Modeling and Forecasting Walmart Stock Prices: A Comparative Analysis of ARMA and GARCH Approaches

Shrivats Sudhir

December 16, 2023

Contents

1 Introduction

Understanding stock price dynamics is crucial for informed investment decisions. This study models and forecasts Walmart Inc.'s (WMT) daily adjusted closing prices from `2020-01-01` to `2023-12-06` with the following objectives: (1) Compute log returns, (2) Find the optimal `ARIMA` and `GARCH` models to capture volatility clustering and conditional heteroskedasticity, (3) Fit an ensemble Validate models using AIC, BIC, and residual diagnostics, and (4) Generate a 10-step ahead forecast to provide actionable insights for investors and risk managers by modeling and forecasting stock price movements.

Let P_t be the price of an asset at time t, then the log returns is defined as:

$$r_t = \log P_t - \log P_{t-1}$$

The code for computing log returns and bollinger bands can be found here.

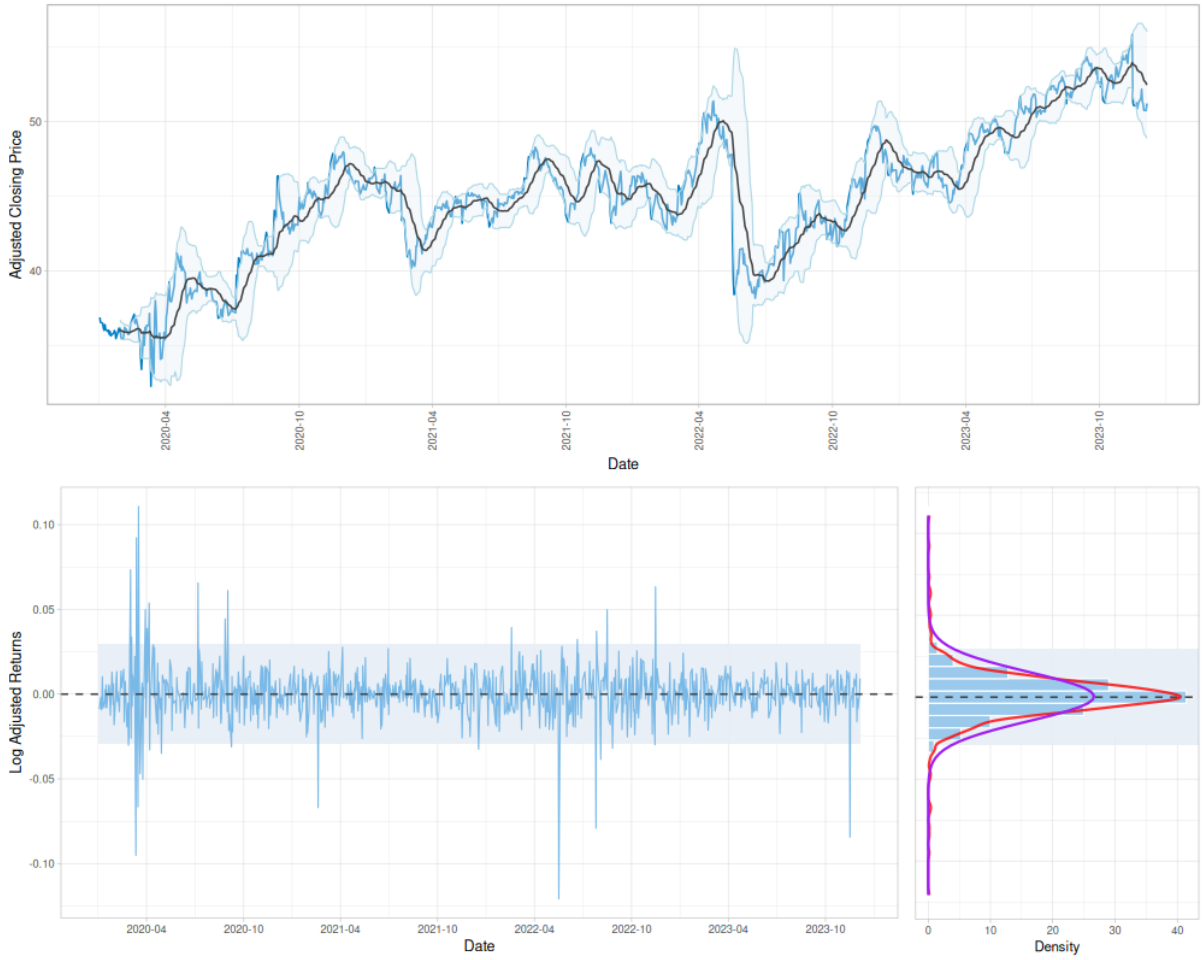

Figure 1: Walmart Inc.'s adjusted closing prices with Bollinger Bands, log returns with a 95% confidence interval, and a histogram of log returns with KDE and theoretical normal distribution.

From Figure 1, there seems to exist some volatility clusters that need to be addressed. Furthermore, the log return distribution seems to be Leptokurtic in nature, illustrating heavy tails and deviations from normality. We now proceed with rigorously testing for normality and stationarity.

2 Fitting `ARIMA` Model

The `forecast` package in R allows us to use `auto.arima` which automatically selects the best model based on AIC/BIC.

Model	**ARIMA(0,0,1) with zero mean**
MA1 Coefficient	-0.0666
Standard Error	0.0312
σ^2	0.0002266
Log-Likelihood	2694.43
AIC	-5384.85
AICc	-5384.84
BIC	-5375.1

Table 1: Summary of fitted ARMA model.

The fitted `ARIMA(0,0,1)` model for Walmart Inc.'s log returns is summarized in Table 1, with a moving average coefficient of -0.0666 (standard error $= 0.0312$), indicating weak short-term autocorrelation. The residual variance is estimated as $\sigma^2 = 0.0002266$, and model selection criteria, including AIC (-5384.85) and BIC (-5375.1), confirm its suitability.

In mathematical terms, we can write the `ARIMA` model as:

$$r_t = a_t - (-0.0666) \cdot a_{t-1}$$

where $a_t \sim N(0, 0.0312)$ and $\mathbb{E}[r_t] = 0$.

3 Fitting `ARCH/GARCH` Models

The `fGarch` library in R allows us to fit various `GARCH` models and compare performances between the them. For the purposes of keeping the models simple and explainable, the focus is on `GARCH`(1,0) and `GARCH`(1,1), using conditional distributions from `c("norm", "ged", "std", "snorm", "sged", "sstd")`.

Model	`GARCH(1,0), cond.dist="std"`
mu	6.090e-04
omega	1.505e-04
alpha1	2.988e-01
shape	3.436e+00
Log-Likelihood	2908.097
AIC	-5.987829
BIC	-5.967716
SIC	-5.987863
HQIC	-5.980173

Model	`GARCH(1,1), cond.dist="std"`
mu	5.220e-04
omega	1.560e-05
alpha1	1.291e-01
beta1	7.905e-01
shape	4.062e+00
Log-Likelihood	2929.534
AIC	-6.029967
BIC	-6.004826
SIC	-6.030020
HQIC	-6.020398

Table 2: Summary of fitted `GARCH(1,0)` and `GARCH(1,1)` model, with students-t conditional distribution.

Table 2 summarizes the fitted `GARCH(1,0)` and `GARCH(1,1)` models using a Student-t conditional distribution, which among all other conditional distribution options best maximizes log-likelihood with reasonable AIC, BIC, SIC, HQIC values. Although `GARCH(1,0)` achieves slightly lower AIC (-5.99 vs. -6.03) and BIC (-5.97 vs. -6.00), indicating a marginally better balance between model fit and complexity, we find that `GARCH(1,1)` achieves a higher log-likelihood (2929.534 vs. 2908.097) and captures long-term volatility persistence through the additional β_1 parameter, which is crucial for financial time series exhibiting volatility clustering. Therefore, we prefer `GARCH(1,1)` over `GARCH(1,0)`.

In mathematical terms, we can write the `GARCH(1,1)` model as:

$$a_t = \sigma_t \epsilon_t, \quad \sigma_t^2 = (1.560\text{e-}05) + (1.291\text{e-}01)a_{t-1}^2 + (7.905\text{e-}01)\sigma_{t-1}^2$$

where $\epsilon_t \sim \text{std}(0, 1)$ with shape parameter 4.0624. The relatively large value of β_1 (0.7905) relative to α_1 (0.1291) reflects the long memory in volatility, which is consistent with financial time series exhibiting volatility clustering.

4 Fitting Ensemble `ARIMA(0,0,1)` + `GARCH(1,1)` Model

A preliminary `ARIMA(0,0,1)` + `GARCH(1,1)` model with a Student-t conditional distribution is fitted to the log returns to capture volatility clustering and heavy tails. We iteratively detect and remove influential points (standardized residuals exceeding a threshold of ± 3) by fitting until the process continues until no new influential points are identified, or the changes in residual diagnostics become negligible.

Model	`ARIMA(0,0,1) + GARCH(1,1)`
mu	4.9198e-04
omega	6.3505e-06
alpha1	1.0224e-01
beta1	8.5001e-01
shape	1.0000e+01
Log-Likelihood	2972.232
AIC	-6.246803
BIC	-6.221243
SIC	-6.246858
HQIC	-6.237065

Table 3: Summary of fitted `ARMA(0,0,1)+GARCH(1,1)` model.

From Table 3, we can see that there is a substantial increase in log-likelihood compared to individual `GARCH(1,1)` or `ARIMA(0,0,1)` models, with only slight increase in AIC and BIC. Furthermore, the statistically significant β_1 value (0.8501) indicates strong long-term persistence in volatility, while the moderate statistically significant α_1 value (0.1022) captures short-term shocks effectively.

5 Residual Diagnostics

Figure 2 show that the fitted ensemble model is satisfactory. The residuals over time are random around zero, the histogram aligns with a normal distribution, the Q-Q plot shows minimal deviation from normality, and the ACF plots confirm no significant autocorrelation in residuals, squared residuals, or absolute residuals, indicating no remaining structure or volatility clustering.

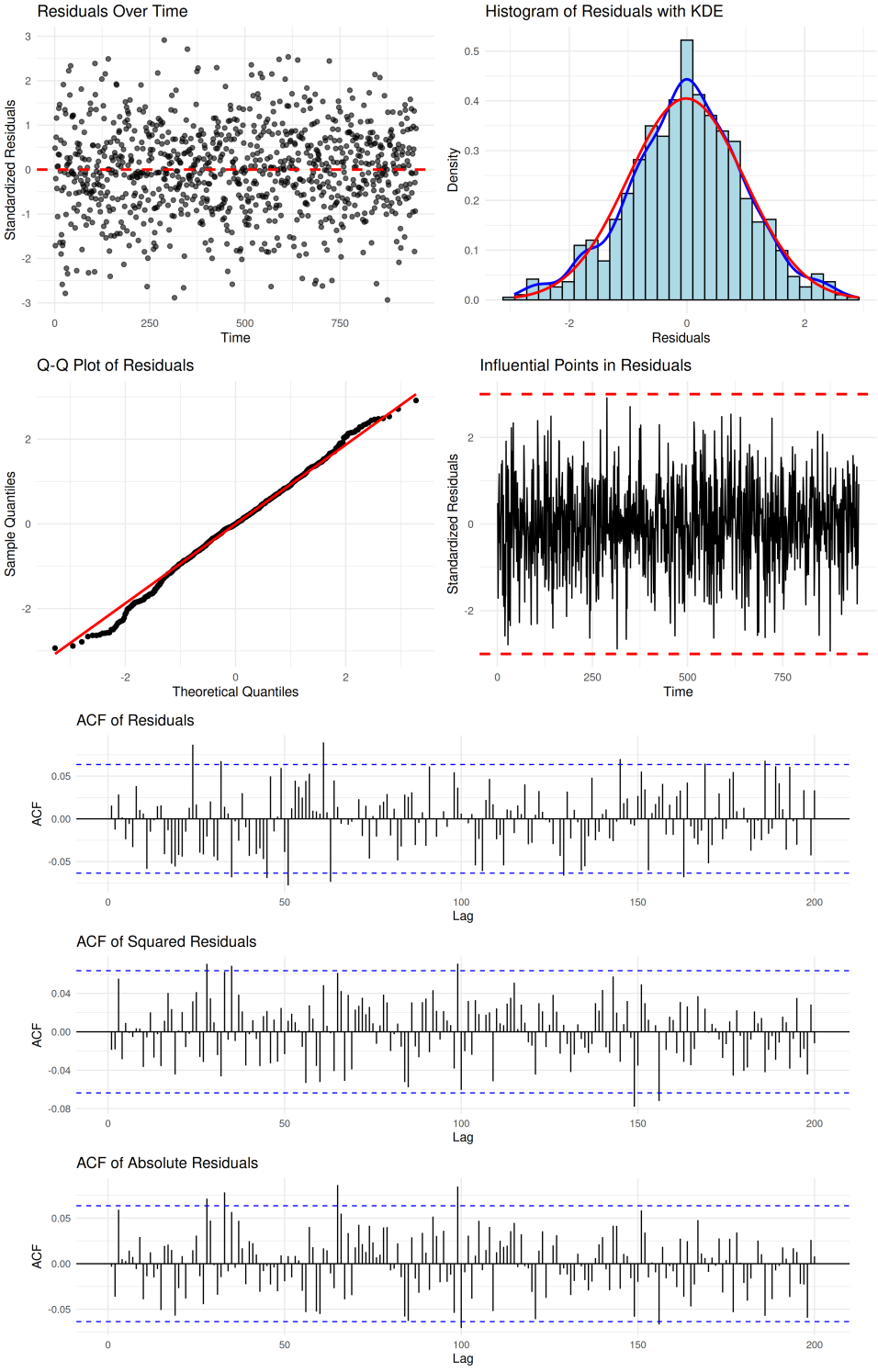

Figure 2: Residual diagnostics for the `ARIMA(0,0,1) + GARCH(1,1)` model.

6 10-Day Forecast

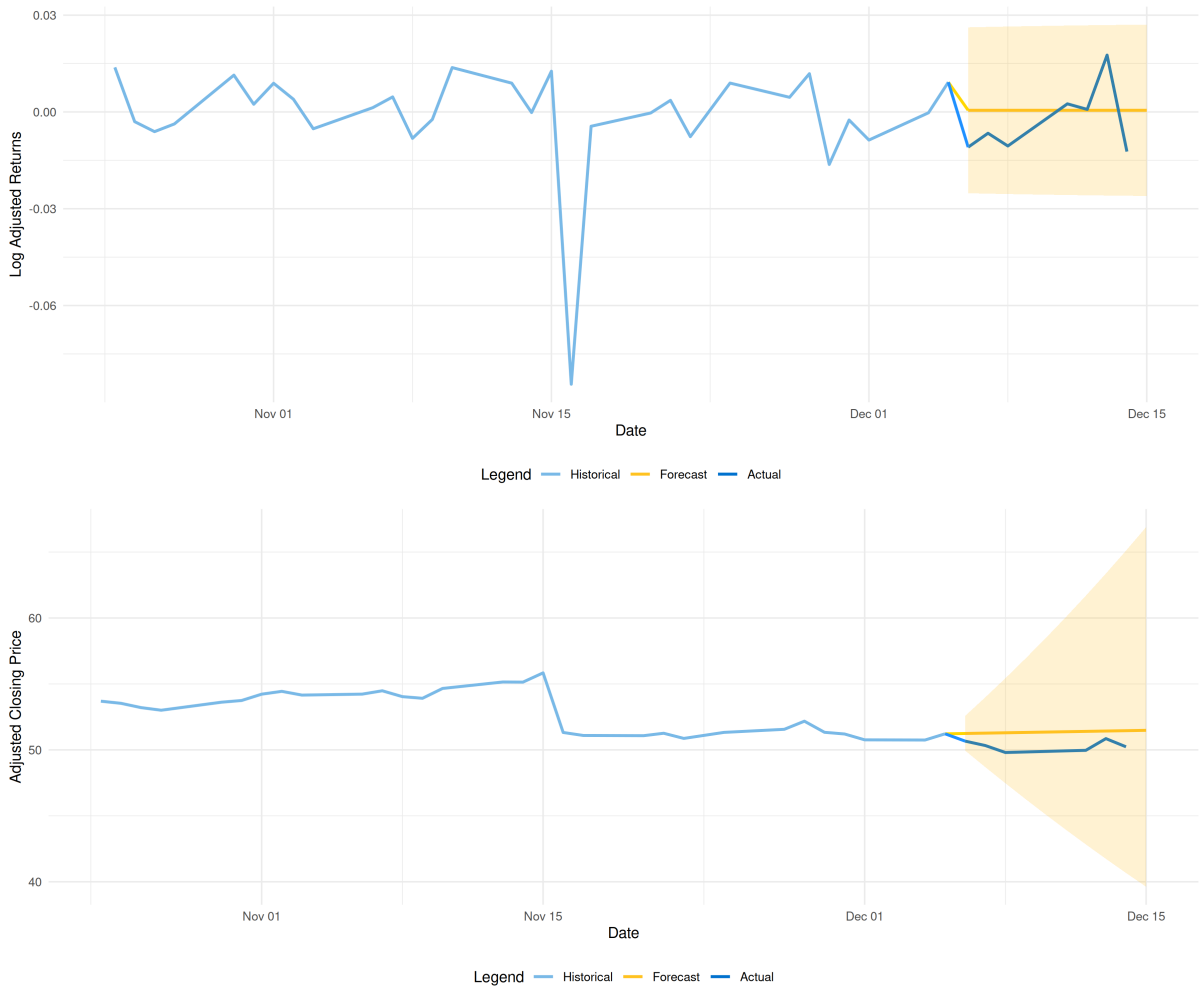

Figure 3: 10-day forecast for log returns and adjusted closing prices, including 95% confidence intervals, alongside historical and actual data for comparison. (last 30+10 days only). Blue represents historical data, yellow indicates the forecast, and dark blue corresponds to actual observed values.

Figure 3 allows us to compare our 10-day forecast with the true log returns and adjusted closing price. It is crucial to note that the model only fits data from `2020-01-01` to `2023-12-05`. The actual log returns from `2023-12-06` to `2023-12-15` is used to compare model performance on unseen data.

The forecasted log returns align well with the actual values, with all observations falling within the 95% confidence interval. This indicates that the `ARIMA(0,0,1)` + `GARCH(1,1)` model effectively captures the short-term dynamics of the stock. However, the increasing width of the confidence interval for closing prices reflects the compounding effect of forecast uncertainty over time. The RMSE for the 10-day forecast, actual log returns, and adjusted prices were $ 0.0103 and $ 1.1693, respectively.

Given the residual diagnostics in Figure 2 affirmation that the `ARIMA(0,0,1)` + `GARCH(1,1)` model effectively captures the underlying structure of Walmart's stock returns, we can conclude that this is a good and effective model for forecasting.

7 Reference

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 31(3), 307-327. https://doi.org/10.1016/0304-4076(86)90063-1.

Box, George E. P., et al. Time Series Analysis: Forecasting and Control. United Kingdom, Wiley, 2008. https://www.google.com/books/edition/Time_Series_Analysis/lJnnPQAACAAJ?hl=en.

Campbell, John Y., et al. The Econometrics of Financial Markets. Princeton University Press, 1997. JSTOR, https://doi.org/10.2307/j.ctt7skm5.

Engle, Robert F. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. Econometrica, vol. 50, no. 4, 1982, pp. 987–1007. JSTOR, https://doi.org/10.2307/1912773.

Hansen, P. R., & Lunde, A. (2005). A forecast comparison of volatility models: Does anything beat a GARCH(1,1)? Journal of Applied Econometrics, 20(7), 873-889. https://doi.org/10.1002/jae.800.

Pfaff, Bernhard. (2008). Analysis of Integrated and Cointegrated Time Series with R. DOI: https://doi.org/10.1007/978-0-387-75967-8.

Poon, Ser-Huang, and Clive W.J. Granger. 2003. Forecasting Volatility in Financial Markets: A Review. Journal of Economic Literature, 41 (2): 478–539. DOI: https://doi.org/10.1257/002205103765762743.

8 Appendix

8.1 Code for R Library Imports

```
rm(list = ls())

# Install required libraries
if (!require("quantmod")) install.packages("quantmod", dependencies = TRUE)
if (!require("fBasics")) install.packages("fBasics", dependencies = TRUE)
if (!require("fGarch")) install.packages("fGarch", dependencies = TRUE)
if (!require("timeSeries")) install.packages("timeSeries", dependencies = TRUE)
if (!require("zoo")) install.packages("zoo", dependencies = TRUE)
if (!require("reactable")) install.packages("reactable", dependencies = TRUE)
if (!require("ggplot2")) install.packages("ggplot2", dependencies = TRUE)
if (!require("grid")) install.packages("grid", dependencies = TRUE)
if (!require("gridExtra")) install.packages("gridExtra", dependencies = TRUE)
if (!require("tseries")) install.packages("tseries", dependencies = TRUE)
if (!require("forecast")) install.packages("forecast", dependencies = TRUE)
if (!require("dplyr")) install.packages("dplyr", dependencies = TRUE)

library(quantmod)
library(fBasics)
library(fGarch)
library(timeSeries)
library(zoo)
library(reactable)
library(ggplot2)
library(grid)
library(gridExtra)
library(tseries)
library(forecast)
library(dplyr)
```

8.2 Code for Loading Data:

```
getSymbols("WMT", from = "2020-01-01", to = "2023-12-06")
AdjClose = Ad(WMT) # Adjusted Close Prices

# Create a data frame with adjusted close prices and log returns
WMT_df <- data.frame(
  Date = index(AdjClose),
  AdjClose = coredata(AdjClose),
```

```
  LogReturns = c(NA, diff(log(coredata(AdjClose))))
)

# Rename columns for clarity
colnames(WMT_df) <- c("Date", "AdjClose", "LogReturns")

# Load csv
WMT_df <- read.csv("Walmart_AdjPrice.csv")
WMT_df$Date <- as.Date(WMT_df$Date, format = "%Y-%m-%d")

# Calculate Bollinger Bands
rolling_mean <- rollmean(WMT_df$AdjClose, k = 20, fill = NA, align = "right")
rolling_sd <- rollapply(WMT_df$AdjClose, width = 20, FUN = sd,
                        fill = NA, align = "right")
bollinger_upper <- rolling_mean + 2 * rolling_sd
bollinger_lower <- rolling_mean - 2 * rolling_sd

# Add Bollinger Bands to the data frame
WMT_df$BollingerMean <- rolling_mean
WMT_df$BollingerUpper <- bollinger_upper
WMT_df$BollingerLower <- bollinger_lower

WMT_df <- na.omit(WMT_df)
```

8.3 Code for Figure 1:

```
# Compute empirical statistics for Walmart's log returns
empirical_mean <- mean(WMT_df$LogReturns, na.rm = TRUE)
empirical_sd <- sd(WMT_df$LogReturns, na.rm = TRUE)
ci_lower <- empirical_mean - 1.96 * empirical_sd
ci_upper <- empirical_mean + 1.96 * empirical_sd

# Normal distribution for comparison
normal_x <- seq(min(WMT_df$LogReturns,
                    na.rm = TRUE),
                max(WMT_df$LogReturns,
                    na.rm = TRUE), length.out = 100)
normal_y <- dnorm(normal_x, mean = empirical_mean, sd = empirical_sd)

# Plot 1: Adjusted Close Prices with Bollinger Bands
price_plot <- ggplot(WMT_df, aes(x = Date)) +
  geom_line(aes(y = AdjClose), linewidth = 0.7,
```

```
            na.rm = TRUE, color = "#007dc6") +
  geom_ribbon(aes(ymin = BollingerLower, ymax = BollingerUpper),
              alpha = 0.4, fill = "#e7f0f7", na.rm = TRUE) +
  geom_line(aes(y = BollingerUpper), linetype = "solid",
            linewidth = 0.5, na.rm = TRUE, color = "lightblue") +
  geom_line(aes(y = BollingerLower), linetype = "solid",
            linewidth = 0.5, na.rm = TRUE, color = "lightblue") +
  geom_line(aes(y = BollingerMean),
            linewidth = 0.7, na.rm = TRUE, color = "#444444") +
  theme_light() +
  labs(x = "Date", y = "Adjusted Closing Price") +
  scale_x_date(date_breaks = "6 months", date_labels = "%Y-%m") +
  theme(
    axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),
    plot.title = element_text(size = 14, face = "bold"),
    axis.title = element_text(size = 12),
    legend.position = "none" # Remove legend
  )

# Plot 2: Time-Series of Log Returns
time_series_plot <- ggplot(WMT_df, aes(x = Date)) +
  geom_rect(aes(xmin = min(Date, na.rm = TRUE),
                xmax = max(Date, na.rm = TRUE),
                ymin = ci_lower,
                ymax = ci_upper), alpha = 0.3, fill = "#e7f0f7") +
  geom_line(aes(y = LogReturns),
            linewidth = 0.5, na.rm = TRUE, color = "#79b9e7") +
  geom_hline(yintercept = 0.0,
             color = "#444", linetype = "dashed", linewidth = 0.7) +
  theme_light() +
  labs(x = "Date", y = "Log Adjusted Returns") +
  scale_x_date(date_breaks = "6 months", date_labels = "%Y-%m") +
  theme(
    legend.position = "none", # Remove legend
    axis.title = element_text(size = 12)
  )

# Plot 3: Rotated Histogram of Log Returns
rotated_histogram <- ggplot(WMT_df, aes(x = LogReturns)) +
  geom_rect(aes(xmin = ci_lower, xmax = ci_upper),
            ymin = 0, ymax = Inf, alpha = 0.3, fill = "#e7f0f7") +
  geom_histogram(aes(y = after_stat(density)),
```

```
                 binwidth = 0.0075, color = "white",
                 alpha = 0.7, fill = "#79b9e7", na.rm = TRUE) +
  geom_vline(xintercept = 0.0, color = "#444",
             linetype = "dashed", linewidth = 0.7) +
  stat_density(geom = "line", color = "red",
               linewidth = 1, alpha = 0.8, na.rm = TRUE) +
  geom_line(data = data.frame(x = normal_x, y = normal_y),
            aes(x = x, y = y), color = "purple", linewidth = 1) +
  coord_flip() +
  labs(x = "", y = "Density") +
  theme_light() +
  theme(
    legend.position = "none", # Remove legend
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    axis.title.y = element_blank()
  )

# Combine bottom row
bottom_row <- arrangeGrob(grobs = list(time_series_plot, rotated_histogram),
                          ncol = 2, widths = c(3, 1))

# Arrange the final layout
final_layout <- grid.arrange(
  price_plot,       # Top row
  bottom_row,       # Bottom two plots
  nrow = 2,         # Two rows
  heights = c(1, 1) # Adjust row heights
)
```

8.4 Code for Table 1:

```
# Fit best ARMA model automatically
arma_fit <- auto.arima(WMT_df$LogReturns, seasonal = FALSE)
print(summary(arma_fit))
```

8.5 Code for Table 2:

```
# Fit ARCH / GARCH(1,0)
fit_arch <- garchFit(~ garch(1, 0), data = WMT_df$LogReturns, cond.dist =
    "std")
summary(fit_arch)
```

```r
# Fit GARCH(1,1)
fit_garch <- garchFit(~ garch(1, 1), data = WMT_df$LogReturns, cond.dist =
    "std")
summary(fit_garch)
```

8.6 Code for Table 3:

```r
# Initialize cleaned data
data_cleaned <- WMT_df$LogReturns

# Threshold for influential points (standardized residuals)
threshold <- 3

# Iterative process to remove influential points
iteration <- 1
repeat {
  cat("\nIteration:", iteration, "\n")

  # Fit ARMA+GARCH(1,1) model
  arma_garch_model <- garchFit(~arma(0,0,1) + garch(1,1),
                               data = data_cleaned,
                               cond.dist = "std",
                               trace = FALSE)

  # Extract standardized residuals
  residuals_model <- residuals(arma_garch_model, standardize = TRUE)

  # Identify influential points
  influential_points <- which(abs(residuals_model) > threshold)
  cat("Number of influential points detected:", length(influential_points),
      "\n")

  # Stop if no more influential points are found
  if (length(influential_points) == 0) {
    cat("No more influential points. Stopping iteration.\n")
    break
  }

  # Remove influential points and refit
  data_cleaned <- data_cleaned[-influential_points]
  cat("Removed influential points at indices:", influential_points, "\n")
```

```
  iteration <- iteration + 1
}

# Final model with cleaned data
arma_garch_model <- garchFit(~arma(0,0,1) + garch(1,1),
                                    data = data_cleaned,
                                    cond.dist = "std",
                                    trace = FALSE)

# Model summary
print(summary(arma_garch_model))
```

8.7 Code for Figure 2

```
# Extract standardized residuals for the cleaned model
residuals_cleaned <- residuals(arma_garch_model, standardize=TRUE)

# 1. Residuals over time
residuals_time <- data.frame(Time=1:length(residuals_cleaned),
                             Residuals=residuals_cleaned)
p1 <- ggplot(residuals_time, aes(x=Time, y=Residuals)) +
  geom_point(alpha=0.6) +
  geom_hline(yintercept=0, color="red", linetype="dashed", linewidth=1) +
  labs(title="Residuals Over Time", x="Time", y="Standardized Residuals") +
  theme_minimal()

# 2. Residual histogram with KDE and theoretical distribution
p2 <- ggplot(data.frame(Residuals = residuals_cleaned), aes(x = Residuals)) +
  geom_histogram(aes(y = after_stat(density)),
                 bins=30, fill="lightblue", color="black") +
  geom_density(color = "blue", linetype = "solid", linewidth=1) +
  stat_function(fun=dnorm, args=list(mean=mean(residuals_cleaned),
                                     sd=sd(residuals_cleaned)),
                color="red", linetype="solid", linewidth=1) +
  labs(title = "Histogram of Residuals with KDE", x = "Residuals", y =
      "Density") +
  theme_minimal()

# Q-Q Plot of residuals
p3 <- ggplot(data.frame(Residuals = residuals_cleaned), aes(sample =
    Residuals)) +
```

```
  stat_qq() +
  stat_qq_line(color = "red", linewidth=1) +
  labs(title = "Q-Q Plot of Residuals", x = "Theoretical Quantiles", y =
      "Sample Quantiles") +
  theme_minimal()

# Influential points graph (standardized residuals > 3 or < -3)
influential_points_cleaned <- which(abs(residuals_cleaned) > 3)
influential_residuals <- residuals_time[influential_points_cleaned, ]

p4 <- ggplot(residuals_time, aes(x = Time, y = Residuals)) +
  geom_line() +
  geom_hline(yintercept = c(-3, 3), color = "red",
             linetype = "dashed", linewidth=1) +
  labs(title = "Influential Points in Residuals", x = "Time", y =
      "Standardized Residuals") +
  theme_minimal()

# Display the plots
residual_plots <- grid.arrange(p1, p2,
                               p3, p4, ncol=2, nrow=2)

# ACF plots (Residuals, Squared Residuals, Absolute Residuals)
acf_residuals_plot <- ggAcf(residuals_cleaned, lag.max=200) +
  ggtitle("ACF of Residuals") +
  theme_minimal()

# ACF of squared residuals
acf_squared_residuals_plot <- ggAcf(residuals_cleaned^2, lag.max=200) +
  ggtitle("ACF of Squared Residuals") +
  theme_minimal()

# ACF of absolute residuals
acf_abs_residuals_plot <- ggAcf(abs(residuals_cleaned), lag.max=200) +
  ggtitle("ACF of Absolute Residuals") +
  theme_minimal()

# Display the plots
ACF_plots <- grid.arrange(acf_residuals_plot,
                          acf_squared_residuals_plot,
                          acf_abs_residuals_plot, ncol = 1)
```

8.8 Code for Figure 3:

```
# Forecast parameters
forecast_length <- 10
last_date <- max(WMT_df$Date)
last_price <- tail(WMT_df$AdjClose, 1)
last_log_return <- tail(WMT_df$LogReturns, 1)

# Forecast log returns
forecasts <- predict(arma_garch_model, n.ahead = forecast_length, plot = FALSE)

# Create forecast dataframe
forecast_df <- data.frame(
  Date = seq(last_date + 1, by = 1, length.out = forecast_length),
  LogReturns = forecasts$meanForecast,
  Lower95 = forecasts$meanForecast - 1.96 * forecasts$standardDeviation,
  Upper95 = forecasts$meanForecast + 1.96 * forecasts$standardDeviation
)

# Calculate prices and confidence intervals
forecast_df$AdjPrice <- last_price * exp(cumsum(forecast_df$LogReturns))
forecast_df$LowerAdjPrice <- last_price * exp(cumsum(forecast_df$Lower95))
forecast_df$UpperAdjPrice <- last_price * exp(cumsum(forecast_df$Upper95))

# Fetch actual recent prices
getSymbols("WMT", from = last_date,
           to = last_date + forecast_length,
           auto.assign = TRUE)

# Prepare actual prices dataframe
actual_prices <- data.frame(
  Date = index(WMT),
  AdjClose = as.numeric(Ad(WMT))
)
actual_prices$LogReturns <- c(NA, diff(log(actual_prices$AdjClose)))
actual_prices <- na.omit(actual_prices)

# Prepare recent historical data
recent_historical_data <- tail(WMT_df, 30)

# Prepare connecting lines for price
connect_price <- data.frame(
  Date = c(last_date, forecast_df$Date[1], last_date, actual_prices$Date[1]),
```

```
  AdjClose = c(last_price, forecast_df$AdjPrice[1],
               last_price, actual_prices$AdjClose[1]),
  Connector = c("Forecast Connector", "Forecast Connector",
                "Historical Connector", "Historical Connector")
)

# Prepare connecting lines for log returns
connect_log_returns <- data.frame(
  Date = c(last_date, forecast_df$Date[1], last_date, actual_prices$Date[1]),
  LogReturns = c(last_log_return, forecast_df$LogReturns[1],
                 last_log_return, actual_prices$LogReturns[1]),
  Connector = c("Forecast Connector", "Forecast Connector",
                "Historical Connector", "Historical Connector")
)

# Convert to date
connect_price$Date <- as.Date(connect_price$Date)
connect_log_returns$Date <- as.Date(connect_log_returns$Date)

palette <- c(
  "Historical" = "#79b9e7",
  "Forecast" = "#FFC220",
  "Actual" = "#0071CE",
  "Forecast Connector" = "#FFD700",  # Yellow
  "Historical Connector" = "#1E90FF" # Blue
)
# Price Plot
price_plot <- ggplot() +
  # Historical price line
  geom_line(data = recent_historical_data,
            aes(x = Date, y = AdjClose, color = "Historical"), size = 1) +
  # Forecast price line
  geom_line(data = forecast_df,
            aes(x = Date, y = AdjPrice, color = "Forecast"), size = 1) +
  # Actual market prices line
  geom_line(data = actual_prices,
            aes(x = Date, y = AdjClose, color = "Actual"), size = 1) +
  # Connecting lines (with legend suppressed)
  geom_line(data = connect_price,
            aes(x = Date, y = AdjClose, color = Connector, linetype =
                Connector),
            size = 1, show.legend = FALSE) +
```

```
  # Confidence interval ribbon
  geom_ribbon(data = forecast_df,
              aes(x = Date, ymin = LowerAdjPrice, ymax = UpperAdjPrice),
              fill = "#FFC220", alpha = 0.2) +
  scale_color_manual(values = palette,
                     name = "Legend",
                     breaks = c("Historical", "Forecast", "Actual")) +
  scale_linetype_manual(values = c("Forecast Connector" = "solid",
                                   "Historical Connector" = "solid")) +
  labs(
    x = "Date",
    y = "Adjusted Closing Price"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    legend.position = "bottom",
  )

# Log Returns Plot
log_return_plot <- ggplot() +
  # Historical log returns
  geom_line(data = recent_historical_data,
            aes(x = Date, y = LogReturns, color = "Historical"), size = 1) +
  # Forecast log returns
  geom_line(data = forecast_df,
            aes(x = Date, y = LogReturns, color = "Forecast"), size = 1) +
  # Actual market log returns
  geom_line(data = actual_prices,
            aes(x = Date, y = LogReturns, color = "Actual"), size = 1) +
  # Connecting lines (with legend suppressed)
  geom_line(data = connect_log_returns,
            aes(x = Date, y = LogReturns, color = Connector, linetype =
                Connector),
            size = 1, show.legend = FALSE) +
  # Confidence interval ribbon
  geom_ribbon(data = forecast_df,
              aes(x = Date, ymin = Lower95, ymax = Upper95),
              fill = "#FFC220", alpha = 0.2) +
  scale_color_manual(values = palette,
                     name = "Legend",
                     breaks = c("Historical", "Forecast", "Actual")) +
```

```
  scale_linetype_manual(values = c("Forecast Connector" = "solid",
                                   "Historical Connector" = "solid")) +
  labs(
    x = "Date",
    y = "Log Adjusted Returns"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    legend.position = "bottom",
  )

# Combine and save plots
forecast_plot <- grid.arrange(log_return_plot, price_plot, ncol = 1)

# Align forecast_df and actual_prices by Date
aligned_data <- inner_join(forecast_df, actual_prices, by = "Date", suffix =
    c("_forecast", "_actual"))

# Calculate RMSE for log returns
log_returns_rmse <- sqrt(mean((aligned_data$LogReturns_forecast -
    aligned_data$LogReturns_actual)^2, na.rm = TRUE))
cat("RMSE for Log Returns: ", log_returns_rmse, "\n")

# Calculate RMSE for adjusted prices
price_rmse <- sqrt(mean((aligned_data$AdjPrice - aligned_data$AdjClose)^2,
    na.rm = TRUE))
cat("RMSE for Adjusted Prices: ", price_rmse, "\n")
```

8.9 Public GitHub Repository

All codes and images associated with this project can be found in the GitHub repository below:

https://github.com/Stochastic1017/Walmart-Stock-Forecasting

Note: The repository will be made public after ***December 18, 2024 (00:00 AM)*** *to ensure academic integrity by preventing public access to these materials before the due date.*

Figure/Table	**Code**
Figure 1	Code for Figure 1
Table 1	Code for Table 1
Table 2	Code for Table 2
Table 3	Code for Table 3
Figure 2	Code for Figure 2
Figure 3	Code for Figure 3

For a more comprehensive analysis and consolidated access to all the code used in the manner in which it was intended, the following files are available in the repository. Note that these greatly exceeds the 5-page limit and consolidates all R codes and outputs used for this project:

File	**URL**
`Consolidated-Walmart-Stock-Forecasting.Rmd`	Rmd File Link
`Consolidated-Walmart-Stock-Forecasting.pdf`	PDF File Link

Finally, the LaTeX file and the rendered pdf file associated to this submission write-up is available in this repository:

File	**URL**
`Walmart-Stock-Forecasting.tex`	TEX File Link
`Walmart-Stock-Forecasting.pdf`	PDF File Link