# Time Series Analysis of Nvidia Stock Prices


## Abstract:

This project focuses on analyzing the time series data of Nvidia's stock prices. The primary objective is to apply various time series modeling techniques, including SARIMA and GARCH modeling, to understand the patterns and forecast future values. The dataset spans several years, providing a comprehensive overview of Nvidia's market performance. This study aims to uncover significant trends and offer reliable predictions, which are crucial for investors and analysts.



**Daren Aguilera**

daren@ucsb.edu

## 1. Introduction:

Brief description of your project, the purpose, the reason why you chose the topic and data set, what have been studied in the past on the data set, what kind of methods you are applying, important discovery from the data set,

This project aims to analyze the stock prices of Nvidia using time series data. Nvidia, a leading technology company known for its graphics processing units (GPUs), has shown significant growth in the stock market. The purpose of this analysis is to apply time series models to understand the underlying patterns and forecast future stock prices. Financial market data is an excellent resource for practicing time series analysis, providing a consistent source of data and a generous timeframe. Previous studies have utilized similar methodologies for other tech giants who retain a large equity of total market sales. We will be conducting SARIMA and GARCH models, popular methods for modeling financial time series. By focusing on Nvidia, this project seeks to provide insights into its stock price behavior and identify key trends.

## 2. Data:

Describe the data set (time range, frequency, values, size of the data set, reason why you chose, web pages with webpage links, background of the data set, who collected the data, how the data set was collected, why the data set is important, the purpose of studying the data set, . . . )

We will practice working with the package `quantmod` to extract our desired data directly from Yahoo while staying within our R environment! The dataset used here consists of daily closing stock prices of Nvidia from January 3, 2017, to December 29, 2023 (since the market is publicly closed on holidays). The data was sourced with the help of `quantmod` from Yahoo Finance, which collects and provides historical financial data. The dataset includes the following columns: Date, Open, High, Low, Close, Adjusted Close, and Volume. The primary focus will be on the 'Close' prices for modeling and forecasting purposes.

## 3. Methodology:

### 3.1 SARIMA (p, d, q) x (P, D, Q) Model:

The Seasonal ARIMA (SARIMA) model will be applied to capture the seasonality and trend in Nvidia's stock prices. The parameters (p, d, q) and seasonal parameters (P, D, Q) will be determined using the Box-Jenkins approach, involving identification, estimation, and diagnostic checking.

### 3.2 GARCH Model:

Given the financial nature of the data, volatility modeling using GARCH (Generalized Autoregressive Conditional Heteroskedasticity) will be employed. This model helps in understanding and forecasting the volatility, which is crucial for risk management. We will be testing the adequacy of our model and deciding which to select for forecasting.

## 4. Results:

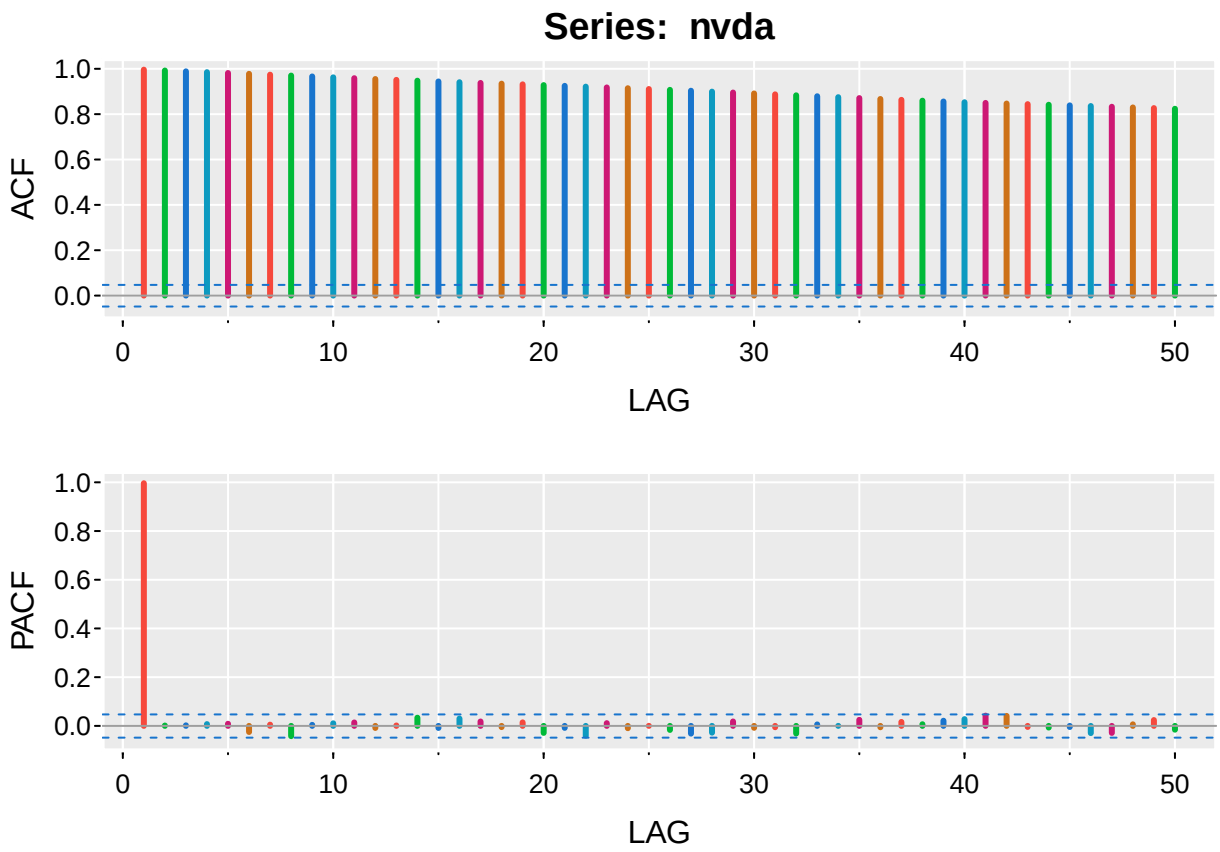
### 4.1 SARIMA Model:

We will be following the Box-Jenkins approach when building our SARIMA model.

**Plot the time series data:** Plotting the original time series, we get the following:


We notice a lot of heteroscedasticity with strong, year-long trends.

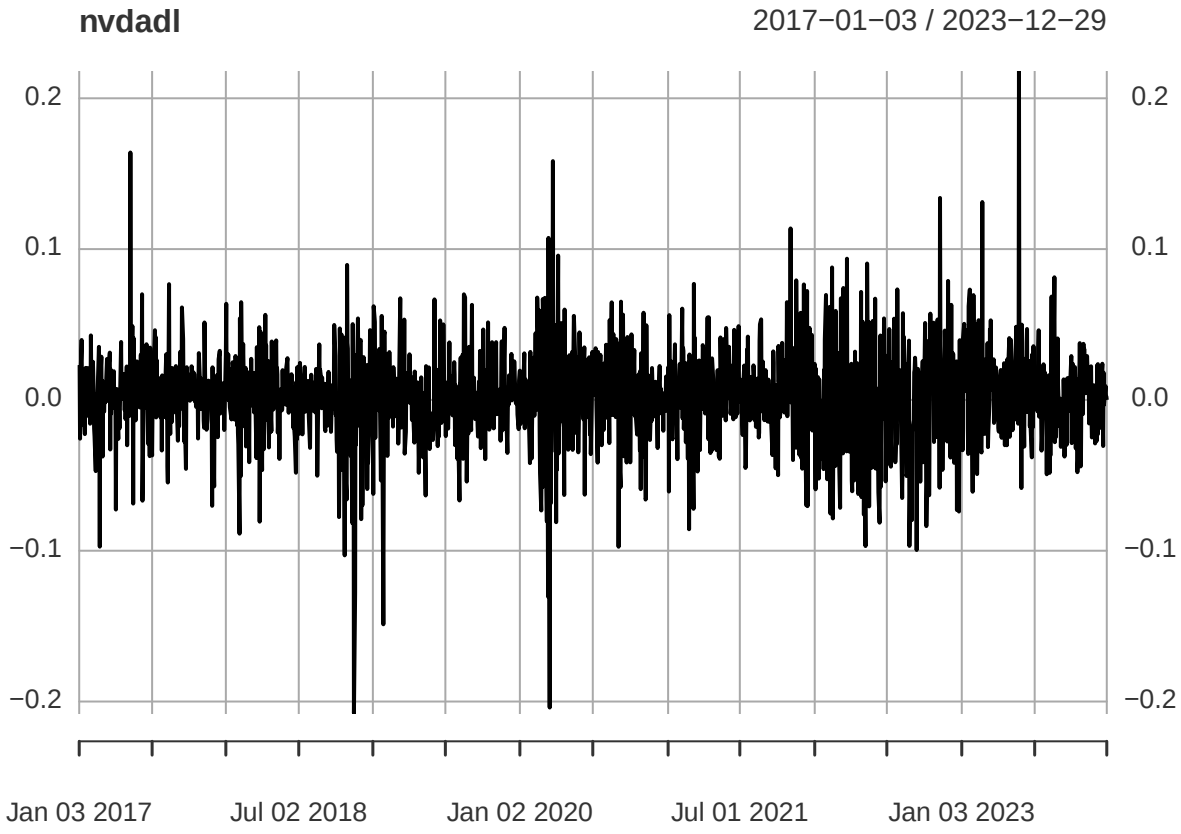
Next, we observe the ACF and PACF plots:

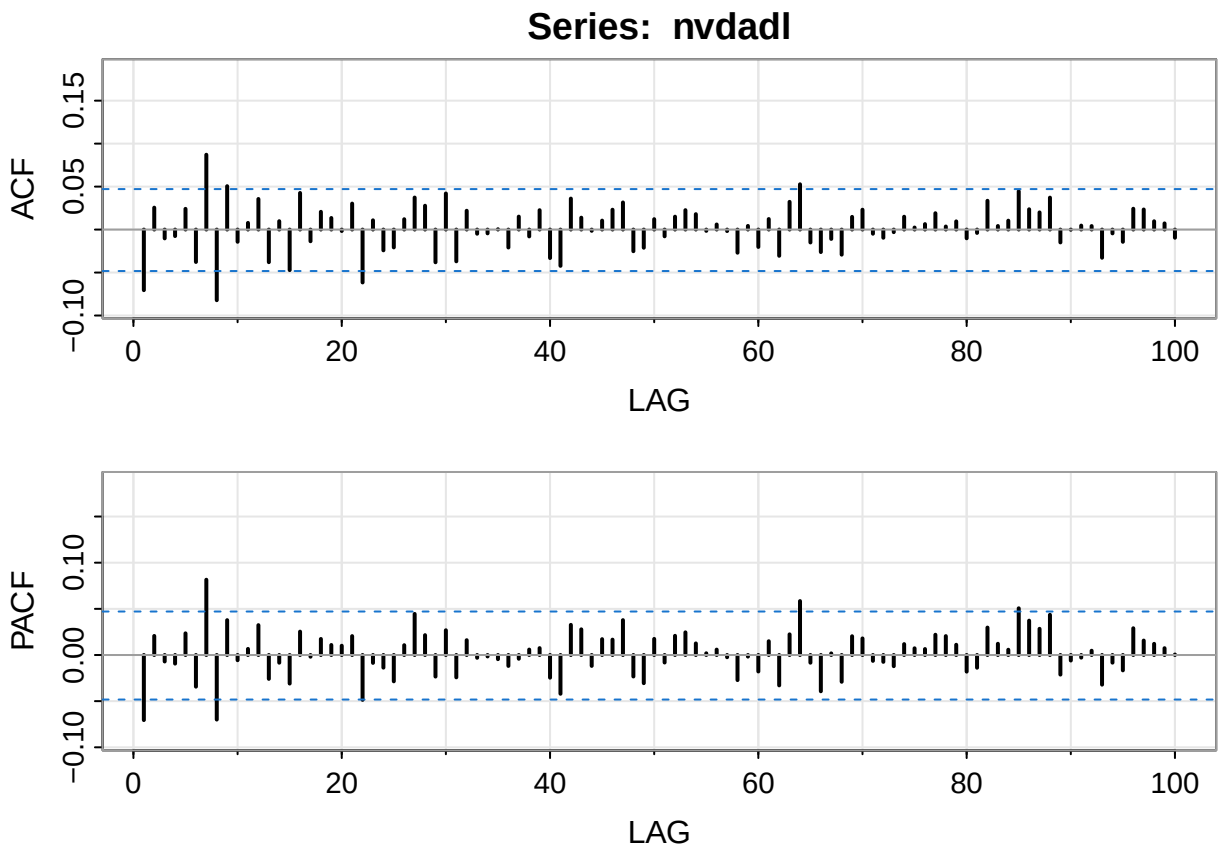



We notice a very slow decay in the ACF, telling us there it is likely there is dependence from the auto-correlation $\hat{\rho}(h)$ between lags.

Therefore we'll perform a log-difference transformation, classically known as the return/growth rate.

Plotting our transformed time series, we obtain:


nvdadl
2017−01−03 / 2023−12−29
0.2
0.1
0.0
−0.1
−0.2
Jan 03 2017
Jul 02 2018
Jan 02 2020
Jul 01 2021
Jan 03 2023




Where we see a much more stationary sequence in the plot. As well as much more stability in our ACF and PACF

**Identification of the difference (d).** Taking the return rate of the original dataset, we identify the difference as 1 since no more is needed, else we risk creating dependence when there should not.

**Estimation of parameters:** Looking at the transformation, we hypothesize two different models from where the ACF and PACF cut off.

The PACF roughly appears to cut off at lag 1 , so our first model will be based off AR(1)

The ACF appears to cut off at 1 as well so our other model will be MA(1)

For AR(1)

```
## <><><><><><><><><><><><><><><>
##
## Coefficients:
##       Estimate     SE t.value p.value
## ar1    -0.0705 0.0238 -2.9635  0.0031
## xmean   0.0017 0.0007  2.4161  0.0158
##
## sigma^2 estimated as 0.0009806595 on 1757 degrees of freedom
##
## AIC = -4.085994  AICc = -4.08599  BIC = -4.076661
##
```

Coefficient estimate for 'ar1' is $-0.0705$ with a standard deviation of $9.8066 \times e^{-4}$

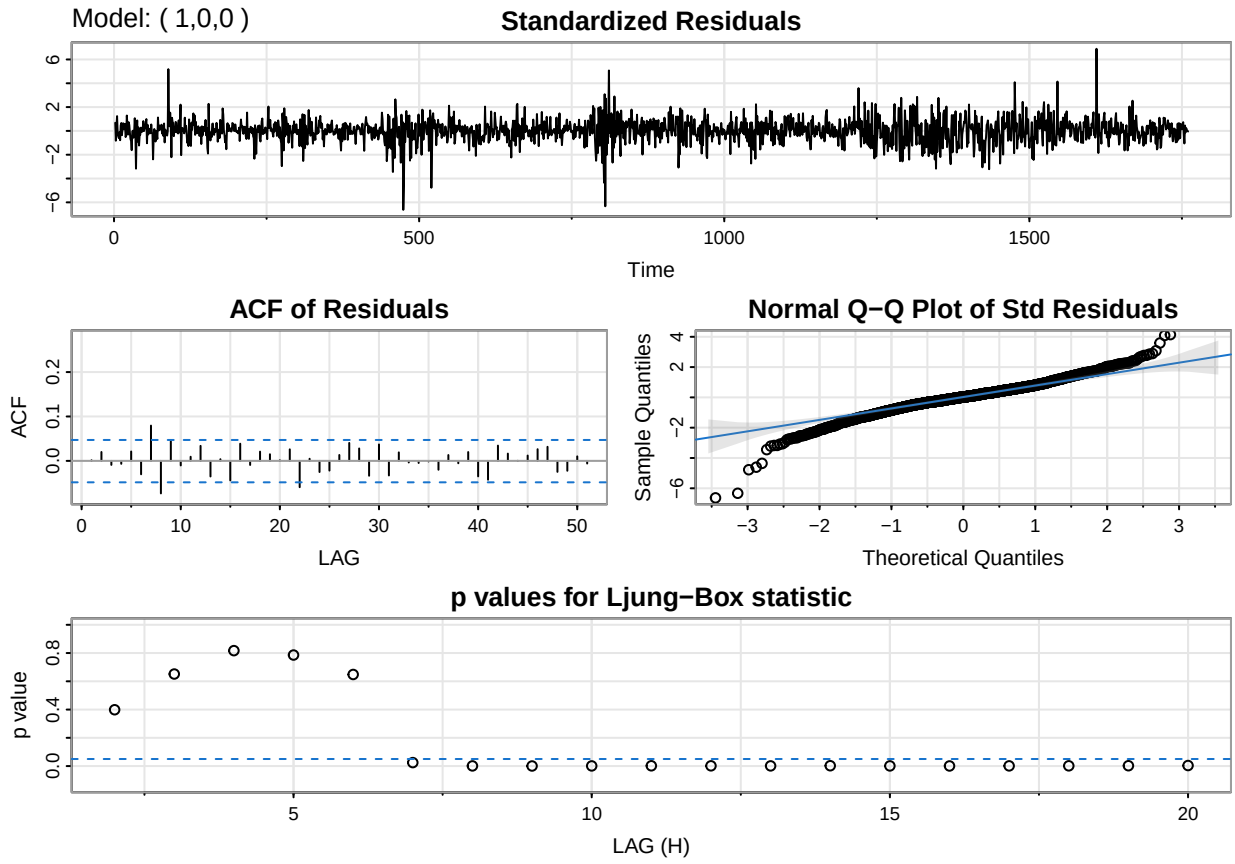
For MA(1):

```
## <><><><><><><><><><><><><><>
##
## Coefficients:
##       Estimate     SE t.value p.value
## ma1    -0.0675 0.0232 -2.9037  0.0037
## xmean   0.0017 0.0007  2.4200  0.0156
##
## sigma^2 estimated as 0.00098087 on 1757 degrees of freedom
##
## AIC = -4.08578  AICc = -4.085776  BIC = -4.076447
##
```

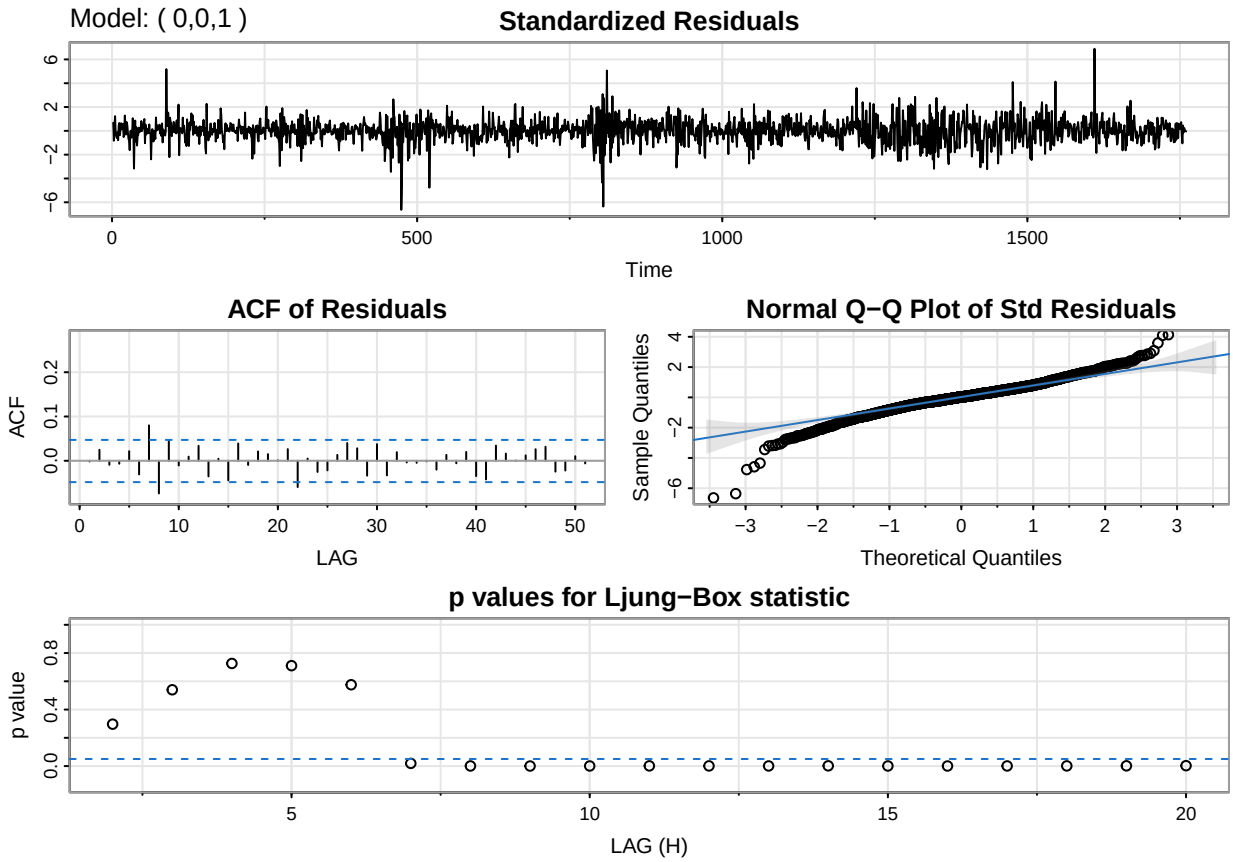
Parameter estimates are $-0.0675$ with standard deviation of $9.8087 \times e^{-4}$

**Diagnostics** Now we will perform a residual analysis to check the adequacy of the models.

```
## initial  value -3.461151
## iter   2 value -3.463641
## iter   2 value -3.463641
## iter   2 value -3.463641
## final  value -3.463641
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##       Estimate     SE t.value p.value
## ar1    -0.0705 0.0238 -2.9635  0.0031
## xmean   0.0017 0.0007  2.4161  0.0158
##
## sigma^2 estimated as 0.0009806595 on 1757 degrees of freedom
##
## AIC = -4.085994  AICc = -4.08599  BIC = -4.076661
##
```




```
## initial  value -3.461151
## iter   2 value -3.463529
## iter   3 value -3.463534
## iter   3 value -3.463534
## iter   3 value -3.463534
## final  value -3.463534
## converged
## <><><><><><><><><><><><><><><>
##
## Coefficients:
##       Estimate     SE t.value p.value
## ma1    -0.0675 0.0232 -2.9037  0.0037
## xmean   0.0017 0.0007  2.4200  0.0156
##
## sigma^2 estimated as 0.00098087 on 1757 degrees of freedom
##
## AIC = -4.08578  AICc = -4.085776  BIC = -4.076447
##
```




Now looking at the diagnostic plots, we can see a similar result for both models.

- The standardized residuals plot does have an average of 0, although we see a few outliers that are at least three standard deviations greater than the average.
- Looking at the ACF of residuals, we notice that both of them are within normal bounds for all lag values displayed up to 30.
- Next, the Q-Q plots of standardized residuals show us the, approximately four, outliers mentioned earlier.
- Lastly, the p-values for Ljung-Box statistic (Q), have high values with the first lag being the smallest just below 0.8 for both models. Then thoroughly maintaining low values for lags displayed up to 20.

**Model Selection:** Criteria such as AIC, BIC for selecting the best model.

Now based off our two models we obtained we will do a comparative analysis to determine which model would be best for predictions.

For our first model we obtained criterions: AIC = -4.085994 AICc = -4.08599 BIC = -4.076661

For the second model, we obtain: AIC = -4.08578 AICc = -4.085776 BIC = -4.076447

Generally, the smallest BIC model will be of smaller order than the smallest AIC model. Luckily for our models, we can see that the second model has both a smaller BIC and smaller AIC. Therefore the choice becomes clear that it is best to move forward with the second model: SARIMA(0,0,1)

### 4.2 GARCH Model Results:

```
## 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")

##
## Attaching package: 'fGarch'

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

**Volatility Clustering: Identification and modeling of volatility clustering.**

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(0, 1) + garch(1, 1), data = nvdadl,
##     cond.dist = "std", trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(0, 1) + garch(1, 1)
## <environment: 0x0000021e8f10ced8>
##  [data = nvdadl]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##          mu          ma1        omega       alpha1        beta1        shape
##  2.5133e-03  -4.0091e-02   3.1208e-05   1.0359e-01   8.7057e-01   5.0326e+00
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## mu      2.513e-03   5.502e-04    4.568 4.91e-06 ***
## ma1    -4.009e-02   2.390e-02   -1.677  0.09348 .
## omega   3.121e-05   1.181e-05    2.642  0.00824 **
## alpha1  1.036e-01   2.303e-02    4.498 6.86e-06 ***
## beta1   8.706e-01   2.862e-02   30.417  < 2e-16 ***
## shape   5.033e+00   6.034e-01    8.340  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  3795.693    normalized:  2.15787
```

```
##
## Description:
##  Tue Jul 30 11:55:24 2024 by user: seren
##
##
## Standardised Residuals Tests:
##                                   Statistic   p-Value
##  Jarque-Bera Test   R    Chi^2  3381.264630 0.0000000
##  Shapiro-Wilk Test  R    W         0.953157 0.0000000
##  Ljung-Box Test     R    Q(10)    11.543135 0.3167951
##  Ljung-Box Test     R    Q(15)    15.416222 0.4218721
##  Ljung-Box Test     R    Q(20)    16.938082 0.6569905
##  Ljung-Box Test     R^2  Q(10)     2.415883 0.9920456
##  Ljung-Box Test     R^2  Q(15)     3.820583 0.9982680
##  Ljung-Box Test     R^2  Q(20)     5.591432 0.9993475
##  LM Arch Test       R    TR^2      3.470464 0.9912127
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.308918 -4.290251 -4.308941 -4.302019
```

Looking at the overall error analysis, we see low p-values indicating potential significance for 'mu', 'beta1', and 'shape' from our GARCH model.

We are also provided the Standardised Residuals Tests, where we see the Shapiro-Wilks test having a very low level of significane, with the inclusion of the Jarque-Bera test, we suspect that our models does not hold to normality.

Additionally, we see the various information criterion displayed above holds higher values all-around compared to our initial SARIMA model.

**Parameter Estimates: Detailed output of GARCH model parameters.** Additionally from the summary of the model we can identify the estimated parameters.

mu (=3.879e-03 ): This is the mean of the log returns. It represents the average return of the Nvidia stock.

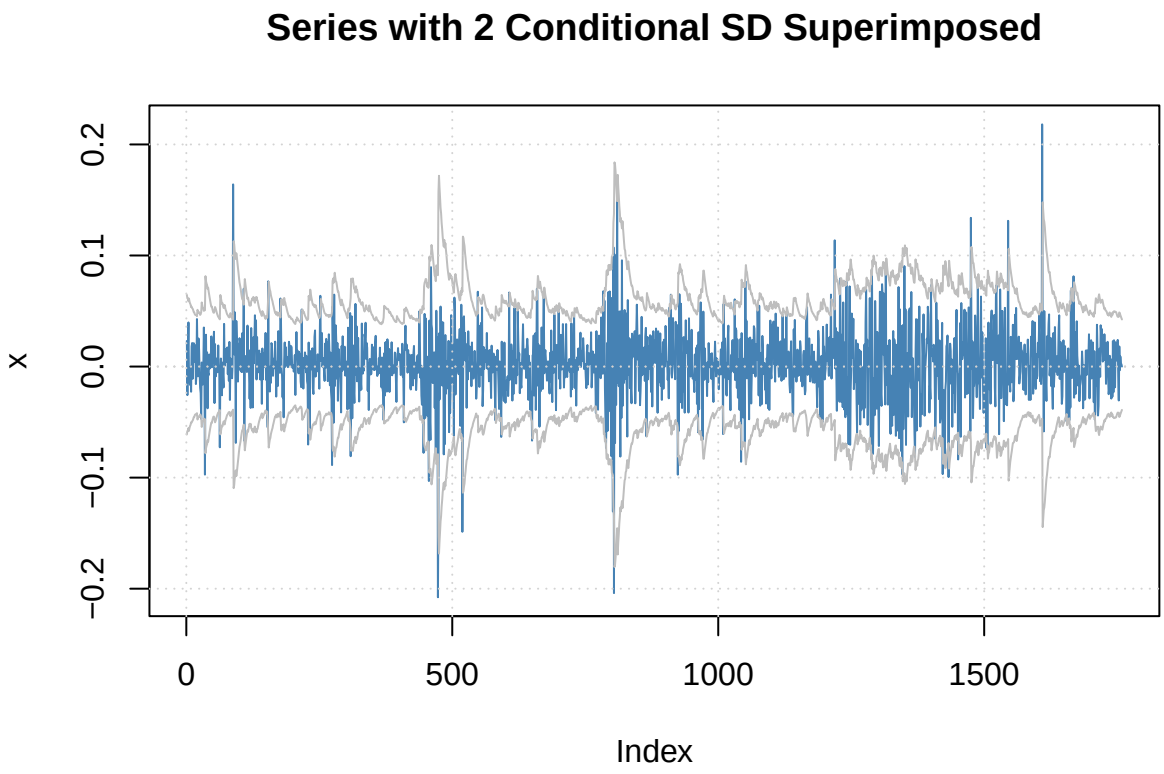
omega (=6.761e-05): This is the constant term in the variance equation. It represents the long-term average variance.

alpha1 (=2.791e-02): This is the coefficient of the lagged squared residuals (ARCH term). It measures the impact of past shocks on current volatility.

beta1 (=8.871e-01): This is the coefficient of the lagged conditional variances (GARCH term). It measures the persistence of volatility.

shape (=5.058): This parameter is specific to the conditional distribution used (in this case, the Student's t-distribution) and indicates the degrees of freedom, affecting the tails of the distribution.

**Forecasting: Future volatility forecast based on the fitted model.**




This plot shows the time series data along with the conditional standard deviations (volatility). The two standard deviations superimposed help visualize the model's ability to forecast volatility. We see approximately four large-scale variations, roughly every 500 indices.

## 5. Conclusion and Future Study:

This study provided a comprehensive analysis of Nvidia's stock prices using time series models. The SARIMA model effectively captured trends, while the GARCH model offered insights into the volatility. Although we had some difficulties with the GARCH model, we got it working although unfortunately decided that is would not be as good of a predictive model than the SARIMA we conducted, as per Information Criteria testing.

Ideally it would be very beneficial and interesting to conduct a portfolio analysis of multiple stocks within the technology sector, modeling a larger equity of the market available.

Future work could explore machine learning methods for further improvement in forecasting accuracy and model selection. Additionally, incorporating macroeconomic indicators could provide a more holistic view of stock price movements. Creating a time series of the GNP alongside to compare and measure squared coherency can possibly assist us in understanding other contributing factors outside of our dataset.

## References:

Books, Journal Papers, Web pages used for the study.

"Time Series Analysis and Its Applications" by Shumway and Stoffer.

## Appendix:

R code used for the analysis. Additional graphs and tables. Mathematical formulas, if any.

```r
# Load the necessary libraries
library(quantmod)
library(xts)
library(naniar)
library(dplyr)
library(tidyverse)
library(graphics)
library(astsa)
library(forecast)
conflicted::conflict_prefer('lag', 'stats')

# Fetch the stock market data
data <- getSymbols("NVDA", src="yahoo", from="2023-01-01", to="2024-05-01", auto.assign=FALSE)

# Check the structure of the data
head(data)

# Convert the data to an xts object and extract the closing prices
nvda_xts <- data[, "NVDA.Close"]

# Display the xts object
nvda <- nvda_xts[,'NVDA.Close']
nvda


# data <- getSymbols("NVDA", src="yahoo", from="2023-01-01", to="2024-05-01", auto.assign=FALSE)
#
# # Let's store our data into a data frame for appropriate usage
# nvidia = data.frame(date = index(data), data, row.names=NULL)
# nvidia
#
# nvda <- ts(nvidia$NVDA.Close,
#            start = c(2023,3),
#            frequency = 252)
# nvda


## BOX-JENKINS APPROACH

##### Step 1. Plot the time series data;

plot(nvda,
     #axes = False,
     ylab = 'Closing Price',
```

```
     main = 'NVIDIA Stock')

acf2(nvda, 50, col = 2:5, lwd = 4, gg = TRUE)



#### Step 2. Transform (log, Box-Cox power transform)

# Take log difference to transform into growth rate or return
#nvdadl <- diff(log(nvidia$NVDA.Close))
nvdadl <- diff(log(nvda))

#plot(x = nvidia$date[-1], y = nvdadl)
plot(nvdadl)
acf2(nvdadl, 100, lwd = 2)



#### Step 4. Estimation of parameters

# AR2
sarima(nvdadl, 2, 0, 0)

# MA2
sarima(nvdadl, 0, 0, 2)


#### GARCH Modeling

library(fGarch)

nvdadl <- diff(log(nvda))[-1]

# Fit a GARCH(1,1) model
garch_model <- garchFit(~ arma(0,1) + garch(1, 1),
                        data = nvdadl,
                        trace = FALSE,
                        cond.dist = 'std')

# Display the summary of the model
summary(garch_model)

# Plot the results
plot(garch_model, which = 3)


#### Forecasting

nvda_ols <- ar.ols(nvda_xts, order = 2, demean = FALSE, intercept = TRUE)
fore <- predict(nvda_ols, n.ahead = 12)

#ts.plot(rec, fore$pred, col=1:2, xlim=c(1980,1990), ylab="Recruitment")
#ts.plot(nvidia$NVDA.Close, fore$pred, col=1:2)#, col = 1:2)#, xlim = c(333,344), ylim = c(0,1))
```

```r
plot(nvda_xts, fore$pred, col=1:2)#, xlim = c(2024.1,2024.37), ylim = c(450,950))
#plot(fore, fore$pred)
U = fore$pred+fore$se; L = fore$pred-fore$se
xx = c(time(U), rev(time(U))); yy = c(L, rev(U))
polygon(xx, yy, border = 8, col = gray(.6, alpha = .2))
lines(fore$pred, type="p", col=2)
```