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

Shrivats Sudhir

December 16th, 2023

```r
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)
```

```r
getSymbols("WMT", from = "2020-01-01", to = "2023-12-06")
```

```
## [1] "WMT"
```

```r
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))))
)
```

```r
# 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)
```

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.

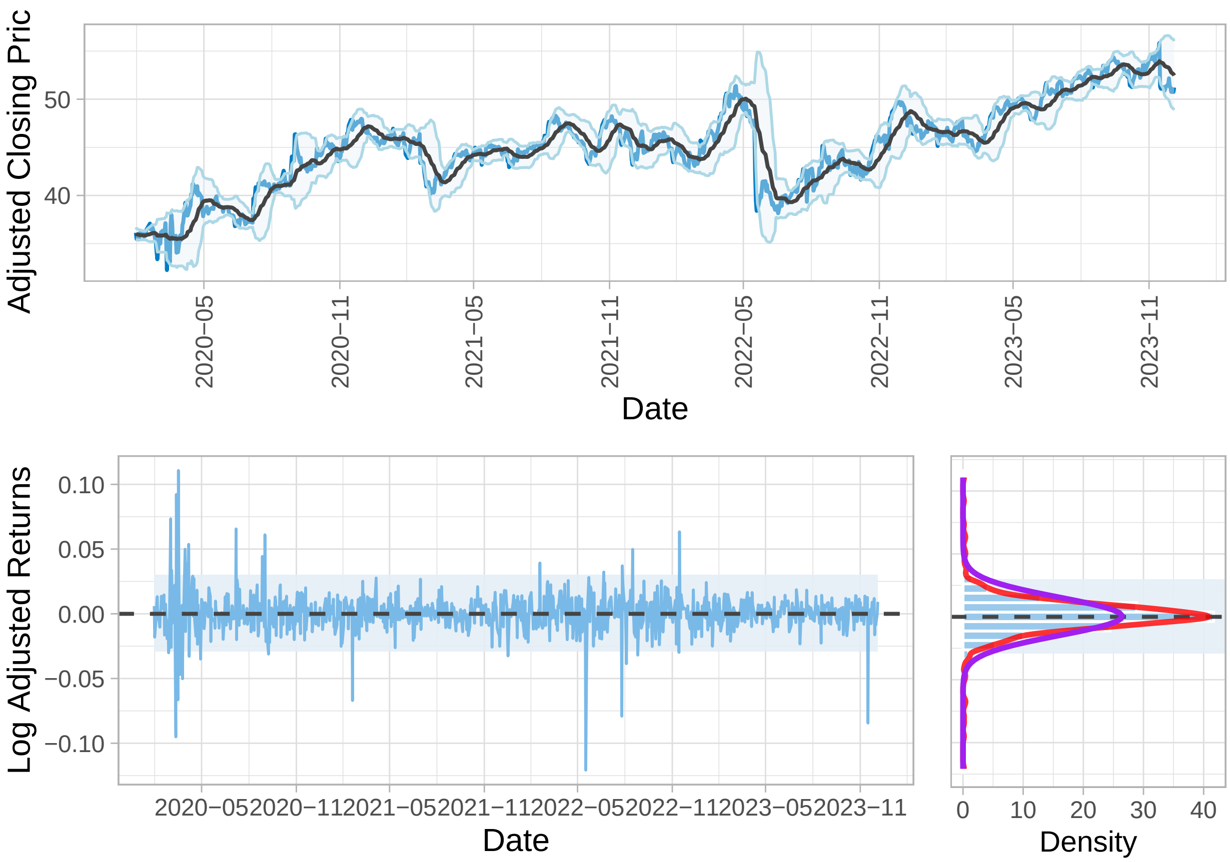
Let P_t be the price of an asset at time t, then the log returns is defined as:

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

```r
# 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) +
```

```r
  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
)
```



From the above figure, 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.

Fitting ARIMA Model

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

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

```
## Series: WMT_df$LogReturns
## ARIMA(0,0,1) with zero mean
##
## Coefficients:
##           ma1
##       -0.0666
## s.e.   0.0312
##
## sigma^2 = 0.0002266:  log likelihood = 2694.43
## AIC=-5384.85   AICc=-5384.84   BIC=-5375.1
##
## Training set error measures:
##                        ME       RMSE         MAE      MPE   MAPE      MASE
## Training set 0.0003912174 0.01504515 0.009658333 96.81537 110.64 0.6897203
##                      ACF1
## Training set -0.002686485
```

The fitted `ARIMA(0,0,1)` model for Walmart Inc.'s log returns is summarized above, 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 $\mathbb{E}[r_t] = 0$.

Fitting ARCH/GARCH Model

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)`, while generalizing conditional distributions `c("norm", "ged", "std", "snorm", "sged", "sstd")`.

`GARCH(1,0)`

```r
fit_arch <- garchFit(~ garch(1, 0), data = WMT_df$LogReturns, cond.dist = "std")
```

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

```
##  h.start:                   2
##  llh.start:                 1
##  Length of Series:          970
##  Recursion Init:            mci
##  Series Scale:              0.01508381
##
## Parameter Initialization:
##  Initial Parameters:          $params
##  Limits of Transformations:   $U, $V
##  Which Parameters are Fixed?  $includes
##  Parameter Matrix:
##                      U           V     params includes
##     mu     -0.24252318   0.2425232 0.02425232     TRUE
##     omega   0.00000100 100.0000000 0.10000000     TRUE
##     alpha1  0.00000001   1.0000000 0.10000000     TRUE
##     gamma1 -0.99999999   1.0000000 0.10000000    FALSE
##     delta   0.00000000   2.0000000 2.00000000    FALSE
##     skew    0.10000000  10.0000000 1.00000000    FALSE
##     shape   1.00000000  10.0000000 4.00000000     TRUE
##  Index List of Parameters to be Optimized:
##     mu  omega alpha1  shape
##      1      2      3      7
##  Persistence:                  0.1
##
##
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
##
##   0:     1578.9869: 0.0242523 0.100000 0.100000  4.00000
##   1:     1200.7500: 0.0242621  1.06188 0.373406  4.00573
##   2:     1186.3627: 0.0245112  1.10176 0.156741  3.55700
##   3:     1165.6168: 0.0246476 0.854538 0.136180  3.29130
##   4:     1162.0314: 0.0247733 0.615595 0.405005  3.23856
##   5:     1161.4280: 0.0280457 0.792418 0.411060  2.95106
##   6:     1161.1112: 0.0302257 0.734243 0.358167  3.29443
##   7:     1160.2829: 0.0369013 0.659299 0.271387  3.50241
##   8:     1160.2284: 0.0418007 0.665885 0.309612  3.42595
##   9:     1160.2137: 0.0406810 0.657399 0.300030  3.45263
##  10:     1160.2124: 0.0402499 0.661297 0.297821  3.44226
##  11:     1160.2122: 0.0403790 0.661346 0.298453  3.43859
##  12:     1160.2121: 0.0403767 0.661557 0.298849  3.43559
##  13:     1160.2121: 0.0403775 0.661566 0.298831  3.43560
##
## Final Estimate of the Negative LLH:
##  LLH:  -2908.097    norm LLH:  -2.998038
##           mu        omega       alpha1        shape
## 0.0006090464 0.0001505204 0.2988305600 3.4355988599
##
## R-optimhess Difference Approximated Hessian Matrix:
##                   mu         omega        alpha1         shape
## mu     -9.040665e+06    -1634075.8     371.64664     -71.10239
## omega  -1.634076e+06 -8604539379.5 -544036.38079 -404984.88704
```

```
## alpha1  3.716466e+02     -544036.4    -196.80960     -39.57583
## shape  -7.110239e+01     -404984.9     -39.57583     -26.75460
## attr(,"time")
## Time difference of 0.007131577 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.02201462 secs
```

```r
summary(fit_arch)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 0), data = WMT_df$LogReturns, cond.dist = "std")
##
## Mean and Variance Equation:
##  data ~ garch(1, 0)
## <environment: 0x6106c8013680>
##  [data = WMT_df$LogReturns]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##         mu       omega      alpha1       shape
## 0.00060905  0.00015052  0.29883056  3.43559886
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     6.090e-04   3.326e-04    1.831 0.067090 .
## omega  1.505e-04   2.019e-05    7.454 9.06e-14 ***
## alpha1 2.988e-01   8.543e-02    3.498 0.000469 ***
## shape  3.436e+00   3.925e-01    8.753  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  2908.097    normalized:  2.998038
##
## Description:
##  Fri Dec 13 16:18:31 2024 by user:
##
##
## Standardised Residuals Tests:
##                                   Statistic   p-Value
##  Jarque-Bera Test   R    Chi^2  8428.0097469 0.0000000
##  Shapiro-Wilk Test  R    W         0.8835441 0.0000000
```

```
##  Ljung-Box Test     R    Q(10)    10.6330501 0.3868118
##  Ljung-Box Test     R    Q(15)    14.9891482 0.4521992
##  Ljung-Box Test     R    Q(20)    17.3231831 0.6318993
##  Ljung-Box Test     R^2  Q(10)     6.0589807 0.8102838
##  Ljung-Box Test     R^2  Q(15)     9.2123320 0.8661560
##  Ljung-Box Test     R^2  Q(20)    10.1141536 0.9660545
##  LM Arch Test       R    TR^2      6.2949686 0.9004887
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -5.987829 -5.967716 -5.987863 -5.980173
```

GARCH(1,1)

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

```
##
## Series Initialization:
##  ARMA Model:                arma
##  Formula Mean:              ~ arma(0, 0)
##  GARCH Model:               garch
##  Formula Variance:          ~ garch(1, 1)
##  ARMA Order:                0 0
##  Max ARMA Order:            0
##  GARCH Order:               1 1
##  Max GARCH Order:           1
##  Maximum Order:             1
##  Conditional Dist:          std
##  h.start:                   2
##  llh.start:                 1
##  Length of Series:          970
##  Recursion Init:            mci
##  Series Scale:              0.01508381
##
## Parameter Initialization:
##  Initial Parameters:          $params
##  Limits of Transformations:   $U, $V
##  Which Parameters are Fixed?  $includes
##  Parameter Matrix:
##                     U           V     params includes
##     mu     -0.24252318   0.2425232 0.02425232     TRUE
##     omega   0.00000100 100.0000000 0.10000000     TRUE
##     alpha1  0.00000001   1.0000000 0.10000000     TRUE
##     gamma1 -0.99999999   1.0000000 0.10000000    FALSE
##     beta1   0.00000001   1.0000000 0.80000000     TRUE
##     delta   0.00000000   2.0000000 2.00000000    FALSE
##     skew    0.10000000  10.0000000 1.00000000    FALSE
##     shape   1.00000000  10.0000000 4.00000000     TRUE
##  Index List of Parameters to be Optimized:
##     mu  omega alpha1  beta1  shape
##      1      2      3      5      8
##  Persistence:                  0.9
##
##
```

```
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
##
##   0:     1144.0365: 0.0242523 0.100000 0.100000 0.800000  4.00000
##   1:     1140.8189: 0.0242527 0.0907284 0.0976184 0.793654  3.99981
##   2:     1139.9958: 0.0242539 0.0801877 0.100110 0.789829  3.99972
##   3:     1139.4862: 0.0242560 0.0801669 0.110852 0.793895  3.99989
##   4:     1139.1930: 0.0242596 0.0704237 0.116546 0.791757  3.99993
##   5:     1138.9625: 0.0242735 0.0694686 0.127116 0.796087  4.00041
##   6:     1138.9471: 0.0243531 0.0662979 0.134805 0.789117  4.00226
##   7:     1138.9036: 0.0244474 0.0722051 0.138167 0.780925  4.00413
##   8:     1138.8905: 0.0246555 0.0706010 0.133614 0.785691  4.00765
##   9:     1138.8901: 0.0246560 0.0707493 0.133643 0.785784  4.00766
##  10:     1138.8899: 0.0246617 0.0707840 0.133276 0.785777  4.00776
##  11:     1138.8895: 0.0246705 0.0708291 0.133249 0.785972  4.00792
##  12:     1138.8814: 0.0251670 0.0671673 0.129505 0.793530  4.01699
##  13:     1138.8050: 0.0315006 0.0681307 0.132208 0.786055  4.12987
##  14:     1138.7985: 0.0372908 0.0698496 0.131679 0.789030  3.97496
##  15:     1138.7771: 0.0346322 0.0686342 0.127097 0.791620  4.07268
##  16:     1138.7760: 0.0343408 0.0685777 0.130443 0.789571  4.06513
##  17:     1138.7751: 0.0346158 0.0685543 0.129147 0.790478  4.06150
##  18:     1138.7751: 0.0346092 0.0685688 0.129127 0.790487  4.06182
##  19:     1138.7751: 0.0346079 0.0685661 0.129128 0.790485  4.06182
##
## Final Estimate of the Negative LLH:
##  LLH:  -2929.534    norm LLH:  -3.020138
##           mu        omega       alpha1        beta1        shape
## 5.220192e-04 1.560025e-05 1.291283e-01 7.904854e-01 4.061824e+00
##
## R-optimhess Difference Approximated Hessian Matrix:
##                  mu         omega        alpha1         beta1         shape
## mu     -9605288.786       2119893 -3.380175e+03  1.421115e+03 -1.798020e+02
## omega   2119893.004 -347391089679 -2.992451e+07 -4.841374e+07 -1.371940e+06
## alpha1    -3380.175     -29924505 -4.599349e+03 -5.342390e+03 -1.564956e+02
## beta1      1421.115     -48413741 -5.342390e+03 -7.640047e+03 -2.119294e+02
## shape      -179.802      -1371940 -1.564956e+02 -2.119294e+02 -1.006576e+01
## attr(,"time")
## Time difference of 0.01111698 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.03285933 secs
```

```
summary(fit_garch)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 1), data = WMT_df$LogReturns, cond.dist = "std")
```

```
##
## Mean and Variance Equation:
##  data ~ garch(1, 1)
## <environment: 0x6106c638ef00>
##  [data = WMT_df$LogReturns]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##         mu       omega      alpha1       beta1       shape
## 0.00052202  0.00001560  0.12912833  0.79048538  4.06182415
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     5.220e-04   3.236e-04    1.613  0.10672
## omega  1.560e-05   6.808e-06    2.292  0.02193 *
## alpha1 1.291e-01   4.705e-02    2.744  0.00606 **
## beta1  7.905e-01   6.870e-02   11.507  < 2e-16 ***
## shape  4.062e+00   5.057e-01    8.032 8.88e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  2929.534    normalized:  3.020138
##
## Description:
##  Fri Dec 13 16:18:31 2024 by user:
##
##
## Standardised Residuals Tests:
##                                   Statistic   p-Value
##  Jarque-Bera Test   R    Chi^2  1.332671e+04 0.0000000
##  Shapiro-Wilk Test  R    W      8.693276e-01 0.0000000
##  Ljung-Box Test     R    Q(10)  6.444710e+00 0.7766203
##  Ljung-Box Test     R    Q(15)  7.411283e+00 0.9452166
##  Ljung-Box Test     R    Q(20)  1.119994e+01 0.9408718
##  Ljung-Box Test     R^2  Q(10)  1.153811e+00 0.9996696
##  Ljung-Box Test     R^2  Q(15)  1.830576e+00 0.9999836
##  Ljung-Box Test     R^2  Q(20)  2.085511e+00 0.9999998
##  LM Arch Test       R    TR^2   1.372330e+00 0.9999192
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -6.029967 -6.004826 -6.030020 -6.020398
```

The summaries above for 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 $\epsilon_t \sim \mathrm{std}(0,1)$ 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.

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.

```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
}
```

```
##
```

```
## Iteration: 1
## Number of influential points detected: 12
## Removed influential points at indices: 22 30 110 147 150 266 519 580 627 642 706 958
##
## Iteration: 2
## Number of influential points detected: 4
## Removed influential points at indices: 29 449 573 619
##
## Iteration: 3
## Number of influential points detected: 3
## Removed influential points at indices: 29 30 621
##
## Iteration: 4
## Number of influential points detected: 1
## Removed influential points at indices: 633
##
## Iteration: 5
## Number of influential points detected: 0
## No more influential points. Stopping iteration.
```

```r
# 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))
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(0, 0, 1) + garch(1, 1), data = data_cleaned,
##     cond.dist = "std", trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(0, 0, 1) + garch(1, 1)
## <environment: 0x6106c47d2548>
##  [data = data_cleaned]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##         mu       omega      alpha1       beta1       shape
## 4.9198e-04  6.3505e-06  1.0224e-01  8.5001e-01  1.0000e+01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     4.920e-04   3.213e-04    1.531  0.12572
```

```
## omega  6.350e-06   3.262e-06    1.947  0.05158 .
## alpha1 1.022e-01   3.102e-02    3.296  0.00098 ***
## beta1  8.500e-01   4.877e-02   17.430  < 2e-16 ***
## shape  1.000e+01   2.507e+00    3.988 6.66e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  2972.232    normalized:  3.128665
##
## Description:
##  Fri Dec 13 16:18:32 2024 by user:
##
##
## Standardised Residuals Tests:
##                                  Statistic    p-Value
##  Jarque-Bera Test   R    Chi^2   3.355696 0.18677549
##  Shapiro-Wilk Test  R    W       0.996212 0.02083633
##  Ljung-Box Test     R    Q(10)   4.340417 0.93068634
##  Ljung-Box Test     R    Q(15)   8.318518 0.91037890
##  Ljung-Box Test     R    Q(20)  17.592676 0.61422106
##  Ljung-Box Test     R^2  Q(10)   5.779881 0.83340515
##  Ljung-Box Test     R^2  Q(15)   8.138288 0.91811888
##  Ljung-Box Test     R^2  Q(20)  12.309297 0.90501891
##  LM Arch Test       R    TR^2    5.568408 0.93625874
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -6.246803 -6.221243 -6.246858 -6.237065
##
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(0, 0, 1) + garch(1, 1), data = data_cleaned,
##     cond.dist = "std", trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(0, 0, 1) + garch(1, 1)
## <environment: 0x6106c47d2548>
##  [data = data_cleaned]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##         mu       omega      alpha1       beta1       shape
## 4.9198e-04  6.3505e-06  1.0224e-01  8.5001e-01  1.0000e+01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
```

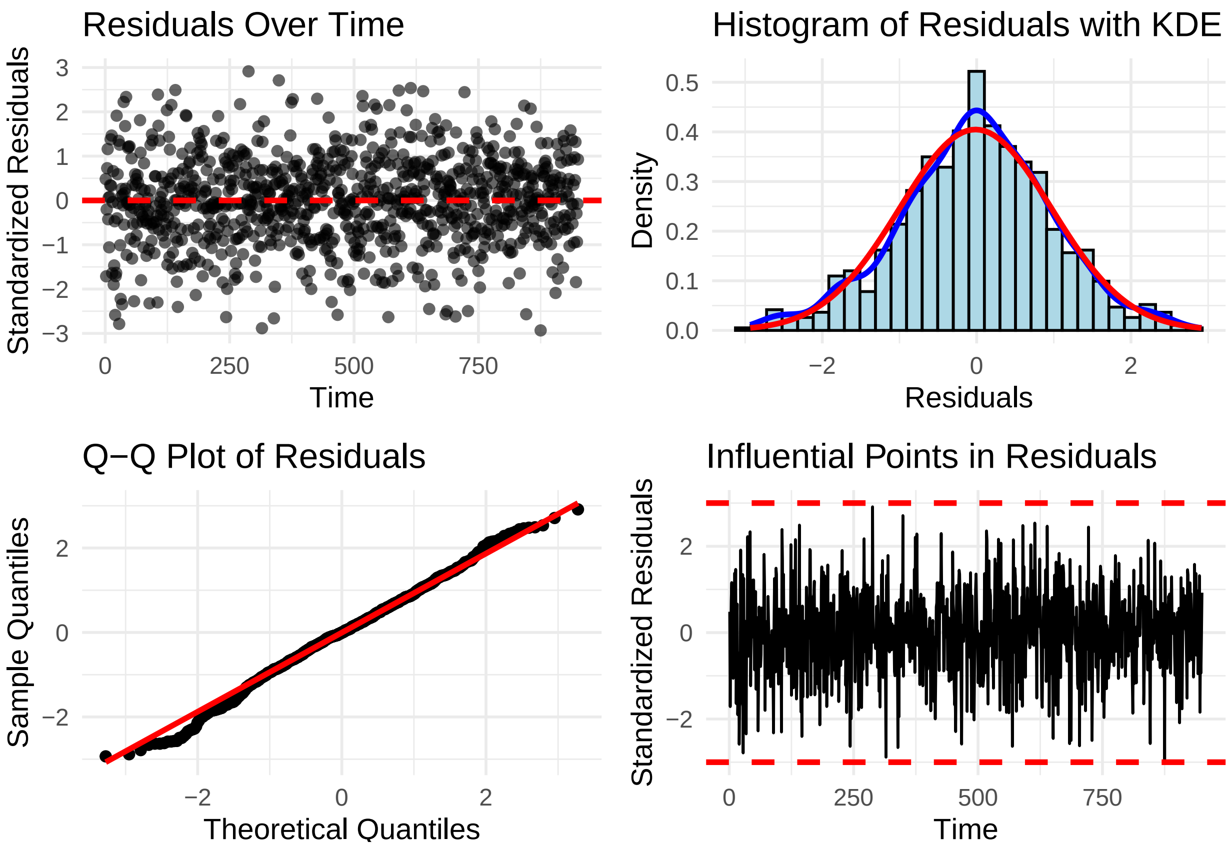
```
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     4.920e-04   3.213e-04    1.531  0.12572
## omega  6.350e-06   3.262e-06    1.947  0.05158 .
## alpha1 1.022e-01   3.102e-02    3.296  0.00098 ***
## beta1  8.500e-01   4.877e-02   17.430  < 2e-16 ***
## shape  1.000e+01   2.507e+00    3.988 6.66e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  2972.232    normalized:  3.128665
##
## Description:
##  Fri Dec 13 16:18:32 2024 by user:
##
##
## Standardised Residuals Tests:
##                                   Statistic    p-Value
##  Jarque-Bera Test   R    Chi^2   3.355696 0.18677549
##  Shapiro-Wilk Test  R    W       0.996212 0.02083633
##  Ljung-Box Test     R    Q(10)   4.340417 0.93068634
##  Ljung-Box Test     R    Q(15)   8.318518 0.91037890
##  Ljung-Box Test     R    Q(20)  17.592676 0.61422106
##  Ljung-Box Test     R^2  Q(10)   5.779881 0.83340515
##  Ljung-Box Test     R^2  Q(15)   8.138288 0.91811888
##  Ljung-Box Test     R^2  Q(20)  12.309297 0.90501891
##  LM Arch Test       R    TR^2    5.568408 0.93625874
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -6.246803 -6.221243 -6.246858 -6.237065
```

From the above summary, 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, the Standardized Residual tests also indicate a better fit. 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.

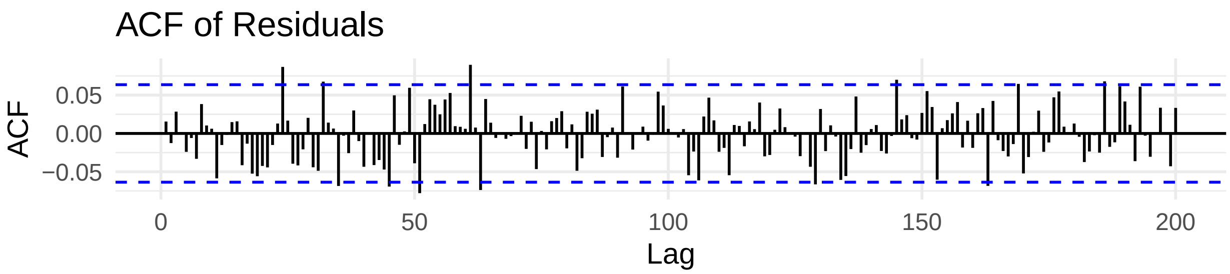
Residual Diagnostics

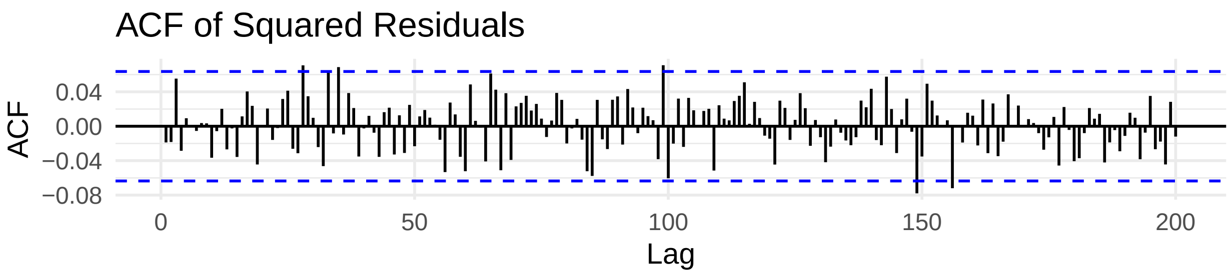
```
# Extract standardized residuals for the cleaned model
residuals_cleaned <- residuals(arma_garch_model, standardize=TRUE)

# 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
```

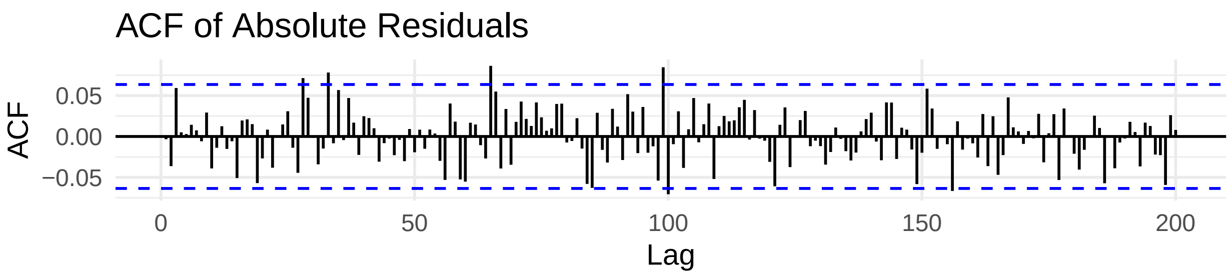
```r
p2 <- ggplot(data.frame(Residuals = residuals_cleaned), aes(x = Residuals)) +
  geom_histogram(aes(y = after_stat(density)),
                 bins=30, fill="lightblue", color="black") +
  geom_density(color = "blue", linetype = "solid", linewidth=1) +
  stat_function(fun=dnorm, args=list(mean=mean(residuals_cleaned),
                                     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)
```







The above figures 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.

10-Day Forecast

```r
# 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))
```

```r
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)
```

```
## [1] "WMT"
```

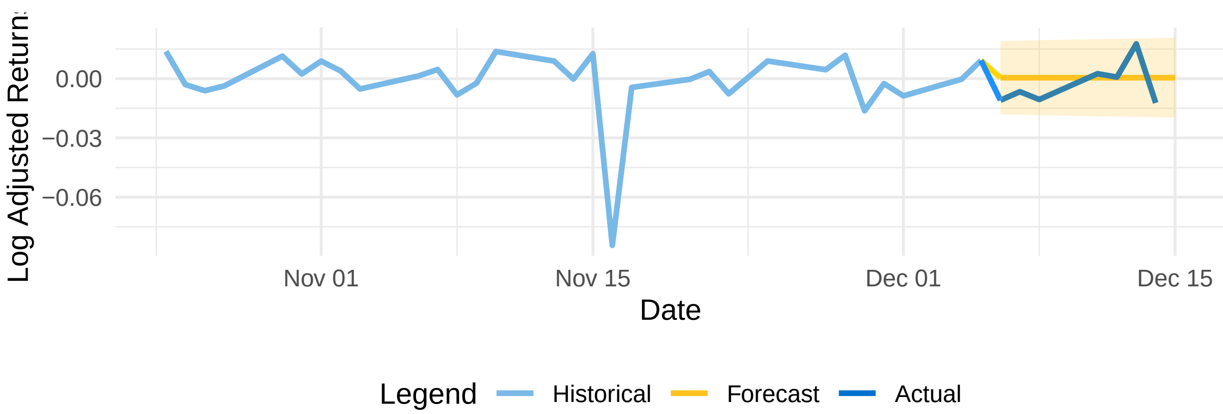
```r
# Prepare actual prices dataframe
actual_prices <- data.frame(
  Date = index(WMT),
  AdjClose = as.numeric(Ad(WMT))
)
actual_prices$LogReturns <- c(NA, diff(log(actual_prices$AdjClose)))
actual_prices <- na.omit(actual_prices)

# 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,
```

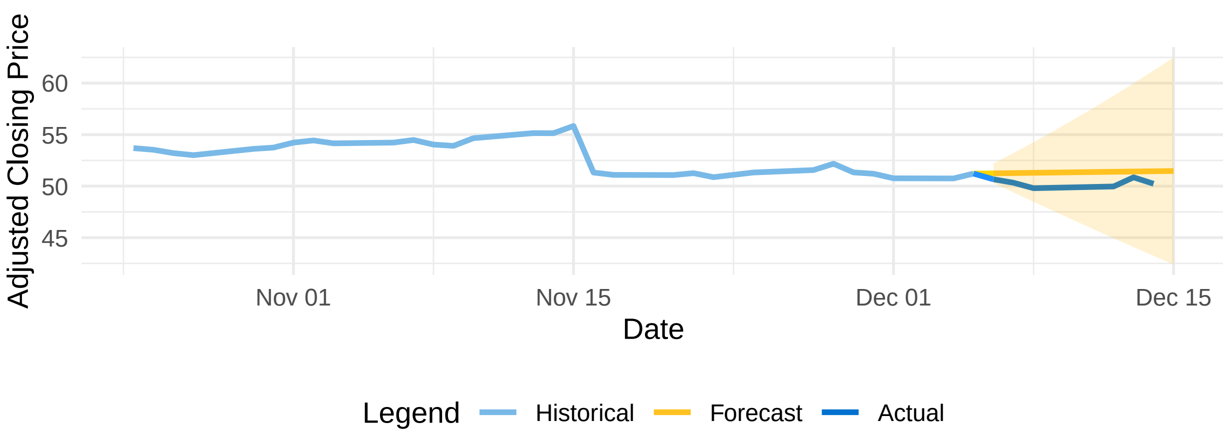
```r
            aes(x = Date, y = AdjClose, color = "Historical"), size = 1) +
  # Forecast price line
  geom_line(data = forecast_df,
            aes(x = Date, y = AdjPrice, color = "Forecast"), size = 1) +
  # Actual market prices line
  geom_line(data = actual_prices,
            aes(x = Date, y = 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",
  )
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
# 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
cat("RMSE for Log Returns: ", log_returns_rmse, "\n")
```

```
## RMSE for Log Returns:  0.01043903
```

```r
# 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")
```

```
## RMSE for Adjusted Prices:  1.156709
```

The above figure 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 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.

References

Bollerslev, T. (1986). Generalized Autoregressive Conditional Heteroskedasticity. Journal of Econometrics, 31(3), 307–327.

Box G.E.P., Jenkins., G.M., & Reinsel, G.C. (1994). Time Series Analysis: Forecasting and Control.

Campbell, J.Y., Lo, A.W., & MacKinlay, A.C. (1997). The Econometrics of Financial Markets.

Diebold, F.X. (2001). Elements of Forecasting.

Engle, R.F. (1982). Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of United Kingdom Inflation. Econometrica, 50(4), 987-1007.

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.

Hamilton, J.D. (1994). Time Series Analysis.

Hyndman, R.J., & Athanasopoulos, G. (2021). Forecasting: Principles and Practice.

Hull, J.C. (2021). Options, Futures, and Other Derivatives.

Pfaff, B. (2008). Analysis of Integrated and Cointegrated Time Series with R.

Poon, S.H., & Granger, C.W.J. (2003). Forecasting Volatility in Financial Markets: A Review. Journal of Economic Literature, 41(2), 478–539.

Tsay, R.S. (2010). Analysis of Financial Time Series.

Zivot, E., & Wang, J. (2006). Modeling Financial Time Series with S-PLUS.