Modeling and Forecasting Volatility in Financial Time Series:

A Comparative Study of GARCH Models

Brianna Cirillo, Brendan Kenny, Kimberley Maldonado, Emily Su

Final Project for MA 641: Time Series Analysis I

August 18, 2024

Introduction

The ability to accurately model and forecast financial time series data is crucial for economic planning, policymaking, and business strategy. Retail sales and stock price data provide valuable insights into consumer behavior and serve as indicators of a society's economic health. However, the inherent volatility and seasonal nature of the financial sector poses challenges to these objectives.

In this paper we analyze two datasets. One is a seasonal dataset taken from the Federal Reserve Economic Data (FRED) that tracks retail sales. The other dataset is the Apple stock closing price taken from Yahoo Finance. The datasets were chosen based on shared commonalities in finance, which indicate the potential for a cohesive exploration of this kind of data. Retail sales are a useful barometer of consumer behaviors, as certain products exhibit consistent sales throughout the year, while others experience fluctuations based on product popularity and seasonal holidays. Apple stocks were of particular interest as that is one of the most prominent technology companies, making it intriguing to assess whether its popularity remains as strong today as it was in the past. Given these characteristics, models like ARIMA alone may struggle to fully capture the complex dynamics present in financial data. To address this, we explore the applicability of GARCH (Generalized Autoregressive Conditional Heteroskedasticity) models, which are specifically designed to model and forecast time-varying volatility.

With these datasets, we aim to find a model to explain the datasets' behavior, and then test our suggested models to identify the best one, thereby developing a robust model for future forecasting. We begin by conducting a thorough analysis of the datasets' stationarity using Autocorrelation Functions (ACF) and Partial Autocorrelation Functions (PACF), applying Box-Jenkins methodology to identify suitable ARIMA models from which to extract residuals to be fitted to a GARCH model. We compare different ARMA-GARCH model configurations and evaluate the performance of these various models in capturing the patterns exhibited by the data through residual analysis. The best-performing model is chosen based on criteria such as AIC, log likelihood, BIC and the significance of Ljung-Box p-values.

FRED

Data

The following dataset is part of the *Advance Monthly Retail Trade Survery* conducted by the U.S Census Bureau, retrieved from FRED, which measures the retail trade activity in the United States. It provides an early estimate of monthly sales for retail and food service companies capturing consumer demand for goods and services. The RSXFSN data series measures the dollar value of sales and is reported on millions of dollars. This data is not seasonally adjusted and is released monthly. The period of observation for this assessment spans a 32 year period from January 1992 to July 2024, totaling 391 observations.

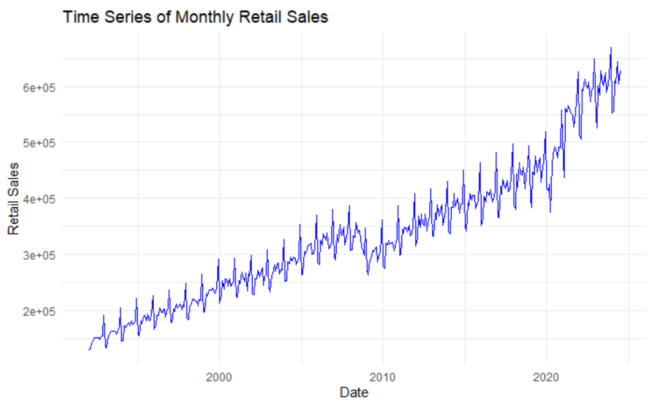


Fig. 1. Time Series of Monthly Retail Sales

Fig. 1 shows an upward trend in retail sales with periodic spikes in the final months of the year, indicating potential seasonality. Notably, there are two distinct periods of elevated volatility corresponding to the years 2008 and 2020. These periods are significant as they align with the 2008 financial crisis and the COVID-19 pandemic, both of which led to substantial disruptions in retail sales activity. To determine whether or not the data is stationary, we analyzed the ACF and PACF graphs and conducted an ADF test.

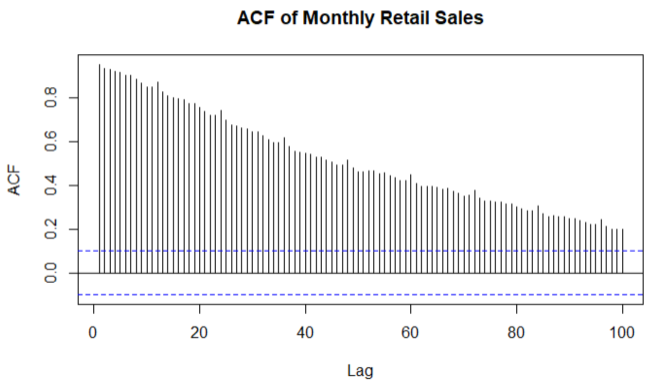


Fig. 2. ACF of Monthly Retail Sales

In the ACF graph, we observe a gradual decay over many lags, suggesting non-stationarity in the data. Additionally, there are several instances of marginally higher spikes in the correlation of retail sales, which align with periods of increased sales activity. In the PACF, we notice a sharp drop-off after the first lag, followed by gradually decreasing periodic significant lags. This pattern suggests that ARIMA and GARCH models could be potentially viable for modeling the data. The ADF test produces a p-value of $0.9752 > 0.05$, thus we reject the null hypothesis of a unit root, which serves as further confirmation of non-stationarity in the data.

Box-Jenkins models

To assess which differencing measures are necessary and appropriate to attain stationarity in the data, a seasonal de-

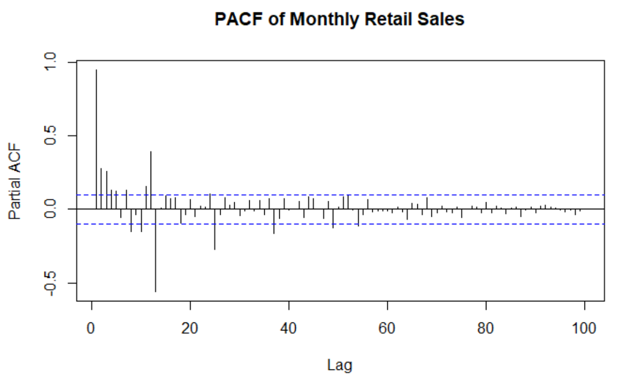


Fig. 3. PACF of Monthly Retail Sales

TABLE I
AUGMENTED DICKEY-FULLER TEST

Test Statistic	Value	Lag Order	p-value
Dickey-Fuller	-0.64059	7	0.9752

composition of the time series into trend, seasonal and residual components was performed.

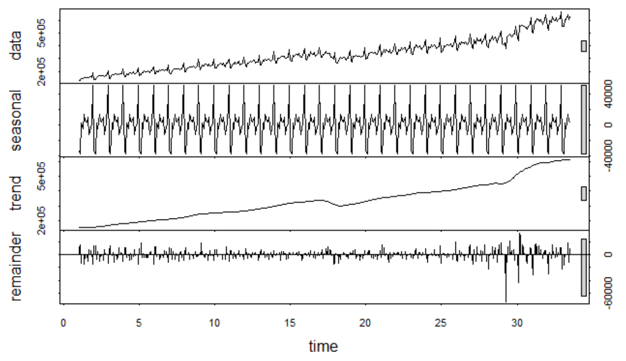


Fig. 4. Seasonal Decomposition of FRED Time Series

The second panel isolates the seasonal component and shows a repeating pattern over each cycle. Given the context of the monthly data, it likely reflects a yearly cycle. The amplitude and shape of the seasonal pattern are consistent, indicating a strong, stable seasonal effect in the data. For this reason, both seasonal differencing and first differencing were applied to the data in order to achieve stationarity.

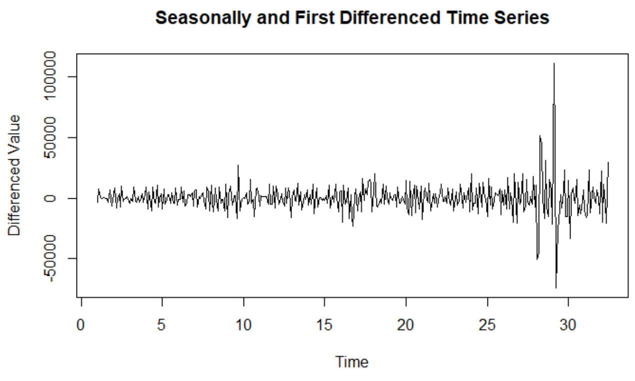


Fig. 5. Combined Seasonally and First Differenced Time Series

The combined seasonally and first differenced data now resembles white noise with no clear trend or seasonality. However, high levels of volatility remain present, particularly in later lags. To confirm stationarity, we perform a second ADF Test as well as a KPSS Test for Level Stationarity.

TABLE II
AUGMENTED DICKEY-FULLER TEST

Test Statistic	Value	Lag Order	p-value
Dickey-Fuller	-8.3704	7	0.01

TABLE III
KPSS TEST FOR LEVEL STATIONARITY

Test Statistic	Value	Truncation lag parameter	p-value
KPSS Level	0.015243	5	0.1

The ADF test confirms that the p-value of 0.01 is less than 0.05, thus we conclude that the series is stationary. In the KPSS, the p-value of 0.1 is greater than 0.05, thus we do not reject the null hypothesis, and the data can be considered stationary. Since stationarity has been confirmed, we proceed by analyzing the ACF, PACF, and EACF of the differenced time series to determine the appropriate orders for ARIMA modeling. Additionally, the volatility observed in the data, which is common in financial time series, is addressed by fitting the residuals of the best-performing ARIMA model with a GARCH model. This approach is particularly suitable for capturing volatility clustering.

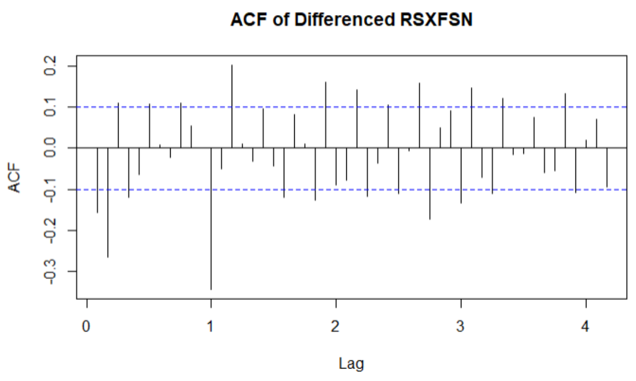


Fig. 6. ACF of Differenced Time Series

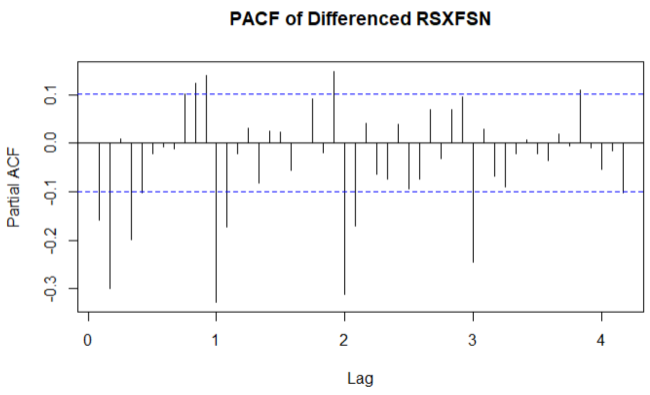
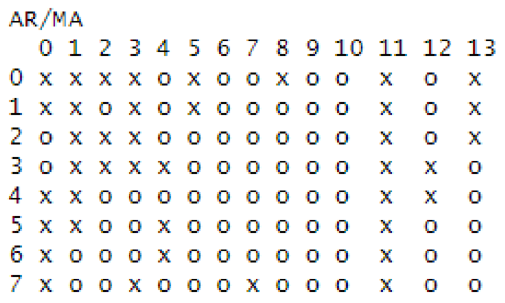


Fig. 7. PACF of Differenced Time Series



```
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x x x x o x o o x o o  x  o  x
1 x x o x o x o o o o o  x  o  x
2 o x x x o o o o o o o  x  o  x
3 o x x x x o o o o o o  x  x  o
4 x x o o o o o o o o o  x  x  o
5 x x o o x o o o o o o  x  o  o
6 x o o o x o o o o o o  x  o  o
7 x o o x o o o x o o o  x  o  o
```

Fig. 8. EACF of Differenced Time Series

Based on the EACF in Fig. 8, several ARMA orders seem viable. To be considered are ARMA(2,2), ARMA(3,3) and ARMA(2,4). The PACF in Fig. 7 suggests as many as 3 significant seasonal MA lags, which will also need to be considered. We assess several combinations of ARMA orders p and q based on AIC and BIC criteria and seasonality order $s = 12$.

TABLE IV
ARIMA MODEL COMPARISON

Model	AIC	Log Likelihood	σ^2
$\text{ARIMA}(1,0,2) \times (0,0,1)_{12}$	8031.19	-4009.8	94415758
$\text{ARIMA}(2,0,2) \times (0,0,1_{12}$	8032.99	-4009.49	94762612
$\text{ARIMA}(2,0,4) \times (0,0,1)_{12}$	7990.07	-3986.04	82779648
$\text{ARIMA}(2,0,4) \times (0,0,3)_{12}$	7986.46	-3982.23	81198384
$\text{ARIMA}(3,0,3) \times (0,0,1)_{12}$	7992.86	-3987.43	83325181
$\text{ARIMA}(3,0,3) \times (0,0,3)_{12}$	7988.21	-3983.11	81455728

TABLE V
LJUNG-BOX TEST P-VALUES AT VARIOUS LAGS
$\text{ARIMA}(2,0,4) \times (0,0,3)_{12}$

Lag	p-value
1	0.5720
6	0.2414
12	0.0046
18	0.0169
24	0.0212
30	0.0252

Table IV shows that based on criteria listed, $\text{ARIMA}(2,0,4) \times (0,0,3)_{12}$ is the most promising model in terms of AIC. However, Table V shows that after this model's residuals were fitted to a GARCH implementation, the Ljung-Box Test p-values were well below the 0.05 significance level after several lags. In an effort to ensure that the residuals are uncorrelated and generally well-behaved for better future forecasting, the decision was made to prioritize the model with significant Ljung-Box p-values, even if its AIC is marginally higher. Thus, we proceed with the $\text{ARIMA}(3,0,3) \times (0,0,3)_{12}$, which has the next lowest AIC.

ARIMA$(3,0,3) \times (0,0,3)_{12}$ Model

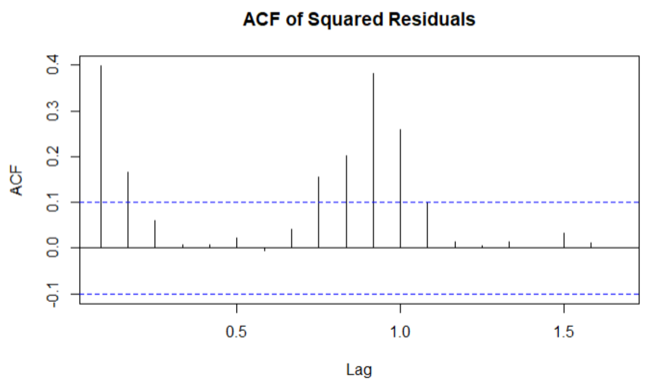


Fig. 9. ACF of Squared Residuals

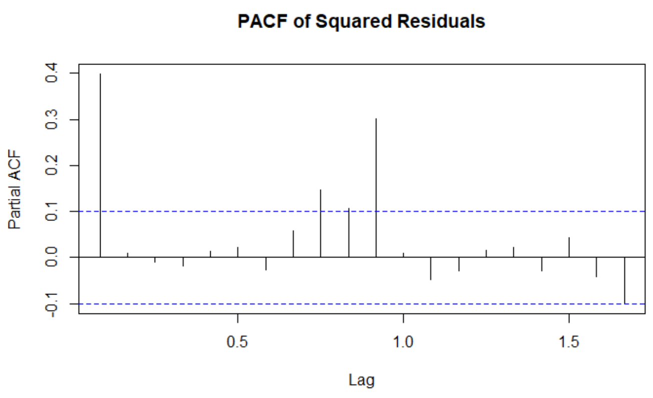


Fig. 10. PACF of Squared Residuals

In order to ascertain the appropriate GARCH order, the residuals of the ARIMA model's residuals were squared. The correct GARCH order is given by the significant lags in the ACF and PACF of the squared residuals, noted in Figs. 9 and 10, which indicate a GARCH(1,1) model. Next, we compare different ARMA orders to ascertain which produces the best AIC criteria. Considering the inherent volatility in the data, we also consider comparing the results from using the normal distribution versus the student t distribution, as the latter might provide more flexibility in modeling the heavy tails often seen in financial time series.

TABLE VI
GARCH MODEL COMPARISON: "NORM" DISTRIBUTION

Model	AIC	BIC
ARMA(1,0)-GARCH(1,1)	20.5898	20.63143
ARMA(1,1)-GARCH(1,1)	20.61563	20.66768
ARMA(2,0)-GARCH(1,1)	20.5979	20.64995
ARMA(2,1)-GARCH(1,1)	20.59256	20.65502
ARMA(2,2)-GARCH(1,1)	20.55244	20.6253

TABLE VII
GARCH MODEL COMPARISON: "STUDENT T" DISTRIBUTION

Model	AIC	BIC
ARMA(1,0)-GARCH(1,1)	20.49068	20.54273
ARMA(1,1)-GARCH(1,1)	20.49591	20.55837
ARMA(2,0)-GARCH(1,1)	20.49921	20.56167
ARMA(2,1)-GARCH(1,1)	20.48521	20.55808
ARMA(2,2)-GARCH(1,1)	20.4684	20.55168

Tables VI and VII show that out of either distribution, the ARMA(2,2)-GARCH(1,1) model evaluated with the student t distribution is the best performing, based on the AIC metric result of 20.4684.

According to Figs. 11, 12, and 13, the ARMA(2,2)-GARCH(1,1) model appears to be a good fit for the data. All lags except for lag 12 in the ACF of Standardized Residuals are within the bounds of the confidence interval, suggesting that the residuals resemble white noise. The correlation at lag 12 could be a reflection of the inherent volatility in the data, as well as the seasonal order considered in our modeling. The goodness of fit is further supported by the QQ plot of the standardized residuals, which shows that the points generally align along the diagonal line, with some deviation in the tails. The histogram of the standardized residuals also indicates that they resemble the bell-shaped curve of a normal distribution. Next, we turn to the Ljung-Box Test for further insight into the stability of the residuals' stability.

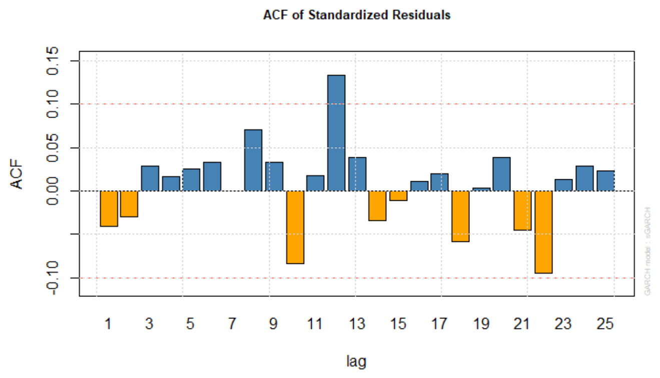


Fig. 11. ACF of Standardized Residuals

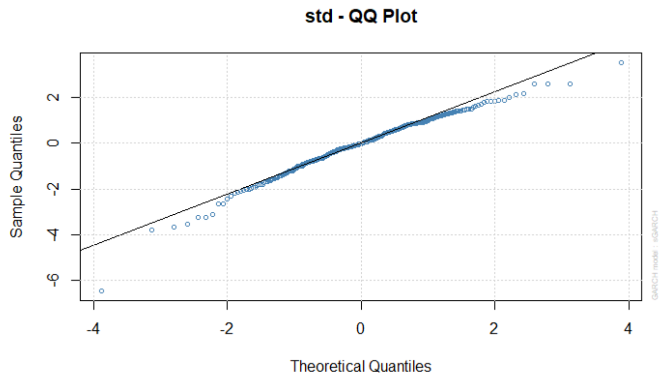


Fig. 12. QQ Plot of Standardized Residuals

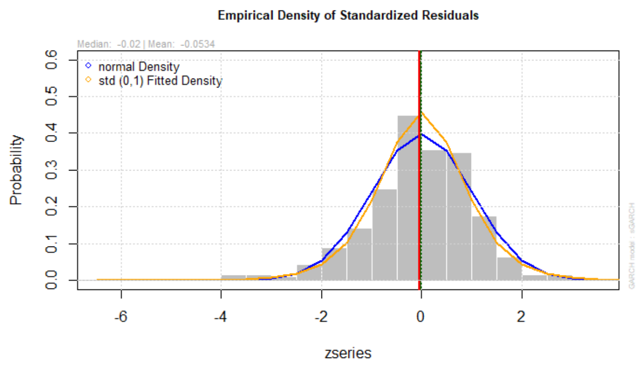


Fig. 13. Histogram of Standardized Residuals

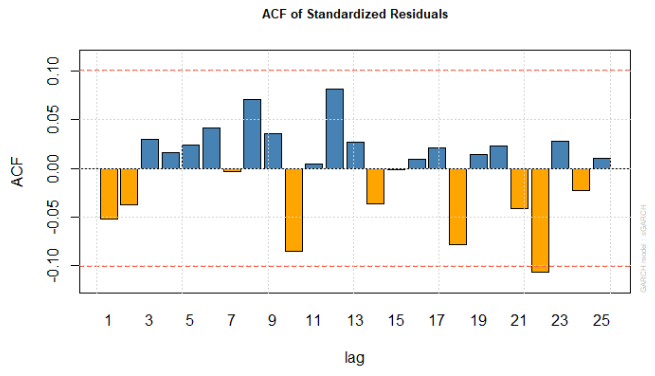


Fig. 15. ACF of Standardized Residuals

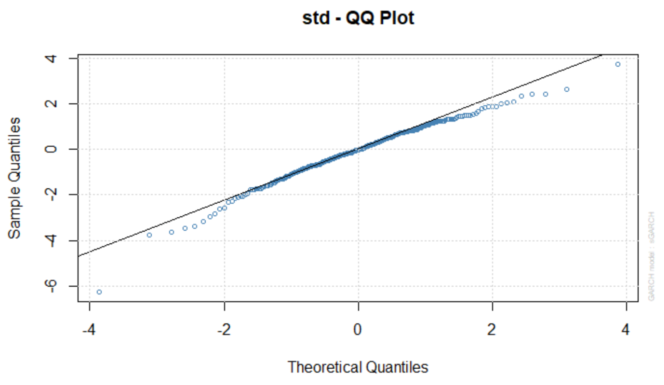


Fig. 16. QQ Plot of Standardized Residuals

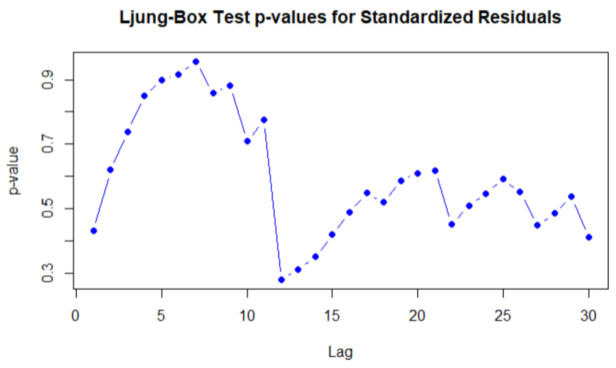


Fig. 14. Ljung-Box Test p-values for ARMA(2,2)-GARCH(1,1)

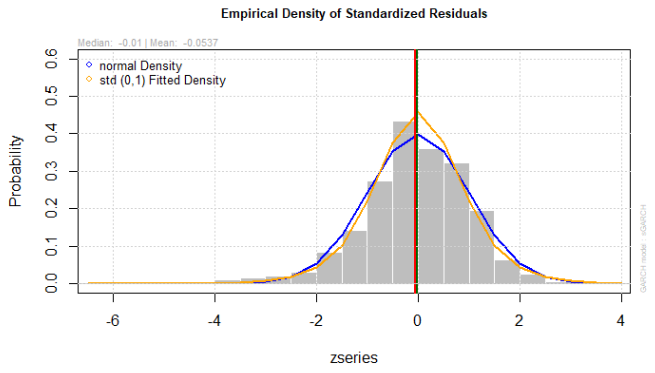


Fig. 17. Histogram of Standardized Residuals

According to Fig. 14, the p-values all appear to be above the significance level of 0.05. However, while this graph shows high p-values initially, there is a sharp drop after lag 10, with some lower p-values in the middle range. This drop and the subsequent lower p-values suggest possible autocorrelation at these lags. For this reason, we refer back to Table IV to consider the ARIMA$(3,0,3) \times (0,0,1)_{12}$ model to assess whether changing the order of the seasonal MA component resolves the issue of instability in the residuals.

ARIMA$(3,0,3) \times (0,0,1)_{12}$

After following the same process described in the previous section, i.e squaring the residuals, comparing ARMA/GARCH orders, and comparing normal versus student t distributions, the ARMA(2,2)-GARCH(1,1) model with a student t distribution remained the best performing model according to AIC.

TABLE VIII
GARCH MODEL FOR ARIMA$(3,0,3) \times (0,0,1)_{12}$: "STUDENT T" DISTRIBUTION

Model	AIC	BIC
ARMA(2,2)-GARCH(1,1)	20.47366	20.55693

Figs. 16 and 17 show that the QQ plot and histogram for the GARCH model fitted to the ARIMA$(3,0,3) \times (0,0,1)_{12}$ residuals are quite similar to those fitted to the ARIMA$(3,0,3) \times$

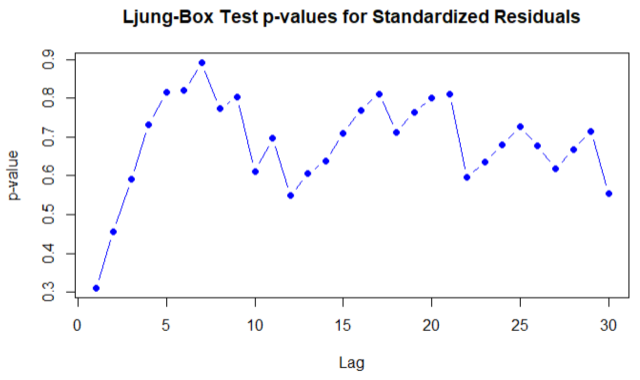


Fig. 18. Ljung-Box Test p-values for ARMA(2,2)-GARCH(1,1)

$(0,0,3)_{12}$ model, and show that the residuals are reasonably normally distributed. Fig. 15, the ACF of the Standardized Residuals, improves the correlation at lag 12 seen in the previous model. Here, only lag 22 is slightly out of bounds of the confidence level.

Fig. 18 shows a generally higher set of p-values compared to Fig. 14, with most of them comfortably above 0.5, and no sharp drops, suggesting that the residuals likely resemble white noise across the lags. It has more consistency and a smoother pattern with higher p-values, suggesting that fitting the GARCH model to the ARIMA$(3,0,3)\times(0,0,1)_{12}$ is more stable, despite it being marginally higher in AIC.

Statistical Conclusions

The previous section illustrated that model implementations that perform the best in metrics like AIC are not necessarily those which produce the most significant Ljung-Box p-values, which check whether the residuals of a model are uncorrelated and serve as an indication that the model has adequately captured the underlying time series structure. A possible cause of this could be overfitting, which occurs when a model is too complex relative to the amount of data it is modeling. If a model is overfitted, it might perform well on these criteria but still leave autocorrelation in the residuals because it is fitting the noise rather than the true signal. This leftover autocorrelation would lead to low p-values in the Ljung-Box test, indicating that the residuals are not white noise. In this paper, we argue for prioritizing the goodness of fit of the residuals most resembling white noise.

Considering the consistency, smoothness and level of the p-values, there is an argument to be made for the suitability of the GARCH model fitted to the ARIMA$(3,0,3)\times(0,0,1)_{12}$ model's residuals for forecasting purposes. However, since the Ljung-Box p-values are significant when fitting to either ARIMA$(3,0,3)\times(0,0,1)_{12}$ and ARIMA$(3,0,3)\times(0,0,3)_{12}$ instances, it may fare better to choose the one with the lower AIC. Thus, we proceed with a final assessment of the ARMA(2,2)-GARCH(1,1) model fitted to ARIMA$(3,0,3)\times(0,0,3)_{12}$ residuals with the student t distribution.

TABLE IX
AIC COMPARISON BETWEEN ARMA(2,2)-GARCH(1,1) FITTED TO DIFFERENT RESIDUALS

Model	AIC
ARIMA$(3,0,3)\times(0,0,1)_{12}$	20.47366
ARIMA$(3,0,3)\times(0,0,3)_{12}$	20.4684

According to Table X, in the ARMA(2,2) component, the ar1, ar2, ma1, and ma2 terms are all highly significant with a p-values of 0 or essentially zero, in the case of ma2. In the GARCH(1,1) component, the alpha1 is statistically significant, indicating that the past lagged residuals have a statistically significant impact on the current conditional variance. This shows that recent volatility significantly influences current volatility. The beta1 term is also highly significant (p-value $=$ 0). This suggests that the volatility is persistent over time. The shape parameter is significant, suggesting that the residual's distribution, in this case, the student t distribution, is appropriately modeled.

TABLE X
ARMA(2,2)-GARCH(1,1) MODEL ESTIMATION RESULTS

Parameter	**Estimate**	**Std. Error**	**t value**	**Pr$>|t|$**
ar1	-1.7148	0.0742	-23.1155	0.0000
ar2	-0.9119	0.0692	-13.1842	0.0000
ma1	1.6115	0.1199	13.4405	0.0000
ma2	0.7791	0.1139	6.8383	8.01e-12
omega	79510.8593	259985.5	0.3058	0.7597
alpha1	0.1281	0.0291	4.4070	1.05e-05
beta1	0.8709	0.0289	30.0826	0.0000
shape	6.7516	1.5766	4.2825	1.85e-05

TABLE XI
FORECASTED TIME SERIES VALUES

Time Period	**Forecasted Value**
T+1	11801.44
T+2	11798.90
T+3	11796.37
T+4	11793.84
T+5	11791.32
T+6	11788.79
T+7	11786.27
T+8	11783.75
T+9	11781.23
T+10	11778.72
T+11	11776.20
T+12	11773.69

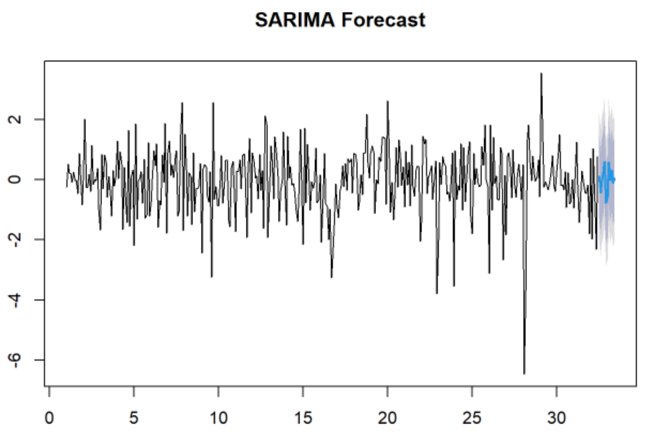


Fig. 19. Forecasted Time Series

The blue line in Fig. 19 extending to the right represents the forecasted values from the model. The model appears to be capturing the general trend and seasonality of the data, given that the forecast does not show large fluctuations and remains within a reasonable range compared to historical data. According to Table XI, the values show a slight downward trend over the 12 periods, consistent with the blue forecast line in the plot.

AAPL

Data

For the non-seasonal data, we investigated the monthly close price of Apple Inc. from January 2016 to May 2024. The dataset included the monthly open, high, low, and close price of Apple stock. We applied various time series techniques, including Autoregressive Integrated Moving Average (ARIMA) and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models, to capture the underlying patterns in the close price. Model selection was based on Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC). Residual analysis of the models ensured the data met the assumptions of the models used. The comprehensive modeling approach allowed for accurate forecasting and created valuable insights about AAPL stock.

To start our analysis we examined the descriptives of the monthly close prices for Apple stock. The mean and median values suggest that the central tendency of the stock prices is around 73 to 96. The range indicates that the stock price has varied significantly from 2016 to 2024. We then looked at the time series plot of the monthly close price.

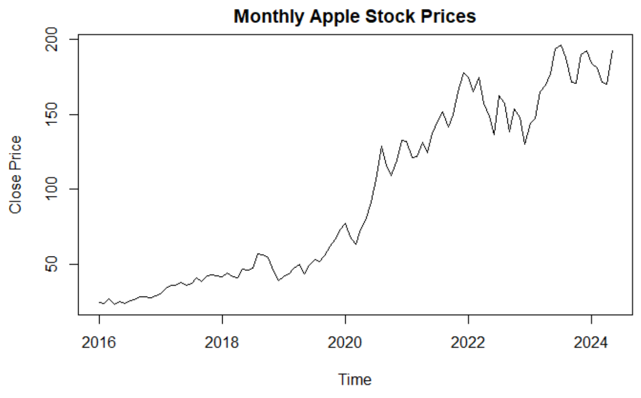


Fig. 20. Monthly Times Series of Apple

There is a clear upward trend in the data over this period of time. There is visible volatility in the data, with the fluctuations becoming increasingly more pronounced in the later years. While Apple is a retail company, there is not any obvious seasonality such as what was seen in the FRED retail sales data set.

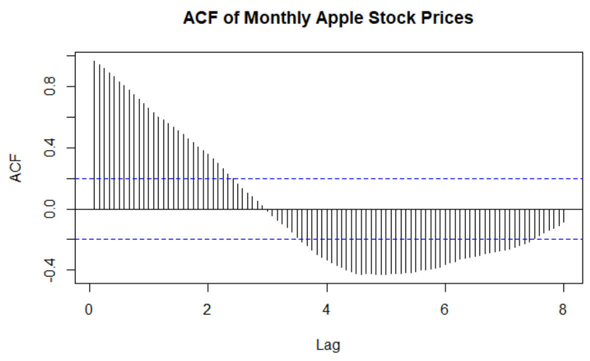


Fig. 21. Monthly ACF

The above plot represents the ACF for the Apple data. The strong gradual decay and an overall sinusoidal shape to the data suggests that the data is non-stationary. The high degree of positive auto correlations in the beginning lags implies momentum in the stock prices, which is common when dealing with financial data.

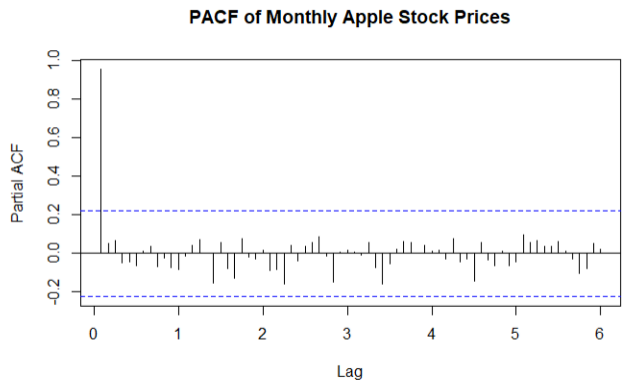


Fig. 22. Monthly PACF

Now, looking at the PACF, we see a sharp drop after the first lag. This is suggests that a potential AR(1) model may be appropriate to capture the dependency in the data. A significant first lag indicates that the immediate past value has a strong influence on the current value, while values further in the past have less so.

TABLE XII
AUGMENTED DICKEY-FULLER TEST

Test Statistic	Value	Lag Order	p-value
Dickey-Fuller	-2.1923	4	0.497

Since the p-value is greater than 0.05, we fail to reject the null hypothesis that the time series has a unit root. This indicates that the series is indeed non-stationary and that differencing is needed. Differencing the data will remove trends and stabilize the mean to allow for more suitable modeling.

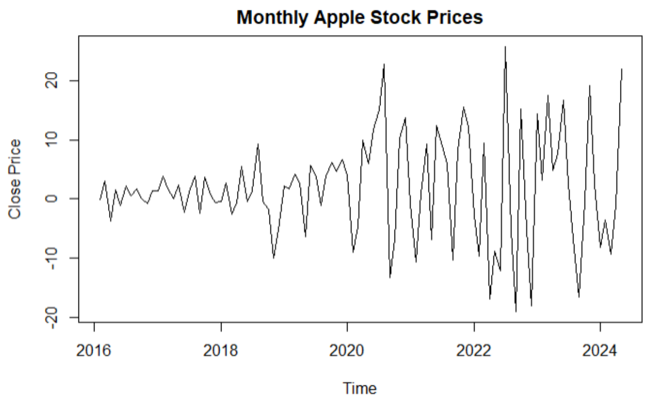


Fig. 23. Differenced Monthly Times Series

After a single round of differencing the above plot shows the Apple stock data. Visually the data looks less dependent on time and therefore closer to a stationary time series. To confirm, we will test it again with the Dickey Fuller test.

TABLE XIII
AUGMENTED DICKEY-FULLER TEST

Test Statistic	Value	Lag Order	p-value
Dickey-Fuller	-4.3039	4	0.01

After applying first-order differencing to the series, the Dicker-Fuller test resulted in a p-value of 0.01. This suggests that the Apple data is now stationary. Therefore, we continue our analysis by looking at the ACF and PACF plots of the differenced data to better understand which models may be suitable.

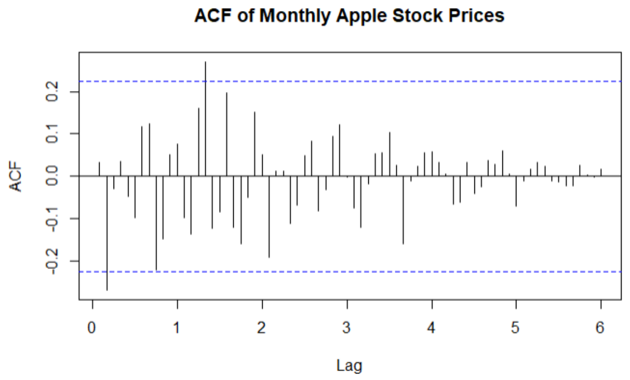


Fig. 24. Monthly Difference ACF

The ACF plot shows exponential decay which may indicate an AR model. There is also a spike at lag 2 and then no significant autocorrelation indicating a MA(2) model may also be suitable. Gradual decay of the autocorrelations suggests a potential AR component, but initial spikes may also indicate an MA component.

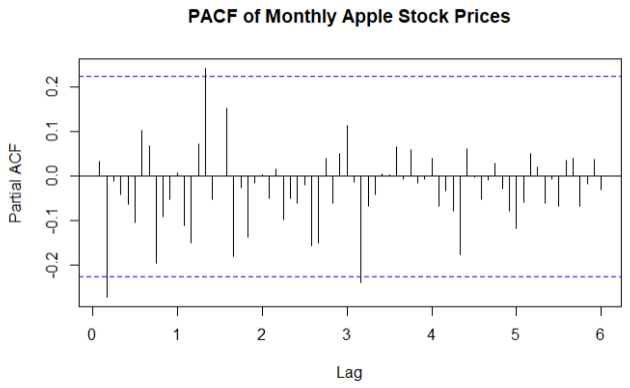


Fig. 25. Monthly Difference PACF

The PACF plot further confirms our findings from the ACF plot, showing a significant spike at lag 2 suggesting an AR(2) component. After the initial spike at lag 2, there seems to be no more significant spikes. This implies there are no strong additional AR terms needed in the model.

Box-Jenkins models

Thus, we continued the analysis by fitting different ARIMA models, to see which best fit the data. We considered AR(2), MA(2), ARIMA(1,1,1), ARIMA(2,1,1), ARIMA(1,1,2), and ARIMA(2,1,2) models.

TABLE XIV
MODEL COMPARISON

Model	AIC	BIC
AR	718.2679	728.6886
MA	718.1726	728.5933
ARIMA(1,1,1)	721.9588	729.7442
ARIMA(2,1,1)	717.1604	727.5408
ARIMA(1,1,2)	721.3812	731.7617
ARIMA(2,1,2)	719.1593	732.1349

Listed above is a table including the AIC and BIC values for a number of different ARIMA models. The AIC, or Akaike Information Criteria, captures the quality of the model to a specific data set with the lowest value indicating a better fitting model. The BIC, or Bayesian Information Criterion, estimates the quality of the model as well but penalizes models that are more complex. Looking at the ARIMA(2,1,1) we have the lowest AIC and BIC value of 717.1604 and 727.5408 respectively. This suggests that the ARIMA(2,1,1) model best captures the dependency in the data compared to the other ARIMA models.

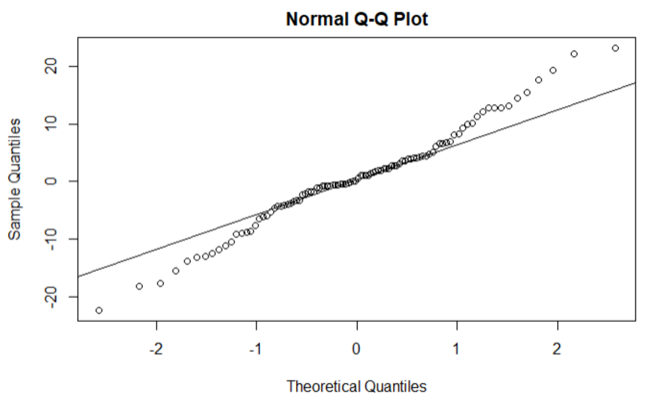


Fig. 26. Residuals QQ Plot

Now we will perform an analysis on the residuals to ensure that the model is well fitting of the data. According to the quantile-quantile plot, we can see some outliers in the upper and lower tails. Generally the data looks normal but we will need to confirm with a Shapiro-Wilk test. With the Shapiro-Wilk Test we measure how close the data is to a normal distribution. With a p-value of 0.2318, greater than 0.05, there is not enough evidence to reject the null hypothesis suggesting that the data is normally distributed. More important than the normality of the data is the independence of the residuals, which can be tested with a Ljung-Box Test. With a p-value of 0.9029, which is greater than 0.05, so we can say there is not enough evidence to reject the null hypothesis suggesting that the residuals are independent. Based on the Shapiro-Wilk and Ljung-Box tests, the residuals from the model appear to be normally distributed and uncorrelated, which suggests that the model might be a good fit for the data.

TABLE XV
MODEL DIAGNOSTICS

Test	Statistic	df	p-value
Shapiro-Wilk	0.98315		0.2318
Box-Ljung	0.014896	1	0.9029

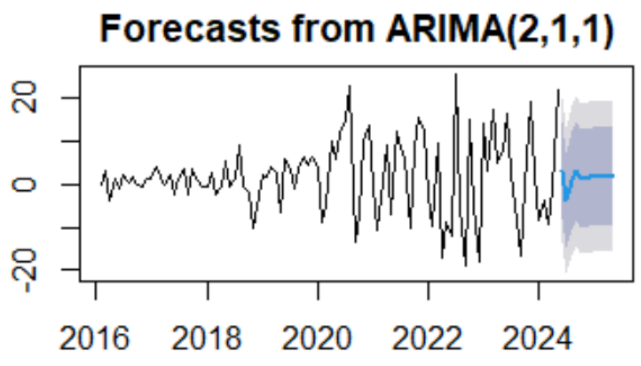


Fig. 27. ARIMA Forecast

Finally, we forecast future value of the Apple data. The confidence intervals widen as we move further into the future, reflecting increasing uncertainty in the predictions as we move further from the current time period.

We then apply a Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model because the Apple data, like most financial time series data, exhibits a lot of volatility. GARCH models are well suited for modeling time-varying volatility and captures the fluctuations well. The aim is to better understand and forecast the monthly returns of Apple stock. First, we calculated the returns by taking the logarithm of the monthly returns and differencing the result to ensure stationarity. Utilizing our findings from the ARIMA models, we created ARFIMA-GARCH(1,1) models which can be seen in the table below.

TABLE XVI
GARCH MODEL COMPARISON

Model	AIC	BIC
ARFIMA(1,0)-GARCH(1,1)	-2.0775	-1.9473
ARFIMA(2,0)-GARCH(1,1)	-2.0822	-1.9259
ARFIMA(0,2)-GARCH(1,1)	-2.0873	-1.9310
ARFIMA(2,1)-GARCH(1,1)	-2.0705	-1.8882
ARFIMA(2,2)-GARCH(1,1)	-2.0843	-1.8759

We compare the models based on their AIC and BIC values, where the AIC is favorable to more complex models while the BIC has a harsher penalty for complexity of the model. From the table above, we can see that the ARFIMA(0,2)-GARCH(1,1) has the lowest AIC value. This means it overall balances model complexity and goodness of fit while still best fitting the data. The BIC value showed this model was favored as strongly, but still was competitive in comparison to the other other models (-1.9310). Thus, the ARFIMA(0,2)-GARCH(1,1) was the most appropriate model for Apple's monthly stock return data. To ensure the assumptions of the model are met, we consider some diagnostic plots.

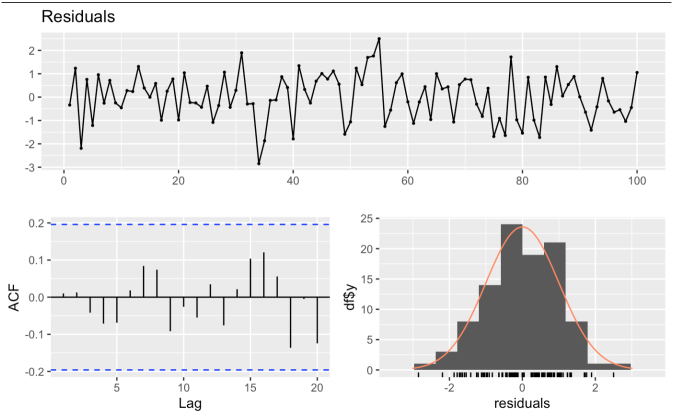


Fig. 28. Standardized Residuals GARCH Model

The above diagnostic plots of the ARFIMA(0,2)-GARCH(1,1) model's standardized residuals are used to evaluate the adequacy of the model in capturing the underlying dynamics of the data. The time series plot of standardized residuals shows that the residuals fluctuate randomly around zero, which indicates the model has successfully captured the main structure of the data. The ACF supports this, as the lags fall within the confidence intervals¿ This means that there is minimal autocorrelation in the residuals and they are independent. Additionally, the histogram of residuals seem to be approximately normally distributed, which aligns with the model's assumptions. We then created a QQ plot of the quantiles of the standardized residuals plotted against the standard normal quantiles.

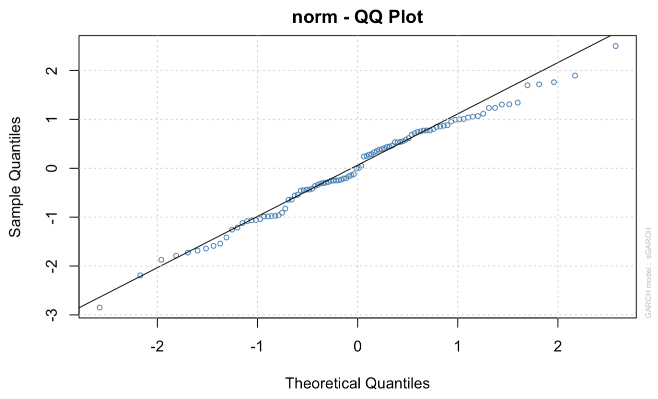


Fig. 29. Norm QQ Plot

The standardized residuals closely follow the theoretical standard normal quantiles. There are some deviations in the tails which suggests that there may be some departures from normality, but overall the fit is strong. This indicates that the residuals are approximately normal and the selected GARCH model adequately captures the volatility in the data. To check for remaining autocorrelation in the volatility of the returns, we looked at the ACF plot of the squared standardized residuals.

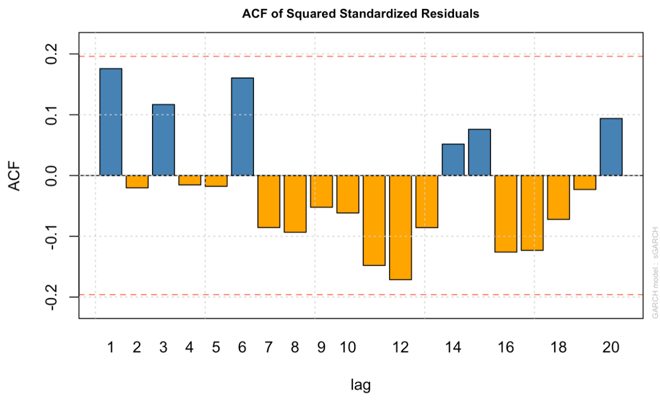


Fig. 30. ACF Squared Residuals for Garch

From this plot, we see that all of the autocorrelations are contained within the 95% confidence interval. Therefore indicating that the residuals act as white noise and that the model has successfully captured the volatility in the data.

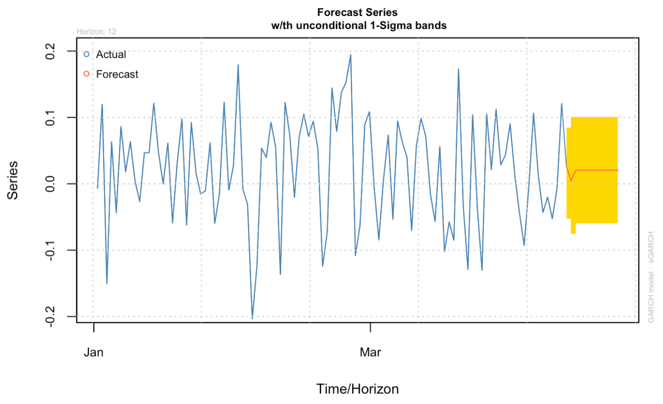


Fig. 31. Apple Forecasted GARCH Model

Since we have identified a GARCH model that appropriately fits the data and has satisfied all necessary assumptions, we proceed with forecasting. The plot shows the original time series with the forecasted values in confidence bands of one standard deviation. The model's forecast appears stable and within the expected range. we can see the forecasts capture the central tendency of the time series, which can occur with financial data.

Statistical Conclusions

Through the process of model selection, we determined the best ARIMA model to be ARIMA(2,1,1) and the best GARCH model to be ARFIMA(0,2)-GARCH(1,1). Both demonstrated being a relatively good fit, however there is a clear winner of the two.

TABLE XVII
ARIMA vs GARCH Model Comparison

Model	AIC	BIC
ARIMA(2,1,1)	717.1604	727.5408
ARFIMA(0,2)-GARCH(1,1)	-2.0873	-1.9310

From the above table, we can see that the ARFIMA(0,2)-GARCH(1,1) model significantly outperforms that of the ARIMA model, both AIC and BIC values are lower. The GARCH model's superiority lies in its ability to model the volatility clustering that can be seen in financial time series data. It does this by explicitly modeling the heteroskedasticity in the data, which leads to more accurate predictions. The ARIMA model, on the other hand, captures the linear components and handles autocorrelations in the data, but fails to account for the variance changing over time. Therefore the ARFIMA(0,2)-GARCH(1,1) is superior in modeling Apple's monthly stock data because it captures underlying volatility and gives a more comprehensive and accurate forecast of the data.

Conclusions

In the seasonal data, the ARMA(2,2)-GARCH(1,1) model, when fitted to the residuals of the ARIMA$(3,0,3)\times(0,0,3)_{12}$ model with a student t distribution, was chosen as the most appropriate model with which to conduct forecasting analysis.

The criteria that led to this choice was that it displayed the best balance between AIC score and significant Ljung-Box p-values. The effectiveness of the GARCH model in capturing volatility clustering in the retail sales data, which is a common characteristic in financial time series, is evidenced by the statistical significance of the estimated ARMA, GARCH and shape parameters.

The implications of this study could be valuable to policymakers or business stakeholders seeking to make the most well-informed decisions possible in the presence of economic shocks or seasonal patters. In particular, studying the volatility exhibited in retail sales helps in understanding the impact of major economic events, such as the 2008 financial crisis and COVID-19 pandemic, on consumer behavior.

In the future, other alternative models could be considered in conjunction with the traditional ARIMA and GARCH methods, which are well-suited for data that exhibit linear relationships, trends, and volatility clustering, all of which were present in the seasonal dataset. However, they might struggle with interactions that are not easily captured by these models. In this sense, ANN and DNN models can be powerful tools for capturing non-linear pattern in time series and improve the predictive accuracy of the behavior of the financial data over time.

In the non-seasonal data, the ARFIMA(0,2)-GARCH(1,1) model resulted in the best fit model. Similar to the seasonal data, the model resulted in the lowest AIC and BIC scores. It also provided residuals that were independent, which was observed through the Box-Ljung test.

Some real world implications to the application of this time series would be predictive modeling of future stock prices. This is important as the financial services sector provides commercial and retail customers guidance on their investments. Understanding key features to Apple stock price, such as volatility and correlation with current and future events, is key to determining the portion of a portfolio or fund that Apple should represent. In doing so companies can better provide upside in stock appreciation while simultaneously understanding and mitigating the potential downside from risk.

A future direction, when dealing with Apple stock data, could be doing a comparison Apple to other companies or economic data. To do so we could leverage a VAR, or vector auto regression, and capture the dependency in Apple to there other economic data sets. In doing so we could better predict the price of Apple stock and compare the values based on what is predicted through the GARCH model.

References

[Pha13] Ly Pham. *Time Series Analysis with ARIMA–ARCH/GARCH model in R*. Tech. rep. © 2013 L-Stern Group. All Rights Reserved. Redistribution is prohibited without written permission. L-Stern Group, 2013.

[SSY20] Ani Shabri, Ruhaidah Samsudin, and Yusliza Yusoff. "Combining Deep Neural Network and Fourier Series for Tourist Arrivals Forecasting". In: *IOP Conference Series: Materials Science and Engineering* 864 (July 2020), p. 012094. DOI: 10.1088/1757-899X/864/1/012094.

[Tia20] Yutong Tian. *Time Series Analysis with ARIMA – ARCH/GARCH Model in R*. Accessed: 2024-08-18. 2020. URL: https://rpubs.com/yyt1017/696507.

[SMR22] Kossi Sako, Barnabé Ngoie Mpinda, and Pedro C. Rodrigues. "Neural Networks for Financial Time Series Forecasting". In: *Entropy* 24.5 (May 2022), p. 657. DOI: 10.3390/e24050657.

[24a] *Apple Inc. (AAPL) Stock Historical Prices Data*. Accessed: 2024-08-18. 2024. URL: https://finance.yahoo.com/quote/AAPL/history/?guccounter=1.

[24b] *Retail Sales: Excluding Food Services (RSXFSN)*. Accessed: 2024-08-18. 2024. URL: https://fred.stlouisfed.org/series/RSXFSN.

R Code

```r
# Load necessary libraries
library(forecast)
library(tseries)
library(tidyverse)
library(readr)
library(ggplot2)
library(zoo)
library(TSA)
library(rugarch)
library(forecast)
library(PerformanceAnalytics)
library(xts)
library(quantmod)

# Load Data
file_path <- "C:\\Users\\maldo\\Downloads\\RSXFSN (2).csv"
retail_sales_data <- read_csv(file_path)

# Convert the DATE column to Date type
retail_sales_data$DATE <- as.Date(retail_sales_data$DATE, format="%Y-%m-%d")

head(retail_sales_data)
summary(retail_sales_data)

# Check the number of rows in the original dataset
num_data_points <- nrow(retail_sales_data)
cat("Number of data points in the original dataset:", num_data_points, "\n")

# Plot the original data
ggplot(data = retail_sales_data, aes(x = DATE, y = RSXFSN)) +
  geom_line(color = "blue") +
  labs(title = "Time Series of Monthly Retail Sales",
       x = "Date",
       y = "Retail Sales") +
  theme_minimal()

# Create a time series object
ts_data <- ts(retail_sales_data$RSXFSN, start = c(1992, 12), frequency = 12)

boxplot(ts_data ~ cycle(ts_data), main="Seasonal Boxplot of Monthly Retail Sales", ylab = "Sal

# Create ACF plot
acf(retail_sales_data$RSXFSN, main = "ACF of Monthly Retail Sales", lag.max = 100)

# Create PACF plot
par(mar=c(5, 5, 4, 2) + 0.1)
pacf(retail_sales_data$RSXFSN, main = "PACF of Monthly Retail Sales", lag.max = 100)

adf_test_result <- adf.test(retail_sales_data$RSXFSN)
print(adf_test_result)

# Perform seasonal decomposition
decomposed <- stl(ts(retail_sales_data$RSXFSN, frequency = 12), s.window = "periodic")
plot(decomposed)
```

```r
# Apply seasonal differencing with a lag of 12
seasonal_diff <- diff(retail_sales_data$RSXFSN, lag = 12)

# Apply first differencing to the seasonally differenced data
combined_diff <- diff(seasonal_diff)

# Convert the differenced data to a time series object for plotting
combined_diff_ts <- ts(combined_diff, frequency = 12)

# Plot the differenced data
plot(combined_diff_ts, main = "Seasonally and First Differenced Time Series",
     ylab = "Differenced Value", xlab = "Time")

# Perform the ADF test
adf_test_result <- adf.test(combined_diff_ts)
print(adf_test_result)

# Perform the KPSS test
kpss_test_result <- kpss.test(combined_diff_ts)
print(kpss_test_result)

# Plot ACF for the differenced data
acf(combined_diff_ts, main="ACF of Differenced RSXFSN", lag.max=50)

# PACF plot for differenced data
pacf(combined_diff_ts, main="PACF of Differenced RSXFSN", lag.max=50)

eacf(combined_diff_ts)

# Fit an ARIMA model (the order can be adjusted based on ACF/PACF analysis)
arima_fit <- auto.arima(combined_diff_ts)

# Print the ARIMA model summary
summary(arima_fit)

# Plot the ARIMA model diagnostics
checkresiduals(arima_fit)

arima_model_204001 <- Arima(combined_diff_ts,
                      order = c(2, 0, 4),
                      seasonal = list(order = c(0, 0, 1), period = 12))

# Print the summary of the fitted model
summary(arima_model_204001)

arima_model_204003 <- Arima(combined_diff_ts,
                      order = c(2, 0, 4),
                      seasonal = list(order = c(0, 0, 3), period = 12))

# Print the summary of the fitted model
summary(arima_model_204003)

arima_model_102001 <- Arima(combined_diff_ts,
                      order = c(1, 0, 2),
                      seasonal = list(order = c(0, 0, 1), period = 12))
```

```
# Print the summary of the fitted model
summary(arima_model_102001)

arima_model_002001 <- Arima(combined_diff_ts,
                      order = c(0, 0, 2),
                      seasonal = list(order = c(0, 0, 1), period = 12))

# Print the summary of the fitted model
summary(arima_model_002001)

arima_model_002002 <- Arima(combined_diff_ts,
                      order = c(0, 0, 2),
                      seasonal = list(order = c(0, 0, 2), period = 12))

# Print the summary of the fitted model
summary(arima_model_002002)

arima_model_101001 <- Arima(combined_diff_ts,
                      order = c(1, 0, 1),
                      seasonal = list(order = c(0, 0, 1), period = 12))

# Print the summary of the fitted model
summary(arima_model_101001)

arima_model_202001 <- Arima(combined_diff_ts,
                      order = c(2, 0, 2),
                      seasonal = list(order = c(0, 0, 1), period = 12))

# Print the summary of the fitted model
summary(arima_model_202001)

arima_model_303001 <- Arima(combined_diff_ts,
                      order = c(3, 0, 3),
                      seasonal = list(order = c(0, 0, 1), period = 12))

# Print the summary of the fitted model
summary(arima_model_303001)

arima_model_303003 <- Arima(combined_diff_ts,
                      order = c(3, 0, 3),
                      seasonal = list(order = c(0, 0, 3), period = 12))

# Print the summary of the fitted model
summary(arima_model_303003)

arima_residuals <- residuals(arima_model_303001)

# Print the first few residuals to inspect them
head(arima_residuals)

# Square the residuals
squared_residuals <- arima_residuals^2

# Plot ACF of squared residuals
acf(squared_residuals, lag.max = 20, main = "ACF of Squared Residuals")
```

```r
# Plot PACF of squared residuals
pacf(squared_residuals, lag.max = 20, main = "PACF of Squared Residuals")

# Calculate ACF and PACF values
acf_values <- acf(squared_residuals, lag.max = 20, plot = FALSE)$acf
pacf_values <- pacf(squared_residuals, lag.max = 20, plot = FALSE)$acf

# Display ACF and PACF values
acf_values
pacf_values

# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = FALSE),
  distribution.model = "norm"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Print the GARCH model summary
summary(garch_fit)

# Plot the GARCH model diagnostics
plot(garch_fit)

# Check if the model converged successfully
summary(garch_fit)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]

cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = FALSE),
  distribution.model = "norm"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)

# Extract coefficients and standard errors
```

```r
coef_table <- garch_fit@fit$matcoef

# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]

cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = FALSE),
  distribution.model = "std"
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)

# Extract coefficients and standard errors
coef_table <- garch_fit@fit$matcoef

# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]
```

```r
cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

plot(garch_fit)

# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(1, 1), include.mean = FALSE),
  distribution.model = "norm"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)

# Extract coefficients and standard errors
coef_table <- garch_fit@fit$matcoef

# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]

cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(1, 0), include.mean = FALSE),
  distribution.model = "std"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)
```

```r
# Extract coefficients and standard errors
coef_table <- garch_fit@fit$matcoef

# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]

cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(2, 0), include.mean = FALSE),
  distribution.model = "norm"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)

# Extract coefficients and standard errors
coef_table <- garch_fit@fit$matcoef

# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]
```

```r
cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(2, 0), include.mean = FALSE),
  distribution.model = "std"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)

# Extract coefficients and standard errors
coef_table <- garch_fit@fit$matcoef

# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]

cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")


# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(2, 2), include.mean = FALSE),
  distribution.model = "norm"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)
```

```r
# Extract coefficients and standard errors
coef_table <- garch_fit@fit$matcoef

# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]

cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(2, 2), include.mean = FALSE),
  distribution.model = "std"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)

# Extract coefficients and standard errors
coef_table <- garch_fit@fit$matcoef

# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]
```

```r
cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

plot(garch_fit)

# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(2, 1), include.mean = FALSE),
  distribution.model = "norm"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)

# Extract coefficients and standard errors
coef_table <- garch_fit@fit$matcoef

# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]

cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(2, 1), include.mean = FALSE),
  distribution.model = "std"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)
```

```r
# Extract coefficients and standard errors
coef_table <- garch_fit@fit$matcoef

# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]

cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

garch_residuals <- residuals(garch_fit, standardize = TRUE)

# Set the maximum number of lags for the Ljung-Box test
max_lags <- 30

# Initialize a vector to store p-values
p_values <- numeric(max_lags)

# Calculate the Ljung-Box test p-values for each lag
for (lag in 1:max_lags) {
  lb_test <- Box.test(garch_residuals, lag = lag, type = "Ljung-Box", fitdf = 0)
  p_values[lag] <- lb_test$p.value
}

# Create a plot of p-values
plot(1:max_lags, p_values, type = "b", pch = 19, col = "blue",
     xlab = "Lag", ylab = "p-value",
     main = "Ljung-Box Test p-values for Standardized Residuals")
abline(h = 0.05, col = "red", lty = 2)  # Add a line at the 0.05 significance level

# Creating a dataframe for plotting
residuals_df <- data.frame(Residuals = as.numeric(garch_residuals))

# Specify the number of periods you want to forecast
forecast_horizon <- 12

# Forecast future values using the SARIMA model
sarima_forecast <- forecast(adjusted_arima_model_2, h = forecast_horizon)

# Plot SARIMA forecast
plot(sarima_forecast, main = "SARIMA Forecast")
```

```r
# Step 4: Forecast using the GARCH model
garch_forecast <- ugarchforecast(garch_fit, n.ahead = forecast_horizon)

# Extract forecasted volatility (standard deviation) from the GARCH model
volatility_forecast <- sigma(garch_forecast)

# Print volatility forecast
print(volatility_forecast)

# Combine SARIMA forecast with GARCH volatility forecast
plot(sarima_forecast, main = "Forecasted Mean with SARIMA", xlab = "Time", ylab = "Values")
lines(volatility_forecast, col = "red", lty = 2)

legend("topright", legend = c("SARIMA Mean", "GARCH Volatility"), col = c("blue", "red"), lty =

arima_residuals <- residuals(arima_model_303003)

# Print the first few residuals to inspect them
head(arima_residuals)

checkresiduals(arima_model_303003)
tsdiag(arima_model_303003)

# Square the residuals
squared_residuals <- arima_residuals^2

# Plot ACF of squared residuals
acf(squared_residuals, lag.max = 20, main = "ACF of Squared Residuals")

# Plot PACF of squared residuals
pacf(squared_residuals, lag.max = 20, main = "PACF of Squared Residuals")

# Calculate ACF and PACF values
acf_values <- acf(squared_residuals, lag.max = 20, plot = FALSE)$acf
pacf_values <- pacf(squared_residuals, lag.max = 20, plot = FALSE)$acf

# Display ACF and PACF values
acf_values
pacf_values

# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(2, 1), include.mean = FALSE),
  distribution.model = "std"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)

# Extract coefficients and standard errors
coef_table <- garch_fit@fit$matcoef
```

```r
# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]

cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(2, 2), include.mean = FALSE),
  distribution.model = "std"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)

# Extract coefficients and standard errors
coef_table <- garch_fit@fit$matcoef

# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]

cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

plot(garch_fit)

garch_residuals <- residuals(garch_fit, standardize = TRUE)

# Set the maximum number of lags for the Ljung-Box test
max_lags <- 30

# Initialize a vector to store p-values
p_values <- numeric(max_lags)

# Calculate the Ljung-Box test p-values for each lag
for (lag in 1:max_lags) {
  lb_test <- Box.test(garch_residuals, lag = lag, type = "Ljung-Box", fitdf = 0)
  p_values[lag] <- lb_test$p.value
}

# Create a plot of p-values
plot(1:max_lags, p_values, type = "b", pch = 19, col = "blue",
     xlab = "Lag", ylab = "p-value",
     main = "Ljung-Box Test p-values for Standardized Residuals")
abline(h = 0.05, col = "red", lty = 2)  # Add a line at the 0.05 significance level

# Creating a dataframe for plotting
residuals_df <- data.frame(Residuals = as.numeric(garch_residuals))

# Specify the number of periods you want to forecast
forecast_horizon <- 12

# Forecast future values using the SARIMA model
sarima_forecast <- forecast(adjusted_arima_model_2, h = forecast_horizon)

# Plot SARIMA forecast
plot(sarima_forecast, main = "SARIMA Forecast")

# Step 4: Forecast using the GARCH model
garch_forecast <- ugarchforecast(garch_fit, n.ahead = forecast_horizon)

# Extract forecasted volatility (standard deviation) from the GARCH model
volatility_forecast <- sigma(garch_forecast)

# Print volatility forecast
print(volatility_forecast)

# Combine SARIMA forecast with GARCH volatility forecast
plot(sarima_forecast, main = "Forecasted Mean with SARIMA", xlab = "Time", ylab = "Values")
lines(volatility_forecast, col = "red", lty = 2)

legend("topright", legend = c("SARIMA Mean", "GARCH Volatility"), col = c("blue", "red"), lty

#ARIMA MODEL 204003
arima_model_204003 <- Arima(combined_diff_ts,
                      order = c(2, 0, 4),
                      seasonal = list(order = c(0, 0, 3), period = 12))
```

```r
# Print the summary of the fitted model
summary(arima_model_204003)
checkresiduals(arima_model_204003)
tsdiag(arima_model_204003)
arima_residuals <- residuals(arima_model_204003)

# Print the first few residuals to inspect them
head(arima_residuals)
# Square the residuals
squared_residuals <- arima_residuals^2

# Plot ACF of squared residuals
acf(squared_residuals, lag.max = 20, main = "ACF of Squared Residuals")

# Plot PACF of squared residuals
pacf(squared_residuals, lag.max = 20, main = "PACF of Squared Residuals")
# Calculate ACF and PACF values
acf_values <- acf(squared_residuals, lag.max = 20, plot = FALSE)$acf
pacf_values <- pacf(squared_residuals, lag.max = 20, plot = FALSE)$acf
# Display ACF and PACF values
acf_values
pacf_values
# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(2, 2), include.mean = FALSE),
  distribution.model = "std"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)

# Extract coefficients and standard errors
coef_table <- garch_fit@fit$matcoef

# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]
```

```r
cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

plot(garch_fit)
garch_residuals <- residuals(garch_fit, standardize = TRUE)

# Set the maximum number of lags for the Ljung-Box test
max_lags <- 30

# Initialize a vector to store p-values
p_values <- numeric(max_lags)

# Calculate the Ljung-Box test p-values for each lag
for (lag in 1:max_lags) {
  lb_test <- Box.test(garch_residuals, lag = lag, type = "Ljung-Box", fitdf = 0)
  p_values[lag] <- lb_test$p.value
}

# Print the list of p-values
print(p_values)

# Create a plot of p-values
plot(1:max_lags, p_values, type = "b", pch = 19, col = "blue",
     xlab = "Lag", ylab = "p-value",
     main = "Ljung-Box Test p-values for Standardized Residuals")
abline(h = 0.05, col = "red", lty = 2)  # Add a line at the 0.05 significance level

# Creating a dataframe for plotting (optional if you need it for further analysis)
residuals_df <- data.frame(Residuals = as.numeric(garch_residuals))

# ARIMA MODEL 303001
arima_model_303001 <- Arima(combined_diff_ts,
                      order = c(3, 0, 3),
                      seasonal = list(order = c(0, 0, 1), period = 12))

# Print the summary of the fitted model
summary(arima_model_303001)
arima_residuals <- residuals(arima_model_303001)
# Print the first few residuals to inspect them
head(arima_residuals)

# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(2, 2), include.mean = FALSE),
  distribution.model = "std"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)

# Extract coefficients and standard errors
```

```r
coef_table <- garch_fit@fit$matcoef

# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")

# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]

cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

plot(garch_fit)

garch_residuals <- residuals(garch_fit, standardize = TRUE)

# Set the maximum number of lags for the Ljung-Box test
max_lags <- 30

# Initialize a vector to store p-values
p_values <- numeric(max_lags)

# Calculate the Ljung-Box test p-values for each lag
for (lag in 1:max_lags) {
  lb_test <- Box.test(garch_residuals, lag = lag, type = "Ljung-Box", fitdf = 0)
  p_values[lag] <- lb_test$p.value
}

# Create a plot of p-values
plot(1:max_lags, p_values, type = "b", pch = 19, col = "blue",
     xlab = "Lag", ylab = "p-value",
     main = "Ljung-Box Test p-values for Standardized Residuals")
abline(h = 0.05, col = "red", lty = 2)  # Add a line at the 0.05 significance level

# Creating a dataframe for plotting
residuals_df <- data.frame(Residuals = as.numeric(garch_residuals))

# Specify the number of periods you want to forecast
forecast_horizon <- 12

# Forecast future values using the SARIMA model
sarima_forecast <- forecast(adjusted_arima_model_2, h = forecast_horizon)

# Plot SARIMA forecast
plot(sarima_forecast, main = "SARIMA Forecast")
```

```r
# Step 4: Forecast using the GARCH model
garch_forecast <- ugarchforecast(garch_fit, n.ahead = forecast_horizon)

# Extract forecasted volatility (standard deviation) from the GARCH model
volatility_forecast <- sigma(garch_forecast)

# Print volatility forecast
print(volatility_forecast)

# Combine SARIMA forecast with GARCH volatility forecast
plot(sarima_forecast, main = "Forecasted Mean with SARIMA", xlab = "Time", ylab = "Values")
lines(volatility_forecast, col = "red", lty = 2)

legend("topright", legend = c("SARIMA Mean", "GARCH Volatility"), col = c("blue", "red"), lty 

# ARIMA MODEL 303003
arima_model_303003 <- Arima(combined_diff_ts,
                         order = c(3, 0, 3),
                         seasonal = list(order = c(0, 0, 3), period = 12))

# Print the summary of the fitted model
summary(arima_model_303003)
arima_residuals <- residuals(arima_model_303003)
# Print the first few residuals to inspect them
head(arima_residuals)

# Specify the GARCH(1,1) model
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(2, 2), include.mean = FALSE),
  distribution.model = "std"  # You can also try "std" for t-distribution
)

# Fit the GARCH model to the ARIMA residuals
garch_fit <- ugarchfit(spec = garch_spec, data = arima_residuals)

# Extract and print the summary
garch_summary <- summary(garch_fit)
print(garch_summary)

# Extract coefficients and standard errors
coef_table <- garch_fit@fit$matcoef

# Extract p-values from the coefficient table
p_values <- coef_table[, 4]

# Print the coefficients and p-values
print(coef_table)

# Output the p-values
cat("P-values for the GARCH(1,1) model parameters:\n")
print(p_values)

# Plot diagnostics
plot(garch_fit, which = "all")
```

```r
# Check AIC, BIC, and other criteria
aic_value <- infocriteria(garch_fit)["Akaike",]
bic_value <- infocriteria(garch_fit)["Bayes",]

cat("AIC:", aic_value, "\n")
cat("BIC:", bic_value, "\n")

plot(garch_fit)

garch_residuals <- residuals(garch_fit, standardize = TRUE)

# Set the maximum number of lags for the Ljung-Box test
max_lags <- 30

# Initialize a vector to store p-values
p_values <- numeric(max_lags)

# Calculate the Ljung-Box test p-values for each lag
for (lag in 1:max_lags) {
  lb_test <- Box.test(garch_residuals, lag = lag, type = "Ljung-Box", fitdf = 0)
  p_values[lag] <- lb_test$p.value
}

# Create a plot of p-values
plot(1:max_lags, p_values, type = "b", pch = 19, col = "blue",
     xlab = "Lag", ylab = "p-value",
     main = "Ljung-Box Test p-values for Standardized Residuals")
abline(h = 0.05, col = "red", lty = 2)  # Add a line at the 0.05 significance level

# Creating a dataframe for plotting
residuals_df <- data.frame(Residuals = as.numeric(garch_residuals))

# Specify the number of periods you want to forecast
forecast_horizon <- 12

# Forecast future values using the SARIMA model
sarima_forecast <- forecast(adjusted_arima_model_2, h = forecast_horizon)

# Plot SARIMA forecast
plot(sarima_forecast, main = "SARIMA Forecast")

# Step 4: Forecast using the GARCH model
garch_forecast <- ugarchforecast(garch_fit, n.ahead = forecast_horizon)

# Extract forecasted volatility (standard deviation) from the GARCH model
volatility_forecast <- sigma(garch_forecast)

# Print volatility forecast
print(volatility_forecast)

# Combine SARIMA forecast with GARCH volatility forecast
plot(sarima_forecast, main = "Forecasted Mean with SARIMA", xlab = "Time", ylab = "Values")
lines(volatility_forecast, col = "red", lty = 2)

legend("topright", legend = c("SARIMA Mean", "GARCH Volatility"), col = c("blue", "red"), lty
## AAPL Monthly Data 2016-2024
# Load & Inspect the Data
# Load the data
aapl_monthly_data <- read.csv("~/Documents/GitHub/MA-641-Course-Project/AAPL_Monthly2016.csv")

# Convert the date column to Date type
aapl_monthly_data$Date <- as.Date(aapl_monthly_data$Date, format="%Y-%m-%d")

# Inspect the data
head(aapl_monthly_data)
summary(aapl_monthly_data)

# Create a time series object
aaplmonthly_ts <- ts(aapl_monthly_data$Close, start=c(2016, 01), end = c(2024, 05), frequency=

# Descriptive Analysis
plot(aaplmonthly_ts, main="Monthly Apple Stock Prices", ylab="Close Price", xlab="Time")
summary(aaplmonthly_ts)

# ACF and PACF Plots
par(mar=c(5, 5, 4, 2) + 0.1)
acf(aaplmonthly_ts, main="ACF of Monthly Apple Stock Prices", lag.max = 72)
pacf(aaplmonthly_ts, main="PACF of Monthly Apple Stock Prices", lag.max = 72)
eacf(aaplmonthly_ts)

# Augmented Dickey-Fuller Test
adf_test <- adf.test(aaplmonthly_ts, alternative="stationary")
print(adf_test)

# Differencing the series if it is not stationary
if (adf_test$p.value > 0.05) {
  ts_data_diff <- diff(aaplmonthly_ts, differences=1)
  adf_test_diff <- adf.test(ts_data_diff, alternative="stationary")
  print(adf_test_diff)

  # Update the time series data to the differenced series
  aaplmonthly_ts <- ts_data_diff
}

# Time Series Plot after Differencing
plot(aaplmonthly_ts, main="Monthly Apple Stock Prices", ylab="Close Price", xlab="Time")

# ACF and PACF Plots
par(mar=c(5, 5, 4, 2) + 0.1)
acf(aaplmonthly_ts, main="ACF of Monthly Apple Stock Prices", lag.max = 72)
pacf(aaplmonthly_ts, main="PACF of Monthly Apple Stock Prices", lag.max = 72)
eacf(aaplmonthly_ts)

# ARIMA Models
# Fit AR model
ar_model <- Arima(aaplmonthly_ts, order=c(2,0,0))

# Fit MA model
ma_model <- Arima(aaplmonthly_ts, order=c(0,0,2))
```

```
# Fit ARMA(1,1,1) model
arma_model1 <- Arima(aaplmonthly_ts, order=c(1,1,1))

# ARMA(2,1,1) Model
arma_model2 <- Arima(aaplmonthly_ts, order=c(2,1,1))

# ARMA(2,1,2) Model
arma_model3 <- Arima(aaplmonthly_ts, order=c(2,1,2))

# Comparing Models using AIC and BIC
models <- list(ar_model, ma_model, arma_model1, arma_model2, arma_model3)
model_names <- c("AR", "MA", "ARIMA(1,1,1)", "ARIMA(2,1,1)", "ARIMA(2,1,2)")

aic_values <- c(sapply(models, AIC))
bic_values <- c(sapply(models, BIC))

comparison <- data.frame(Model=model_names, AIC=aic_values, BIC=bic_values)
print(comparison)

# Perform diagnostics for ARIMA(2,1,1)
par(mar=c(5, 5, 4, 2) + 0.1)
tsdiag(arma_model2, gof.lag = 10, main = "Diagnostics for ARIMA(2,1,1)")
checkresiduals(arma_model2)

# Q-Q plot for ARIMA(2,1,1)
residuals_arma211 <- residuals(arma_model2)
qqnorm(residuals_arma211, main = "Q-Q Plot of Residuals for ARIMA(2,1,1)")
qqline(residuals_arma211, col = "red")

# Forecasting with the ARIMA(2,1,1) model
arma2_forecast <- forecast(arma_model2, h=12)
plot(arma2_forecast, main="Forecasts from ARIMA(2,1,1)")

# GARCH Models
# Create a time series object
aaplmonthly_ts2 <- ts(aapl_monthly_data$Close, start=c(2016, 01), end = c(2024, 05), frequency

# Calculate returns for modeling
returns <- diff(log(aaplmonthly_ts2))
returns <- returns[!is.na(returns)]
plot(returns, main="Monthly Apple Stock Prices", ylab="Close Price", xlab="Time", type="l")

# Specify ARMA(1,0)-GARCH(1,1) model
spec_garch <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                         mean.model = list(armaOrder = c(1, 0)),
                         distribution.model = "norm")

fit_garchAR1 <- ugarchfit(spec = spec_garch, data = returns)

# Specify ARMA(2,0)-GARCH(1,1) model
spec_garch <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                         mean.model = list(armaOrder = c(2, 0)),
                         distribution.model = "norm")

fit_garchAR2 <- ugarchfit(spec = spec_garch, data = returns)
```

```
# Specify ARMA(0,2)-GARCH(1,1) model
spec_garch <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                         mean.model = list(armaOrder = c(0, 2)),
                         distribution.model = "norm")

fit_garchMA2 <- ugarchfit(spec = spec_garch, data = returns)

# Specify ARMA(2,1)-GARCH(1,1) model
spec_garch <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                         mean.model = list(armaOrder = c(2, 1)),
                         distribution.model = "norm")

fit_garchARMA21 <- ugarchfit(spec = spec_garch, data = returns)

# Specify ARMA(2,2)-GARCH(1,1) model
spec_garch <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                         mean.model = list(armaOrder = c(2, 2)),
                         distribution.model = "norm")

fit_garchARMA22 <- ugarchfit(spec = spec_garch, data = returns)

# Comparing Models using AIC and BIC
model_names <- c("ARFIMA(1,0)-GARCH(1,1)", "ARFIMA(2,0)-GARCH(1,1)", "ARFIMA(0,2)-GARCH(1,1)",

aic_values <- c(-2.0775, -2.0822, -2.0873, -2.0705, -2.0843)
bic_values <- c(-1.9473, -1.9259, -1.9310, -1.8882, -1.8759)

comparison <- data.frame(Model=model_names, AIC=aic_values, BIC=bic_values)
print(comparison)

# GARCH Model Assumptions Check
# Extract standardized residuals
std_residuals <- residuals(fit_garchMA2, standardize = TRUE)

# Remove time index, if present
std_residuals <- as.numeric(std_residuals)
checkresiduals(std_residuals)

# Plot diagnostics
plot(fit_garchMA2, which = 1)
plot(fit_garchMA2, which = 2)
plot(fit_garchMA2, which = 3)
plot(fit_garchMA2, which = 4)
plot(fit_garchMA2, which = 5)
plot(fit_garchMA2, which = 6)
plot(fit_garchMA2, which = 7)
plot(fit_garchMA2, which = 8)
plot(fit_garchMA2, which = 9)
plot(fit_garchMA2, which = 10)
plot(fit_garchMA2, which = 11)
plot(fit_garchMA2, which = 12)

# Forecasting with the GARCH model
forecast_garch <- ugarchforecast(fit_garchMA2, n.ahead=12)
plot(forecast_garch, which=1)  # Forecast series
```