6 value 0.700735
## iter   7 value 0.700734
## iter   8 value 0.700733
## iter   9 value 0.700733
## iter  10 value 0.700732
## iter  11 value 0.700732
## iter  11 value 0.700732
## iter  11 value 0.700732
## final  value 0.700732
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE t.value p.value
## ar1       -0.7236 0.1008 -7.1755   0.000
## ma1        0.6254 0.1118  5.5956   0.000
## constant   0.1324 0.0725  1.8278   0.068
##
## sigma^2 estimated as 4.060946 on 685 degrees of freedom
##
## AIC = 4.250968  AICc = 4.251019  BIC = 4.277328
##
```



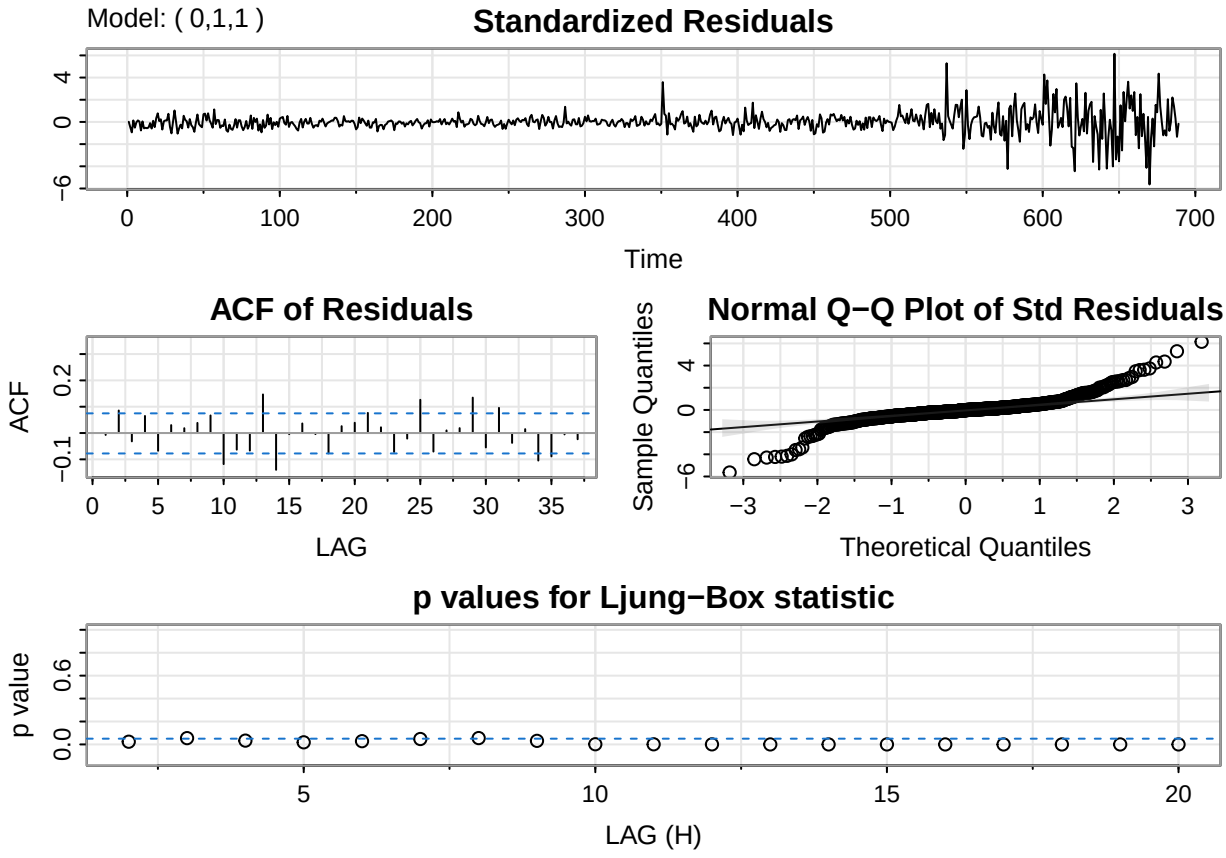

```
summary(arima(NVDAts, order=c(1,1,1)))
```

```
##
## Call:
## arima(x = NVDAts, order = c(1, 1, 1))
##
## Coefficients:
##           ar1     ma1
##       -0.7273  0.6312
## s.e.   0.0990  0.1096
##
## sigma^2 estimated as 4.081:  log likelihood = -1460,  aic = 2926
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE     MASE
## Training set 0.1402642 2.018602 1.208525 0.1490644 2.674539 1.000291
##                      ACF1
## Training set 0.009535824
```

```
Sarima2 <- sarima(NVDAts, p=0, d=1, q=1)
```

```
## initial  value 0.710796
## iter   2 value 0.706507
## iter   3 value 0.706394
```

```
## iter   4 value 0.706394
## iter   4 value 0.706394
## iter   4 value 0.706394
## final  value 0.706394
## converged
## initial  value 0.706398
## iter   1 value 0.706398
## final  value 0.706398
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE t.value p.value
## ma1       -0.0870 0.0352 -2.4689  0.0138
## constant   0.1328 0.0706  1.8828  0.0601
##
## sigma^2 estimated as 4.107378 on 686 degrees of freedom
##
## AIC = 4.259394  AICc = 4.259419  BIC = 4.279163
##
```




```
summary(arima(NVDAts, order=c(0,1,1)))
```

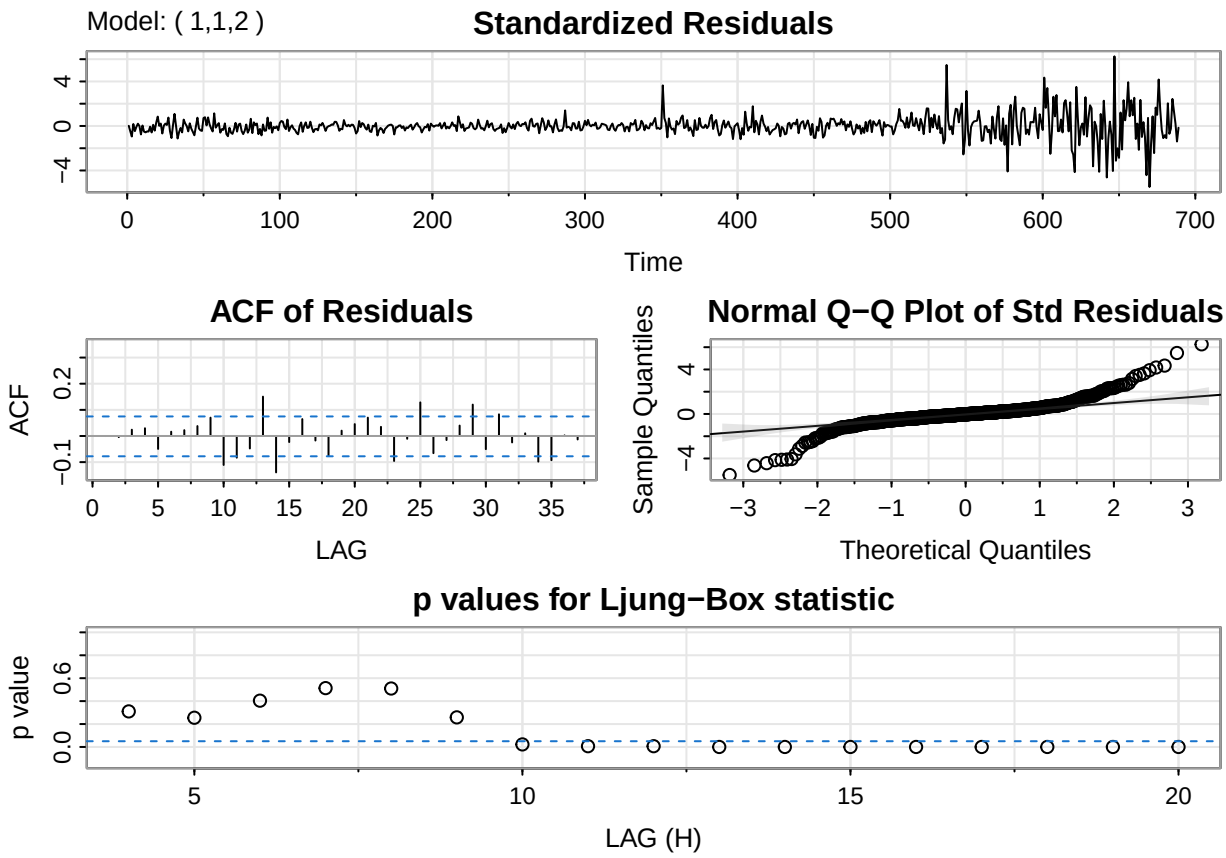
```
##
## Call:
```

```
## arima(x = NVDAts, order = c(0, 1, 1))
##
## Coefficients:
##           ma1
##       -0.0821
## s.e.   0.0351
##
## sigma^2 estimated as 4.128:  log likelihood = -1463.99,  aic = 2931.98
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE     MASE
## Training set 0.1444718 2.030386 1.212188 0.1533743 2.675669 1.003323
##                      ACF1
## Training set -0.01252788
```

```
Sarima3 <- sarima(NVDAts, p=1, d=1, q=2)
```

```
## initial  value 0.711360
## iter   2 value 0.709881
## iter   3 value 0.703759
## iter   4 value 0.703675
## iter   5 value 0.703669
## iter   6 value 0.703668
## iter   7 value 0.703651
## iter   8 value 0.703619
## iter   9 value 0.703469
## iter  10 value 0.703254
## iter  11 value 0.702885
## iter  12 value 0.702570
## iter  13 value 0.702508
## iter  14 value 0.702347
## iter  15 value 0.702326
## iter  16 value 0.702274
## iter  17 value 0.702176
## iter  18 value 0.701999
## iter  19 value 0.701849
## iter  20 value 0.701802
## iter  21 value 0.701713
## iter  22 value 0.701695
## iter  23 value 0.701670
## iter  24 value 0.701664
## iter  25 value 0.701663
## iter  26 value 0.701662
## iter  27 value 0.701660
## iter  28 value 0.701658
## iter  29 value 0.701657
## iter  30 value 0.701657
## iter  31 value 0.701657
## iter  32 value 0.701657
## iter  33 value 0.701657
## iter  34 value 0.701657
## iter  35 value 0.701657
## iter  36 value 0.701657
## iter  37 value 0.701656
```

```
## iter  37 value 0.701656
## final  value 0.701656
## converged
## initial  value 0.700545
## iter   2 value 0.700545
## iter   3 value 0.700543
## iter   4 value 0.700541
## iter   5 value 0.700539
## iter   6 value 0.700535
## iter   7 value 0.700533
## iter   8 value 0.700531
## iter   9 value 0.700530
## iter  10 value 0.700529
## iter  11 value 0.700529
## iter  12 value 0.700529
## iter  13 value 0.700529
## iter  14 value 0.700529
## iter  15 value 0.700529
## iter  15 value 0.700529
## iter  15 value 0.700529
## final  value 0.700529
## converged
## <><><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE t.value p.value
## ar1       -0.7022 0.1185 -5.9235  0.0000
## ma1        0.6161 0.1240  4.9687  0.0000
## ma2        0.0217 0.0406  0.5347  0.5930
## constant   0.1324 0.0739  1.7918  0.0736
##
## sigma^2 estimated as 4.059275 on 684 degrees of freedom
##
## AIC = 4.253469  AICc = 4.253554  BIC = 4.286418
##
```




```
summary(arima(NVDAts, order=c(1,1,2)))
```

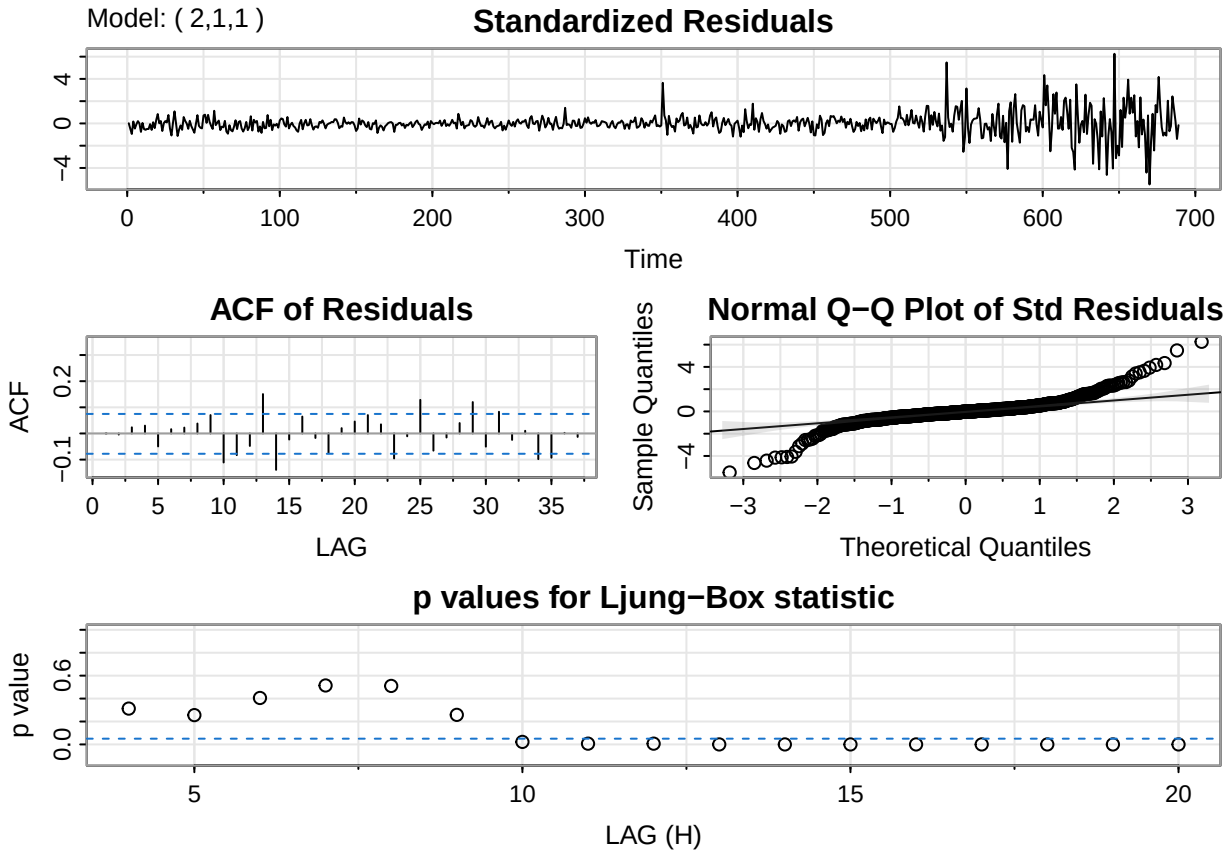
```
##
## Call:
## arima(x = NVDAts, order = c(1, 1, 2))
##
## Coefficients:
##           ar1     ma1     ma2
##       -0.7001  0.6187  0.0266
## s.e.   0.1192  0.1246  0.0402
##
## sigma^2 estimated as 4.078:  log likelihood = -1459.79,  aic = 2927.57
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE    MASE
## Training set 0.1368158 2.017967 1.208355 0.1454333 2.671147 1.00015
##                       ACF1
## Training set -0.004946399
```

```
Sarima4 <- sarima(NVDAts, p=2, d=1, q=1)
```

```
## initial  value 0.711508
## iter   2 value 0.709287
## iter   3 value 0.703321
```

```
## iter   4 value 0.703251
## iter   5 value 0.703247
## iter   6 value 0.703239
## iter   7 value 0.703217
## iter   8 value 0.703172
## iter   9 value 0.703071
## iter  10 value 0.702796
## iter  11 value 0.702745
## iter  12 value 0.702443
## iter  13 value 0.702312
## iter  14 value 0.702160
## iter  15 value 0.702130
## iter  16 value 0.702108
## iter  17 value 0.702069
## iter  18 value 0.701953
## iter  19 value 0.701721
## iter  20 value 0.701507
## iter  21 value 0.701404
## iter  22 value 0.701360
## iter  23 value 0.701299
## iter  24 value 0.701274
## iter  25 value 0.701261
## iter  26 value 0.701245
## iter  27 value 0.701213
## iter  28 value 0.701174
## iter  29 value 0.701164
## iter  30 value 0.701149
## iter  31 value 0.701144
## iter  32 value 0.701142
## iter  33 value 0.701139
## iter  34 value 0.701138
## iter  35 value 0.701137
## iter  36 value 0.701136
## iter  37 value 0.701134
## iter  38 value 0.701134
## iter  39 value 0.701134
## iter  40 value 0.701133
## iter  41 value 0.701133
## iter  42 value 0.701133
## iter  43 value 0.701133
## iter  44 value 0.701133
## iter  45 value 0.701133
## iter  46 value 0.701133
## iter  46 value 0.701133
## final  value 0.701133
## converged
## initial  value 0.700538
## iter   2 value 0.700537
## iter   3 value 0.700537
## iter   4 value 0.700536
## iter   5 value 0.700536
## iter   6 value 0.700536
## iter   7 value 0.700536
## iter   8 value 0.700535
```

```
## iter   9 value 0.700535
## iter  10 value 0.700534
## iter  11 value 0.700534
## iter  11 value 0.700534
## iter  11 value 0.700534
## final  value 0.700534
## converged
## <><><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE t.value p.value
## ar1       -0.6695 0.1580 -4.2377  0.0000
## ar2        0.0243 0.0464  0.5234  0.6009
## ma1        0.5830 0.1537  3.7938  0.0002
## constant   0.1323 0.0739  1.7905  0.0738
##
## sigma^2 estimated as 4.059323 on 684 degrees of freedom
##
## AIC = 4.253481  AICc = 4.253566  BIC = 4.28643
##
```



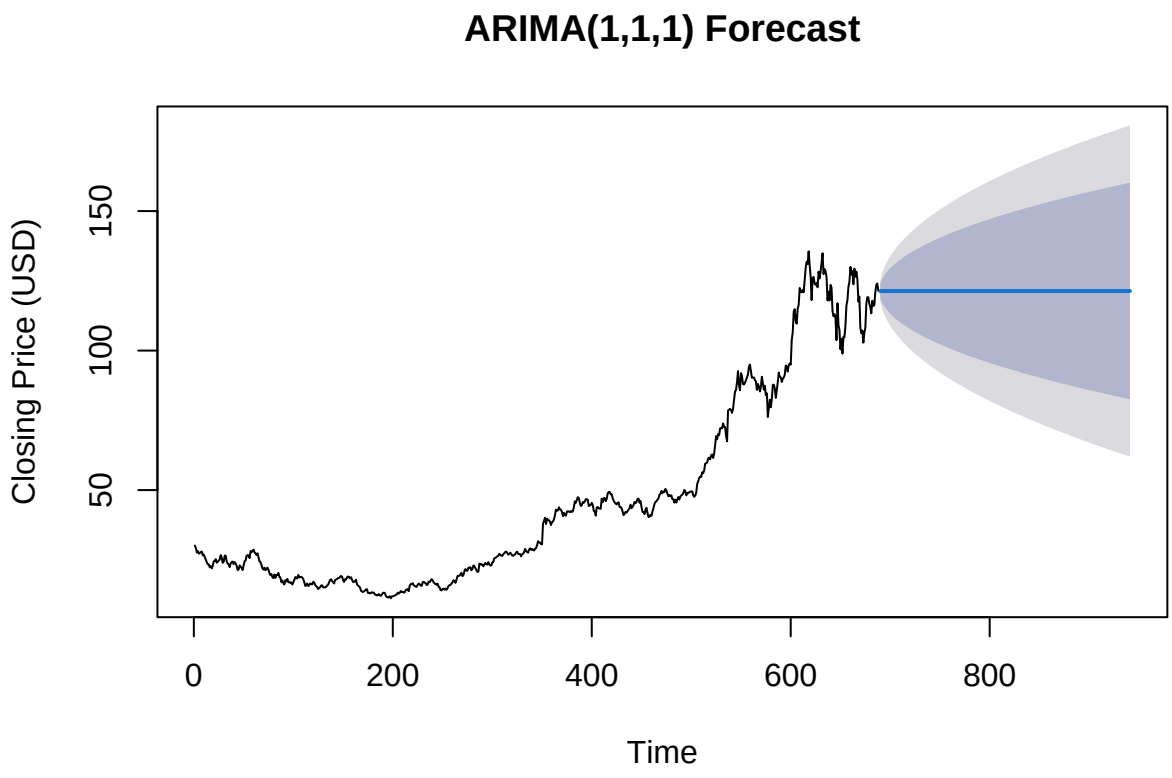

```
summary(arima(NVDAts, order=c(2,1,1)))
```

```
##
## Call:
```

```
## arima(x = NVDAts, order = c(2, 1, 1))
##
## Coefficients:
##           ar1     ar2     ma1
##       -0.6579  0.0304  0.5763
## s.e.   0.1603  0.0462  0.1562
##
## sigma^2 estimated as 4.078:  log likelihood = -1459.79,  aic = 2927.58
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE     MASE
## Training set 0.1367091 2.017975 1.208436 0.1453161 2.671269 1.000218
##                       ACF1
## Training set -0.004775394
```

Auto arima to confirm (1,1,1)

```
NVDASarima <- arima(NVDAts, order = c(1,1,1))
FcastNVDA <- forecast(NVDASarima, h=252)
plot(FcastNVDA, main="ARIMA(1,1,1) Forecast",
     ylab="Closing Price (USD)", xlab="Time")
```

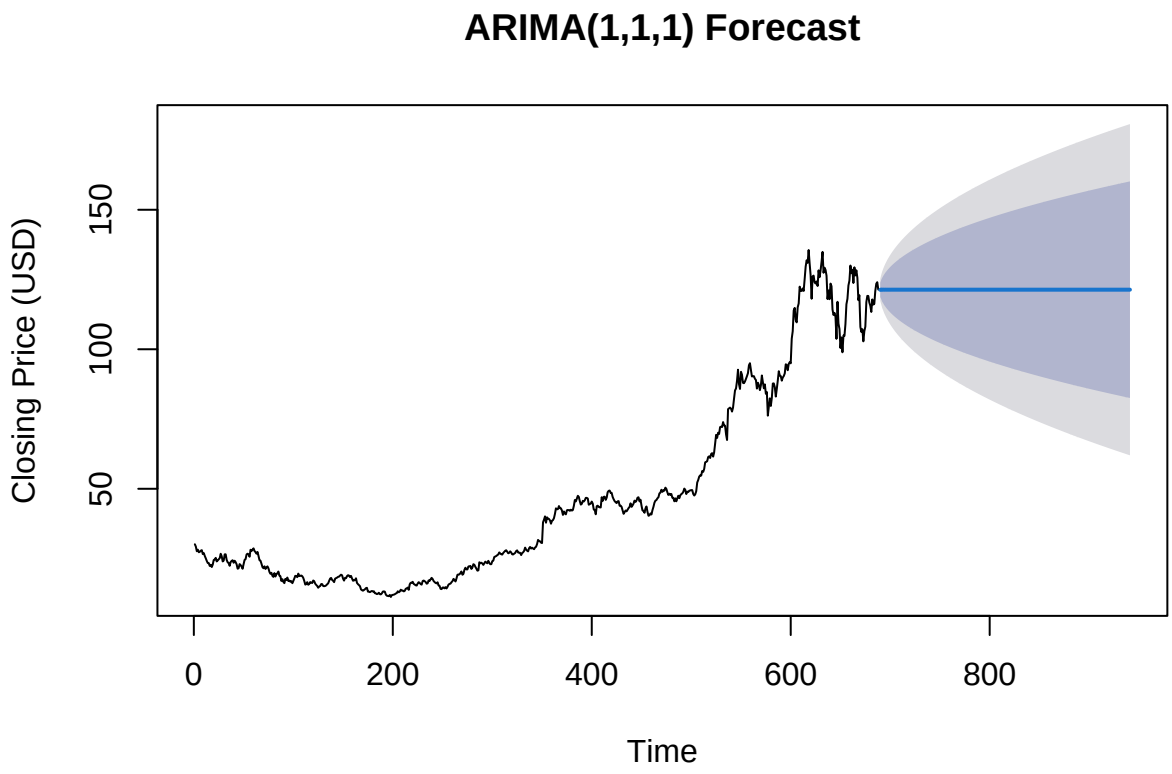


Best Model: Arima (1,1,1) with drift

Nvidia's stock price follows a first-order difference ARIMA process with one AR term and one MA term. Drift term indicates a consistent upward trend in Nvidia stock over time. Model Residual Diagnostics - Residuals from ARIMA(1,1,1) with drift: - Ljung-Box test (p = 0.033) suggests minor residual autocorrelation - Residuals appear to be randomly distributed.

Sarima Forecast

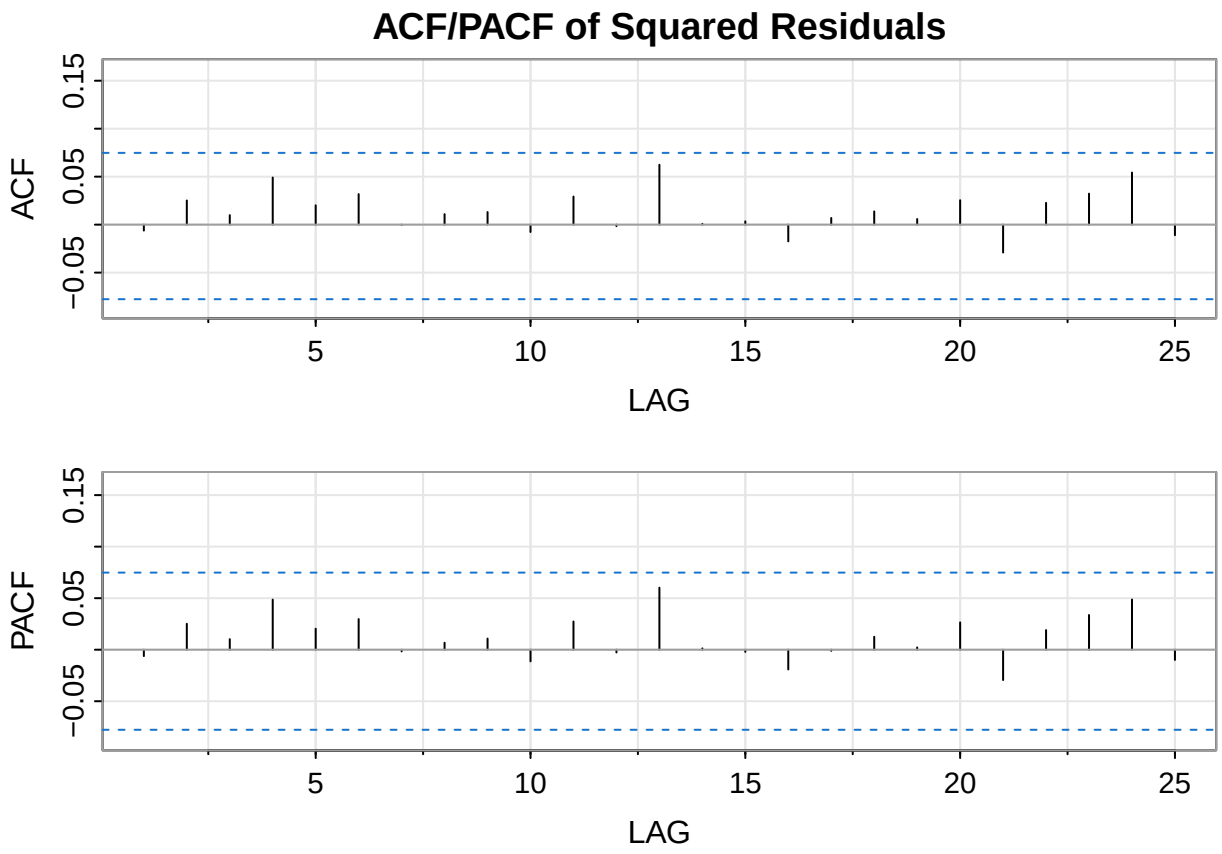
```
FcastNVDA <- forecast(NVDASarima, h=252)
plot(FcastNVDA, main="ARIMA(1,1,1) Forecast",
     ylab="Closing Price (USD)", xlab="Time")
```



252 day (1-year) forecast using SARIMA (1,1,1) with drift. Forecasted values show continued growth. Confidence Intervals (80%, 95%) widen over time, meaning uncertainty increases as we project further. Nvidia's stock is projected to continue its strong growth trend. Potential volatility is reflected in the widening confidence bands.

Garch

```
NVDAarma <- arima(NVDALogDiff, order = c(1,1,1))
residuals <- residuals(NVDAarma)
squaredresiduals = residuals^2
acf2(squaredresiduals, 25, main = "ACF/PACF of Squared Residuals")
```




```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## ACF  -0.01 0.03 0.01 0.05 0.02 0.03    0 0.01 0.01 -0.01  0.03     0  0.06
## PACF -0.01 0.03 0.01 0.05 0.02 0.03    0 0.01 0.01 -0.01  0.03     0  0.06
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF      0     0 -0.02  0.01  0.01  0.01  0.03 -0.03  0.02  0.03  0.05 -0.01
## PACF     0     0 -0.02  0.00  0.01  0.00  0.03 -0.03  0.02  0.03  0.05 -0.01
```

```
library(fGarch)

NVDAGarch1 <- garchFit(~ arma(1,1) + garch(1,1), data=NVDALogDiff, cond.dist="std")
```

```
##
## Series Initialization:
##  ARMA Model:                 arma
##  Formula Mean:               ~ arma(1, 1)
##  GARCH Model:                garch
##  Formula Variance:           ~ garch(1, 1)
##  ARMA Order:                 1 1
##  Max ARMA Order:             1
##  GARCH Order:                1 1
##  Max GARCH Order:            1
##  Maximum Order:              1
##  Conditional Dist:           std
##  h.start:                    2
##  llh.start:                  1
```

```
##  Length of Series:          688
##  Recursion Init:            mci
##  Series Scale:              0.03529361

## Warning in arima(.series$x, order = c(u, 0, v), include.mean = include.mean):
## possible convergence problem: optim gave code = 1

## Parameter Initialization:
##  Initial Parameters:          $params
##  Limits of Transformations:   $U, $V
##  Which Parameters are Fixed?  $includes
##  Parameter Matrix:
##                       U           V      params includes
##     mu     -0.57416913   0.5741691  0.05831408     TRUE
##     ar1    -0.99999999   1.0000000  0.97400281     TRUE
##     ma1    -0.99999999   1.0000000 -0.96903928     TRUE
##     omega   0.00000100 100.0000000  0.10000000     TRUE
##     alpha1  0.00000001   1.0000000  0.10000000     TRUE
##     gamma1 -0.99999999   1.0000000  0.10000000    FALSE
##     beta1   0.00000001   1.0000000  0.80000000     TRUE
##     delta   0.00000000   2.0000000  2.00000000    FALSE
##     skew    0.10000000  10.0000000  1.00000000    FALSE
##     shape   1.00000000  10.0000000  4.00000000     TRUE
##  Index List of Parameters to be Optimized:
##     mu    ar1    ma1  omega alpha1  beta1  shape
##      1      2      3      4      5      7     10
##  Persistence:                 0.9
##
##
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
##
##   0:     1662.8261: 0.0583141 0.974003 -0.969039 0.100000 0.100000 0.800000  4.00000
##   1:     1193.7805: 0.0564181 0.931737 0.0205300 0.130972 0.207660 0.872920  4.00295
##   2:     1148.7285: 0.0564101 0.901805 -0.00822108 0.109073 0.177181 0.792403  4.00168
##   3:     1109.7208: 0.0563893 0.836405 -0.0678371 0.128890 0.199255 0.822414  4.00364
##   4:     1064.8578: 0.0563449 0.714476 -0.175288 0.0878502 0.153290 0.731571  4.00298
##   5:     960.07520: 0.0558363 0.371643 -0.272377 0.240016 0.207637 0.705249  4.00654
##   6:     956.25168: 0.0558183 0.349807 -0.292887 0.205479 0.177480 0.664285  4.00534
##   7:     953.36420: 0.0557256 0.321519 -0.303948 0.219542 0.151580 0.717690  4.01113
##   8:     952.40745: 0.0556081 0.306731 -0.305884 0.186785 0.101805 0.746442  4.01981
##   9:     950.73233: 0.0554260 0.300469 -0.304619 0.143555 0.113445 0.795897  4.03316
##  10:     949.07357: 0.0544666 0.314003 -0.302518 1.00000e-06 0.0893260 0.927885  4.12720
##  11:     947.98586: 0.0544667 0.313988 -0.302543 0.00519586 0.0902141 0.930287  4.12725
##  12:     947.35147: 0.0544437 0.314933 -0.302524 0.00493966 0.0853548 0.929015  4.12993
##  13:     946.60234: 0.0543899 0.317321 -0.302451 0.00864037 0.0783306 0.934128  4.13613
##  14:     945.51800: 0.0542786 0.322372 -0.302259 0.00720801 0.0614869 0.942228  4.14862
##  15:     944.64362: 0.0541703 0.326782 -0.302103 0.00820514 0.0469510 0.955022  4.16036
##  16:     944.56815: 0.0541376 0.327570 -0.302130 0.00693113 0.0412339 0.957768  4.16394
##  17:     944.36806: 0.0540923 0.327557 -0.302158 0.00742540 0.0378767 0.961807  4.16915
##  18:     944.30871: 0.0540396 0.326211 -0.302292 0.00737879 0.0371968 0.961699  4.17639
##  19:     944.23541: 0.0539211 0.321852 -0.302765 0.00841393 0.0406235 0.958211  4.19473
```

```
##  20:     944.03115: 0.0535823 0.315256 -0.304099 0.00862662 0.0356536 0.961661  4.24848
##  21:     944.02743: 0.0535822 0.315264 -0.304092 0.00848399 0.0356372 0.961592  4.24849
##  22:     944.02463: 0.0535821 0.315279 -0.304080 0.00843029 0.0357622 0.961671  4.24851
##  23:     944.02185: 0.0535810 0.315318 -0.304072 0.00824690 0.0357932 0.961621  4.24877
##  24:     944.01757: 0.0535781 0.315387 -0.304080 0.00827914 0.0358902 0.961696  4.24939
##  25:     943.70569: 0.0529749 0.328526 -0.307002 0.0105031 0.0356698 0.958934  4.37814
##  26:     943.27002: 0.0524536 0.333140 -0.312795 0.00752545 0.0332346 0.963421  4.50740
##  27:     942.83337: 0.0473582 0.372240 -0.361303 0.00558741 0.0274266 0.966388  5.01563
##  28:     941.61055: 0.0405979 0.428938 -0.431073 0.00740739 0.0320030 0.962106  5.51373
##  29:     941.43117: 0.0429375 0.493606 -0.495441 0.00665819 0.0321077 0.962355  5.71185
##  30:     941.09863: 0.0238317 0.647655 -0.639649 0.00470854 0.0308558 0.964144  6.29271
##  31:     940.41485: 0.00873360 0.892485 -0.889682 0.00648438 0.0245586 0.967406  6.83191
##  32:     940.34817: 0.00809672 0.902100 -0.899686 0.00679945 0.0247498 0.967831  6.85136
##  33:     940.31434: 0.00746382 0.911620 -0.909592 0.00662716 0.0245045 0.967807  6.87095
##  34:     940.30442: 0.00621000 0.929455 -0.928521 0.00677110 0.0244150 0.968179  6.91124
##  35:     940.20160: 0.00671596 0.916477 -0.917430 0.00627114 0.0250762 0.968084  6.90253
##  36:     940.18386: 0.00671598 0.916453 -0.917457 0.00610193 0.0250152 0.967970  6.90253
##  37:     940.15411: 0.00671629 0.916158 -0.917804 0.00570381 0.0256276 0.968053  6.90253
##  38:     940.14638: 0.00667822 0.916581 -0.918363 0.00561709 0.0255834 0.967984  6.90396
##  39:     940.13555: 0.00659895 0.917303 -0.919661 0.00553695 0.0258663 0.968017  6.90675
##  40:     940.12296: 0.00643746 0.919148 -0.921887 0.00549032 0.0257758 0.967948  6.91237
##  41:     940.10223: 0.00615804 0.922837 -0.926877 0.00551430 0.0259536 0.967913  6.92373
##  42:     940.08843: 0.00650253 0.922501 -0.927711 0.00553884 0.0260731 0.967636  6.91582
##  43:     940.07518: 0.00608700 0.927144 -0.932766 0.00555341 0.0260909 0.967437  6.91593
##  44:     940.01497: 0.00465478 0.945310 -0.951912 0.00566377 0.0281335 0.965558  6.79752
##  45:     939.87029: 0.00318364 0.964661 -0.972111 0.00574374 0.0278632 0.965869  6.67958
##  46:     939.74601: 0.00284218 0.969185 -0.976813 0.00572068 0.0277516 0.965903  6.65777
##  47:     939.22682: 0.00216086 0.978314 -0.986279 0.00566734 0.0275073 0.965980  6.61416
##  48:     939.08772: 0.00190770 0.981703 -0.989802 0.00565054 0.0274193 0.966011  6.59796
##  49:     939.02507: 0.00194777 0.981207 -0.991005 0.00607790 0.0277216 0.966478  6.59796
##  50:     938.82655: 0.00200596 0.980834 -0.990949 0.00560521 0.0273714 0.966075  6.59696
##  51:     938.70878: 0.00202224 0.981597 -0.992268 0.00587694 0.0275627 0.966401  6.59412
##  52:     938.57416: 0.00204435 0.982144 -0.993303 0.00544146 0.0272492 0.966089  6.58773
##  53:     938.50899: 0.00235560 0.978709 -0.992073 0.00543276 0.0273859 0.966135  6.58754
##  54:     938.44905: 0.00235501 0.978896 -0.992620 0.00572121 0.0275983 0.966418  6.58754
##  55:     938.37388: 0.00235680 0.978997 -0.992936 0.00545152 0.0273776 0.966204  6.58702
##  56:     938.25699: 0.00235819 0.979262 -0.993936 0.00573625 0.0275817 0.966516  6.58609
##  57:     938.12443: 0.00236924 0.979406 -0.994466 0.00545212 0.0273517 0.966311  6.58323
##  58:     937.86237: 0.00235587 0.979655 -0.996886 0.00572615 0.0275202 0.966685  6.58322
##  59:     937.69646: 0.00239263 0.980213 -0.997303 0.00519755 0.0271093 0.966263  6.58105
##  60:     937.58991: 0.00241080 0.980253 -0.997942 0.00552277 0.0273807 0.966577  6.57873
##  61:     937.53067: 0.00243581 0.980292 -0.998360 0.00526827 0.0271844 0.966403  6.57634
##  62:     937.48214: 0.00245883 0.980219 -0.998652 0.00546222 0.0273429 0.966620  6.57392
##  63:     937.41645: 0.00248609 0.980020 -0.998852 0.00524443 0.0271740 0.966536  6.56897
##  64:     937.33251: 0.00252472 0.979695 -0.999380 0.00532956 0.0272217 0.966800  6.55904
##  65:     937.33247: 0.00259542 0.979092 -0.999342 0.00533710 0.0273384 0.967053  6.54917
##  66:     937.16333: 0.00260302 0.979036 -0.999994 0.00511931 0.0272021 0.966926  6.54424
##  67:     937.15518: 0.00262770 0.979034 -1.00000 0.00503806 0.0271070 0.966938  6.53927
##  68:     937.13645: 0.00262244 0.978801 -1.00000 0.00505738 0.0271049 0.967035  6.53430
##  69:     937.12456: 0.00263007 0.978649 -1.00000 0.00477025 0.0270419 0.967411  6.49176
##  70:     937.11328: 0.00262740 0.978742 -1.00000 0.00462637 0.0270608 0.967731  6.44923
##  71:     937.07769: 0.00262100 0.978864 -1.00000 0.00438191 0.0268913 0.968215  6.27909
##  72:     937.05453: 0.00262762 0.978787 -1.00000 0.00481463 0.0267704 0.968103  6.10894
##  73:     937.05220: 0.00262450 0.978798 -1.00000 0.00478103 0.0267847 0.968105  6.02524
```

```
##  74:     937.05187: 0.00262528 0.978788 -1.00000 0.00479592 0.0266945 0.968174  6.04954
##  75:     937.05187: 0.00262535 0.978787 -1.00000 0.00479960 0.0267202 0.968147  6.04836
##  76:     937.05187: 0.00262533 0.978788 -1.00000 0.00479878 0.0267157 0.968152  6.04827
##  77:     937.05187: 0.00262533 0.978788 -1.00000 0.00479877 0.0267157 0.968152  6.04829
##
## Final Estimate of the Negative LLH:
##  LLH:  -1363.657    norm LLH:  -1.982059
##           mu           ar1           ma1         omega        alpha1
##  9.265736e-05  9.787876e-01 -1.000000e+00  5.977534e-06  2.671566e-02
##         beta1         shape
##  9.681516e-01  6.048286e+00
##
## R-optimhess Difference Approximated Hessian Matrix:
##                   mu           ar1           ma1         omega        alpha1
## mu     -8.457455e+10 -1.047389e+08 -2.853542e+08  1.951522e+09  2.331045e+06
## ar1    -1.047389e+08 -3.491215e+05 -2.801278e+05  2.325424e+06 -4.974216e+02
## ma1    -2.853542e+08 -2.801278e+05 -2.410851e+06  6.152000e+06  5.236270e+03
## omega   1.951522e+09  2.325424e+06  6.152000e+06 -2.134196e+11 -1.870356e+08
## alpha1  2.331045e+06 -4.974216e+02  5.236270e+03 -1.870356e+08 -2.073731e+05
## beta1   2.698071e+06  1.385636e+03  7.917071e+03 -2.125025e+08 -2.165977e+05
## shape   3.938765e+03 -1.000218e+01  4.551618e+00 -2.184262e+05 -2.505892e+02
##                beta1         shape
## mu      2.698071e+06  3.938765e+03
## ar1     1.385636e+03 -1.000218e+01
## ma1     7.917071e+03  4.551618e+00
## omega  -2.125025e+08 -2.184262e+05
## alpha1 -2.165977e+05 -2.505892e+02
## beta1  -2.407305e+05 -2.543539e+02
## shape  -2.543539e+02 -1.035997e+00
## attr(,"time")
## Time difference of 0.048527 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.304565 secs
```

```
NVDAGarch2 <- garchFit(~ arma(1,1) + garch(1,0), data=NVDALogDiff, cond.dist="std")
```

```
##
## Series Initialization:
##  ARMA Model:                arma
##  Formula Mean:              ~ arma(1, 1)
##  GARCH Model:               garch
##  Formula Variance:          ~ garch(1, 0)
##  ARMA Order:                1 1
##  Max ARMA Order:            1
##  GARCH Order:               1 0
##  Max GARCH Order:           1
##  Maximum Order:             1
##  Conditional Dist:          std
##  h.start:                   2
##  llh.start:                 1
```

```
##  Length of Series:          688
##  Recursion Init:            mci
##  Series Scale:              0.03529361

## Warning in arima(.series$x, order = c(u, 0, v), include.mean = include.mean):
## possible convergence problem: optim gave code = 1

## Parameter Initialization:
##  Initial Parameters:          $params
##  Limits of Transformations:   $U, $V
##  Which Parameters are Fixed?  $includes
##  Parameter Matrix:
##                       U           V      params includes
##     mu     -0.57416913   0.5741691  0.05831408     TRUE
##     ar1    -0.99999999   1.0000000  0.97400281     TRUE
##     ma1    -0.99999999   1.0000000 -0.96903928     TRUE
##     omega   0.00000100 100.0000000  0.10000000     TRUE
##     alpha1  0.00000001   1.0000000  0.10000000     TRUE
##     gamma1 -0.99999999   1.0000000  0.10000000    FALSE
##     delta   0.00000000   2.0000000  2.00000000    FALSE
##     skew    0.10000000  10.0000000  1.00000000    FALSE
##     shape   1.00000000  10.0000000  4.00000000     TRUE
##  Index List of Parameters to be Optimized:
##     mu    ar1    ma1  omega alpha1  shape
##      1      2      3      4      5      9
##  Persistence:                  0.1
##
##
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
##
##   0:     2809.0318: 0.0583141 0.974003 -0.969039 0.100000 0.100000  4.00000
##   1:     1483.2630: 0.0564282 0.935032 0.0147162 0.184445 0.249989  3.99972
##   2:     1099.9409: 0.0563481 0.757194 -0.125651 0.584862 0.445571  4.00883
##   3:     975.69368: 0.0561011 0.381734 -0.410651 0.737803 0.508563  4.02948
##   4:     970.10899: 0.0550123 0.507308 -0.341143  1.09222 0.195130  4.10149
##   5:     960.80564: 0.0548727 0.423680 -0.413184  1.08827 0.179042  4.10238
##   6:     958.30714: 0.0543038 0.411275 -0.413958  1.06400 0.0712838  4.10580
##   7:     957.20205: 0.0519932 0.411457 -0.380924  1.11247 0.00518232  4.16090
##   8:     957.00950: 0.0518968 0.399017 -0.391122  1.11531 0.00287037  4.16303
##   9:     956.93213: 0.0513896 0.402163 -0.384279  1.12037 0.00108184  4.17379
##  10:     956.81955: 0.0501821 0.393071 -0.388813  1.11974 1.00000e-08  4.19752
##  11:     955.87712: 0.0341735 0.383924 -0.369791  1.04742 1.00000e-08  4.53421
##  12:     955.02779: 0.0311291 0.485323 -0.473426  1.03590 1.00000e-08  4.94904
##  13:     954.69764: 0.0137032 0.627923 -0.619891  1.04534 1.00000e-08  5.19961
##  14:     954.25332: 0.0100004 0.743976 -0.742172  1.00153 1.00000e-08  5.60244
##  15:     954.04415: 0.00886885 0.792846 -0.787665 0.991798 1.00000e-08  5.73101
##  16:     953.74011: 0.00646283 0.890009 -0.881255 0.972156 1.00000e-08  5.98701
##  17:     953.43522: 0.00643982 0.890765 -0.896268 0.971909 1.00000e-08  6.01412
##  18:     953.30533: 0.00573429 0.916695 -0.922563 0.970150 1.00000e-08  6.08185
##  19:     953.28607: 0.00545101 0.916416 -0.924678 0.981503 0.0100092  6.08246
##  20:     953.26869: 0.00545475 0.918356 -0.923264 0.981578 0.00924064  6.08247
```

```
##  21:     953.25621: 0.00539049 0.918548 -0.925086 0.981944 0.00802038  6.08283
##  22:     953.24137: 0.00524755 0.921799 -0.926707 0.982730 0.00592698  6.08386
##  23:     953.21951: 0.00490117 0.925720 -0.932606 0.983909 0.00400875  6.08704
##  24:     953.20442: 0.00457342 0.931310 -0.936761 0.984909 0.00270377  6.09128
##  25:     953.19333: 0.00439004 0.935571 -0.942743 0.987848 1.00000e-08  6.09492
##  26:     953.18504: 0.00416928 0.939014 -0.945312 0.987322 0.000642737  6.10316
##  27:     953.12948: 0.00259692 0.962409 -0.969477 0.982396 0.00497351  6.17090
##  28:     952.96626: 0.00208063 0.974157 -0.978433 0.980529 0.00631537  6.19716
##  29:     952.85549: 0.00197180 0.975969 -0.981240 0.980122 0.00659142  6.20281
##  30:     952.74665: 0.00187429 0.978071 -0.990187 0.979383 0.00687830  6.21280
##  31:     952.43309: 0.00189346 0.982100 -0.989910 0.979135 0.00705192  6.21617
##  32:     952.24533: 0.00201715 0.979487 -0.989597 0.979411 0.00678542  6.21215
##  33:     952.03098: 0.00221162 0.981759 -0.992946 0.979343 0.00663842  6.21214
##  34:     951.83135: 0.00217998 0.979899 -0.994254 0.979312 0.00663636  6.21238
##  35:     951.73070: 0.00220242 0.981271 -0.995205 0.979180 0.00670480  6.21399
##  36:     951.66995: 0.00227635 0.979918 -0.994867 0.979256 0.00661420  6.21260
##  37:     951.58252: 0.00233609 0.981071 -0.996541 0.979173 0.00661292  6.21316
##  38:     951.51514: 0.00233191 0.980503 -0.996795 0.979166 0.00660963  6.21316
##  39:     951.45519: 0.00242133 0.980054 -0.996950 0.979124 0.00658669  6.21315
##  40:     951.35799: 0.00245058 0.979759 -0.998236 0.979031 0.00660731  6.21396
##  41:     951.31295: 0.00251669 0.979984 -0.998902 0.978929 0.00663229  6.21487
##  42:     951.28498: 0.00251448 0.979651 -0.999011 0.978924 0.00663083  6.21486
##  43:     951.27540: 0.00253114 0.979578 -0.998825 0.978917 0.00662854  6.21486
##  44:     951.25437: 0.00253601 0.979441 -0.999105 0.978898 0.00663168  6.21500
##  45:     951.22987: 0.00256856 0.979399 -0.999345 0.978854 0.00663916  6.21534
##  46:     951.21552: 0.00257734 0.979287 -0.999640 0.978828 0.00664501  6.21556
##  47:     951.20363: 0.00259616 0.979306 -0.999714 0.978802 0.00665180  6.21579
##  48:     951.19338: 0.00260101 0.979175 -0.999894 0.978786 0.00665502  6.21592
##  49:     951.18733: 0.00260362 0.979085 -0.999711 0.978792 0.00665062  6.21581
##  50:     951.17865: 0.00260856 0.979019 -0.999893 0.978779 0.00665289  6.21591
##  51:     951.17046: 0.00263047 0.979051 -1.00000 0.978754 0.00665756  6.21610
##  52:     951.15368: 0.00263354 0.978712 -1.00000 0.978317 0.00703659  6.21553
##  53:     951.15313: 0.00263899 0.978726 -1.00000 0.978192 0.00707412  6.21466
##  54:     951.15172: 0.00263688 0.978686 -1.00000 0.978080 0.00710573  6.21379
##  55:     951.12161: 0.00262337 0.978959 -1.00000 0.966097 0.0105972  6.11218
##  56:     951.07154: 0.00264239 0.978565 -1.00000 0.969448 0.00973355  5.86731
##  57:     951.05361: 0.00263245 0.978771 -1.00000 0.987386 0.00652783  5.63742
##  58:     951.04943: 0.00263805 0.978683 -1.00000 0.986321 0.0115517  5.65988
##  59:     951.04817: 0.00263677 0.978703 -1.00000 0.985853 0.00984022  5.67531
##  60:     951.04816: 0.00263677 0.978703 -1.00000 0.985867 0.00978060  5.67142
##  61:     951.04816: 0.00263677 0.978703 -1.00000 0.985883 0.00977678  5.67217
##  62:     951.04816: 0.00263677 0.978703 -1.00000 0.985883 0.00977782  5.67215
##
## Final Estimate of the Negative LLH:
##  LLH:  -1349.661    norm LLH:  -1.961716
##           mu           ar1           ma1         omega        alpha1
##  9.306107e-05  9.787031e-01 -1.000000e+00  1.228054e-03  9.777823e-03
##         shape
##  5.672149e+00
##
## R-optimhess Difference Approximated Hessian Matrix:
##                   mu           ar1           ma1         omega        alpha1
## mu     -1.003558e+11 -1.205042e+08 -3.185908e+08  8.862217e+06  9.686639e+04
## ar1    -1.205042e+08 -3.470294e+05 -2.189197e+05 -4.773575e+04  1.619562e+01
```

```
## ma1    -3.185908e+08 -2.189197e+05 -1.998785e+06 -5.735334e+04  2.691364e+02
## omega   8.862217e+06 -4.773575e+04 -5.735334e+04 -1.466121e+08 -1.715144e+05
## alpha1  9.686639e+04  1.619562e+01  2.691364e+02 -1.715144e+05 -7.819999e+02
## shape   2.654838e+03 -1.579586e+01  1.244720e+01 -7.898314e+03 -9.888643e+00
##               shape
## mu      2654.838125
## ar1      -15.795857
## ma1       12.447204
## omega  -7898.314287
## alpha1    -9.888643
## shape     -1.150776
## attr(,"time")
## Time difference of 0.0392828 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.2311332 secs
```

```
NVDAGarch3 <- garchFit(~ arma(0,0) + garch(1,0), data=NVDALogDiff, cond.dist="std")
```

```
##
## Series Initialization:
##  ARMA Model:                arma
##  Formula Mean:              ~ arma(0, 0)
##  GARCH Model:               garch
##  Formula Variance:          ~ garch(1, 0)
##  ARMA Order:                0 0
##  Max ARMA Order:            0
##  GARCH Order:               1 0
##  Max GARCH Order:           1
##  Maximum Order:             1
##  Conditional Dist:          std
##  h.start:                   2
##  llh.start:                 1
##  Length of Series:          688
##  Recursion Init:            mci
##  Series Scale:              0.03529361
##
## Parameter Initialization:
##  Initial Parameters:          $params
##  Limits of Transformations:   $U, $V
##  Which Parameters are Fixed?  $includes
##  Parameter Matrix:
##                      U           V     params includes
##     mu     -0.57416913   0.5741691 0.05741691     TRUE
##     omega   0.00000100 100.0000000 0.10000000     TRUE
##     alpha1  0.00000001   1.0000000 0.10000000     TRUE
##     gamma1 -0.99999999   1.0000000 0.10000000    FALSE
##     delta   0.00000000   2.0000000 2.00000000    FALSE
##     skew    0.10000000  10.0000000 1.00000000    FALSE
##     shape   1.00000000  10.0000000 4.00000000     TRUE
##  Index List of Parameters to be Optimized:
```

```
##     mu  omega alpha1  shape
##      1      2      3      7
##  Persistence:                  0.1
##
##
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
##
##   0:     1460.4614: 0.0574169 0.100000 0.100000  4.00000
##   1:     968.74229: 0.0574723  1.05039 0.411040  4.00337
##   2:     962.28616: 0.0637427  1.17946 1.00000e-08  3.43106
##   3:     957.54939: 0.0613026  1.18566 1.00000e-08  4.41410
##   4:     957.49453: 0.0370926  1.18095 1.00000e-08  4.47603
##   5:     957.29437: 0.0465481  1.17307 1.00000e-08  4.55243
##   6:     956.70398: 0.0640665  1.12897 1.00000e-08  4.95347
##   7:     955.94119: 0.0768831  1.03365 1.00000e-08  5.79527
##   8:     955.64942: 0.0669161 0.986874 1.00000e-08  6.23693
##   9:     955.52521: 0.0500968 0.992565 1.00000e-08  6.22647
##  10:     955.52437: 0.0495725 0.992677 1.00000e-08  6.23576
##  11:     955.52215: 0.0487437 0.992841 1.00000e-08  6.27863
##  12:     955.52089: 0.0488778 0.992646 1.00000e-08  6.32037
##  13:     955.52048: 0.0494553 0.992345 1.00000e-08  6.34379
##  14:     955.52045: 0.0496911 0.992260 1.00000e-08  6.34521
##  15:     955.52045: 0.0497262 0.992259 1.00000e-08  6.34422
##  16:     955.52045: 0.0497269 0.992259 1.00000e-08  6.34404
##
## Final Estimate of the Negative LLH:
##  LLH:  -1345.188    norm LLH:  -1.955215
##          mu       omega      alpha1       shape
## 0.001755043 0.001235997 0.000000010 6.344037254
##
## R-optimhess Difference Approximated Hessian Matrix:
##                  mu         omega        alpha1         shape
## mu     -638655.2126 -5.242655e+04  7.397152e+02   -14.9172956
## omega   -52426.5519 -1.531108e+08 -1.850829e+05 -5666.8815893
## alpha1     739.7152 -1.850829e+05 -9.725789e+02    -7.2244295
## shape      -14.9173 -5.666882e+03 -7.224429e+00    -0.6929013
## attr(,"time")
## Time difference of 0.01578689 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.05040097 secs
```

```
summary(NVDAGarch1)
```

```
##
## Title:
##  GARCH Modelling
##
```

```
## Call:
##  garchFit(formula = ~arma(1, 1) + garch(1, 1), data = NVDALogDiff,
##     cond.dist = "std")
##
## Mean and Variance Equation:
##  data ~ arma(1, 1) + garch(1, 1)
## <environment: 0x7fa02e4b12d0>
##  [data = NVDALogDiff]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##          mu          ar1          ma1        omega       alpha1        beta1
##  9.2657e-05   9.7879e-01  -1.0000e+00   5.9775e-06   2.6716e-02   9.6815e-01
##       shape
##  6.0483e+00
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error   t value Pr(>|t|)
## mu      9.266e-05   5.379e-06    17.226  < 2e-16 ***
## ar1     9.788e-01   2.155e-03   454.120  < 2e-16 ***
## ma1    -1.000e+00   8.383e-04 -1192.953  < 2e-16 ***
## omega   5.978e-06   6.397e-06     0.934  0.35009
## alpha1  2.672e-02   9.429e-03     2.833  0.00461 **
## beta1   9.682e-01   1.125e-02    86.020  < 2e-16 ***
## shape   6.048e+00   1.171e+00     5.164 2.42e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1363.657    normalized:  1.982059
##
## Description:
##  Wed Mar 19 12:44:02 2025 by user:
##
##
## Standardised Residuals Tests:
##                                   Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  1477.8862289 0.000000e+00
##  Shapiro-Wilk Test  R    W         0.9475199 6.670891e-15
##  Ljung-Box Test     R    Q(10)    11.1409513 3.466426e-01
##  Ljung-Box Test     R    Q(15)    16.3426154 3.596539e-01
##  Ljung-Box Test     R    Q(20)    22.6422531 3.067001e-01
##  Ljung-Box Test     R^2  Q(10)     1.4451644 9.990962e-01
##  Ljung-Box Test     R^2  Q(15)     2.5379077 9.998599e-01
##  Ljung-Box Test     R^2  Q(20)     3.7341779 9.999737e-01
##  LM Arch Test       R    TR^2      1.4923737 9.998730e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
```

```
## -3.943770 -3.897641 -3.943974 -3.925924
```

```
summary(NVDAGarch2)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(1, 1) + garch(1, 0), data = NVDALogDiff,
##     cond.dist = "std")
##
## Mean and Variance Equation:
##  data ~ arma(1, 1) + garch(1, 0)
## <environment: 0x7fa018389ee8>
##  [data = NVDALogDiff]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##          mu          ar1          ma1        omega       alpha1        shape
##  9.3061e-05   9.7870e-01  -1.0000e+00   1.2281e-03   9.7778e-03   5.6721e+00
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## mu      9.306e-05   6.101e-06   15.253  < 2e-16 ***
## ar1     9.787e-01   2.391e-03  409.293  < 2e-16 ***
## ma1    -1.000e+00   1.082e-03 -924.607  < 2e-16 ***
## omega   1.228e-03   1.140e-04   10.774  < 2e-16 ***
## alpha1  9.778e-03   4.150e-02    0.236    0.814
## shape   5.672e+00   1.176e+00    4.824 1.41e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1349.661    normalized:  1.961716
##
## Description:
##  Wed Mar 19 12:44:02 2025 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  248.5259790 0.000000e+00
##  Shapiro-Wilk Test  R    W        0.9736419 8.526258e-10
##  Ljung-Box Test     R    Q(10)   12.7344504 2.389003e-01
##  Ljung-Box Test     R    Q(15)   19.6746609 1.847715e-01
##  Ljung-Box Test     R    Q(20)   27.9628370 1.102820e-01
##  Ljung-Box Test     R^2  Q(10)    5.2184776 8.761137e-01
##  Ljung-Box Test     R^2  Q(15)    8.4114371 9.062310e-01
```

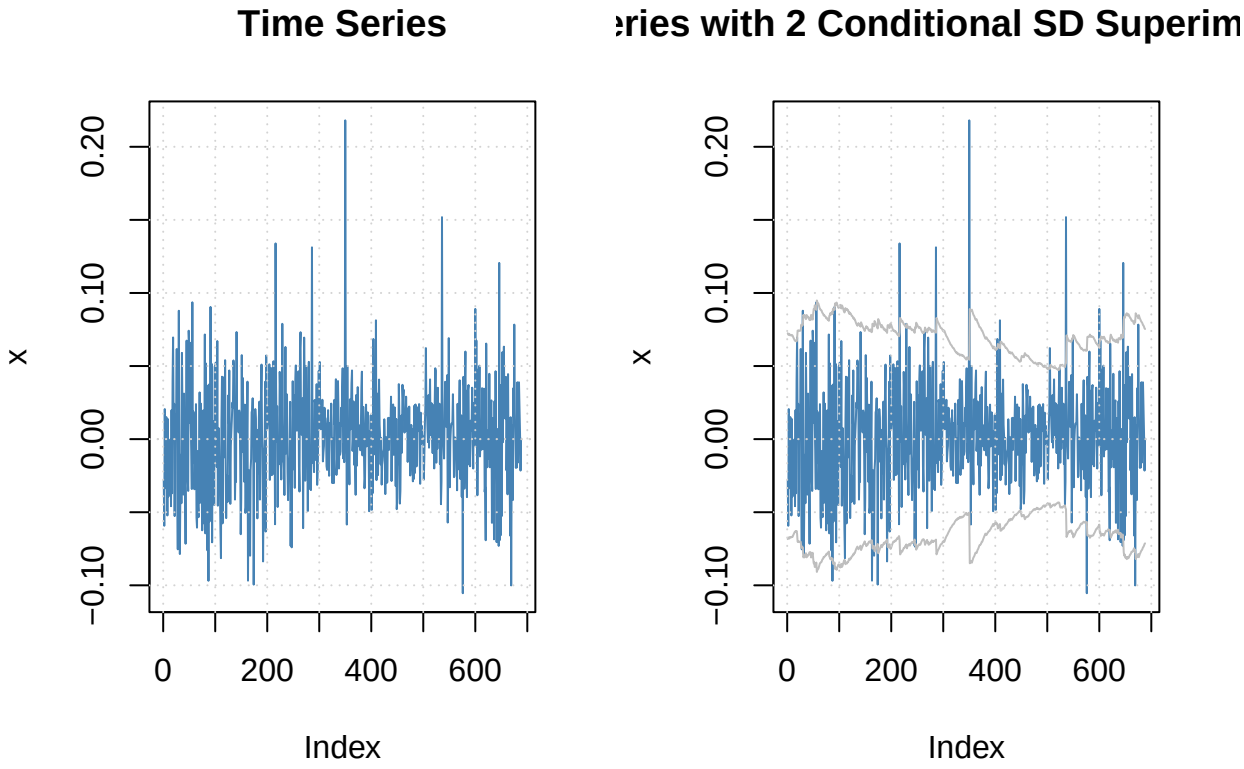
```
##  Ljung-Box Test     R^2  Q(20)    9.1589310 9.809923e-01
##  LM Arch Test       R    TR^2     5.6850164 9.311205e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.905990 -3.866451 -3.906140 -3.890694
```

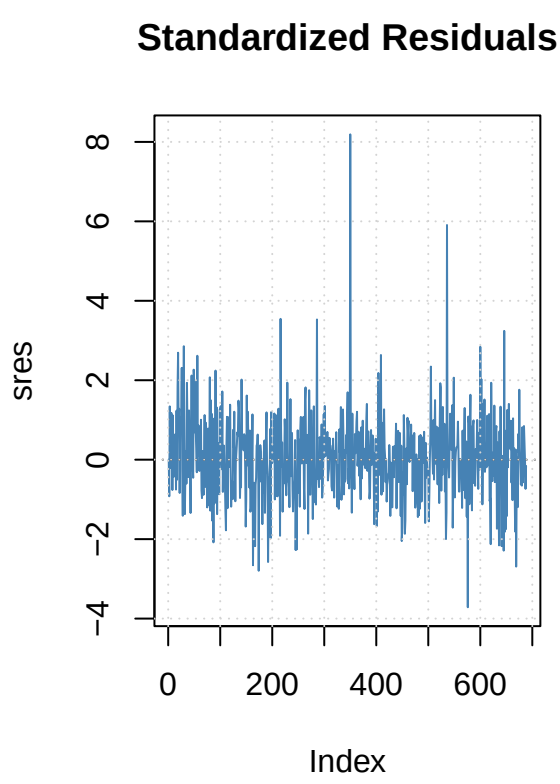
```
summary(NVDAGarch3)
```

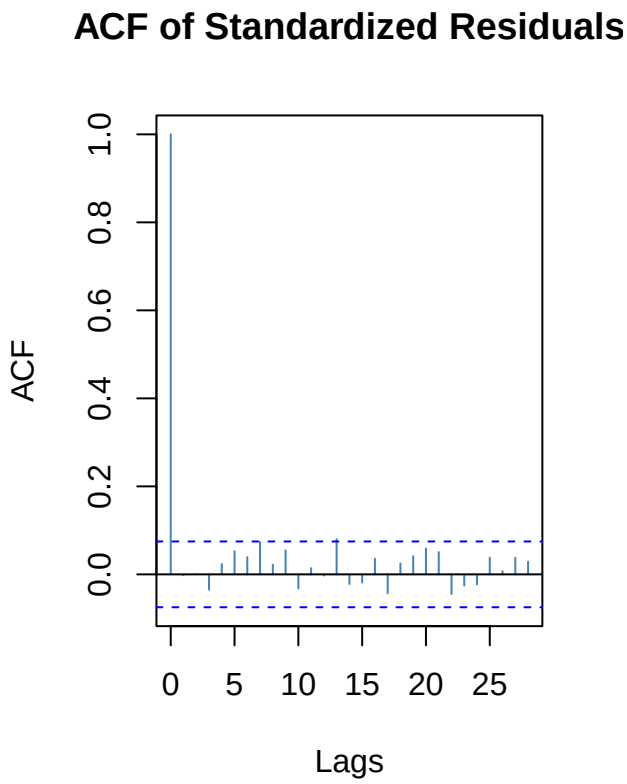
```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(0, 0) + garch(1, 0), data = NVDALogDiff,
##     cond.dist = "std")
##
## Mean and Variance Equation:
##  data ~ arma(0, 0) + garch(1, 0)
## <environment: 0x7fa030f641e0>
##  [data = NVDALogDiff]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##        mu      omega     alpha1      shape
## 1.755e-03  1.236e-03  1.000e-08  6.344e+00
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     1.755e-03   1.253e-03    1.401    0.161
## omega  1.236e-03   1.060e-04   11.665  < 2e-16 ***
## alpha1 1.000e-08   3.658e-02    0.000    1.000
## shape  6.344e+00   1.439e+00    4.408 1.05e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1345.188    normalized:  1.955215
##
## Description:
##  Wed Mar 19 12:44:03 2025 by user:
##
##
## Standardised Residuals Tests:
##                                   Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  236.4681988 0.000000e+00
##  Shapiro-Wilk Test  R    W        0.9752238 2.143006e-09
##  Ljung-Box Test     R    Q(10)   10.5910651 3.902518e-01
##  Ljung-Box Test     R    Q(15)   17.2052708 3.067422e-01
```

```
##  Ljung-Box Test     R    Q(20)   24.2339957 2.323202e-01
##  Ljung-Box Test     R^2  Q(10)    3.7223756 9.590033e-01
##  Ljung-Box Test     R^2  Q(15)    7.8580128 9.293488e-01
##  Ljung-Box Test     R^2  Q(20)    8.8678460 9.843880e-01
##  LM Arch Test       R    TR^2     4.2219629 9.790898e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.898803 -3.872444 -3.898870 -3.888606
```

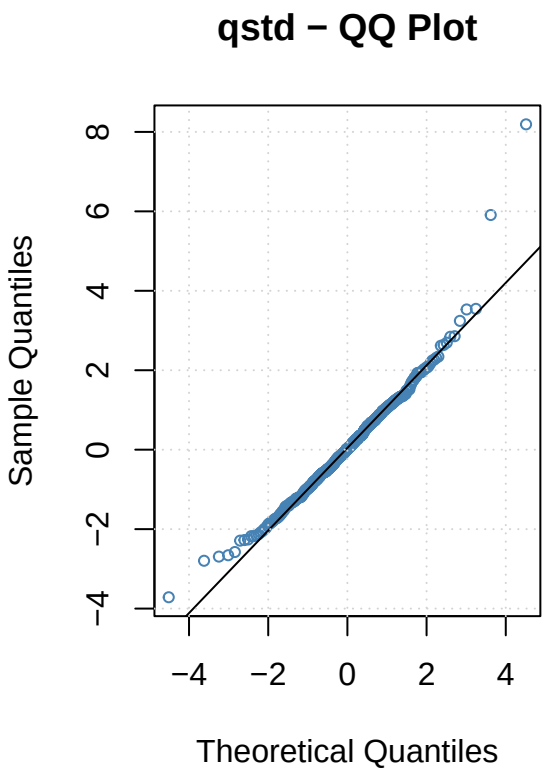
```
par(mfrow=c(1,2))
plot(NVDAGarch1, which = c(1,3,9,10))
```



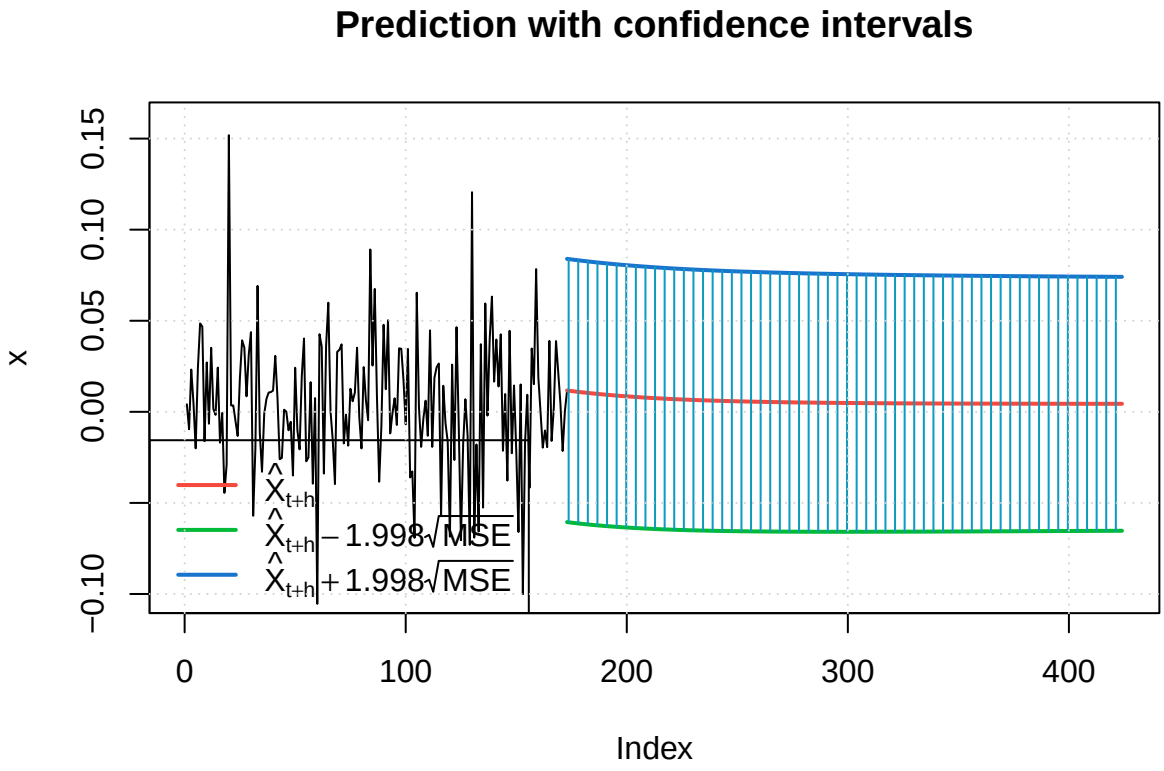





```
plot(NVDAGarch1, which =13)
```




```
ForecastGARCH <- predict(NVDAGarch1, n.ahead = 252, plot=TRUE)
```




```
summary(ForecastGARCH)
```

```
##   meanForecast         meanError        standardDeviation lowerInterval
##  Min.   :0.004402   Min.   :0.03488   Min.   :0.03470   Min.   :-0.06586
##  1st Qu.:0.004498   1st Qu.:0.03510   1st Qu.:0.03491   1st Qu.:-0.06577
##  Median :0.004866   Median :0.03540   Median :0.03521   Median :-0.06555
##  Mean   :0.005736   Mean   :0.03545   Mean   :0.03528   Mean   :-0.06510
##  3rd Qu.:0.006282   3rd Qu.:0.03579   3rd Qu.:0.03561   3rd Qu.:-0.06523
##  Max.   :0.011715   Max.   :0.03616   Max.   :0.03616   Max.   :-0.06054
##  upperInterval
##  Min.   :0.07410
##  1st Qu.:0.07463
##  Median :0.07559
##  Mean   :0.07657
##  3rd Qu.:0.07779
##  Max.   :0.08397
```