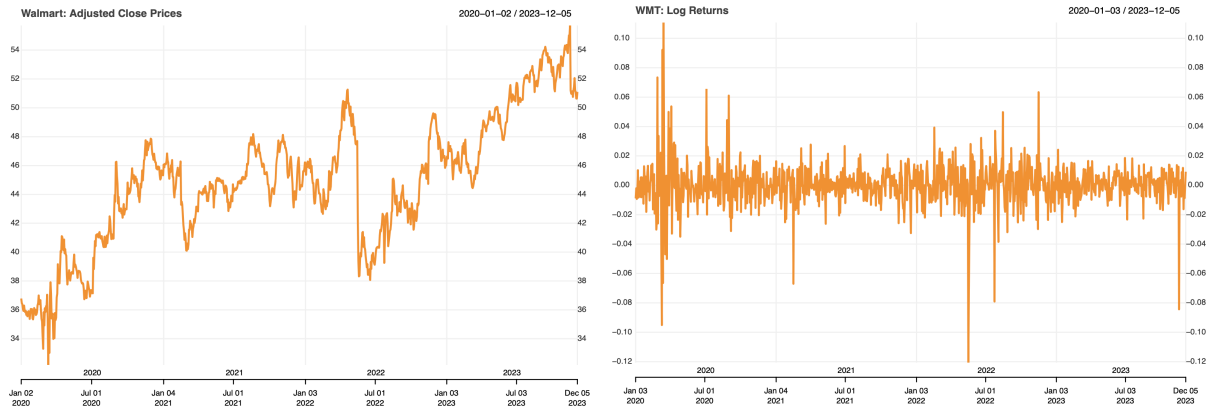
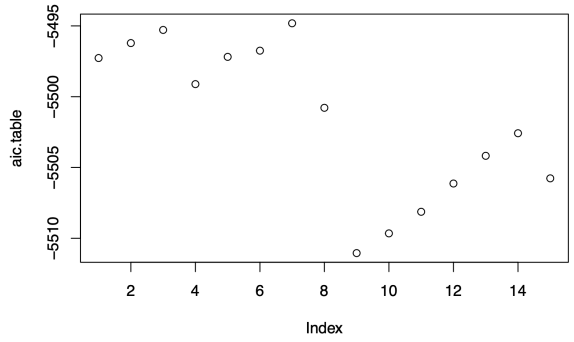
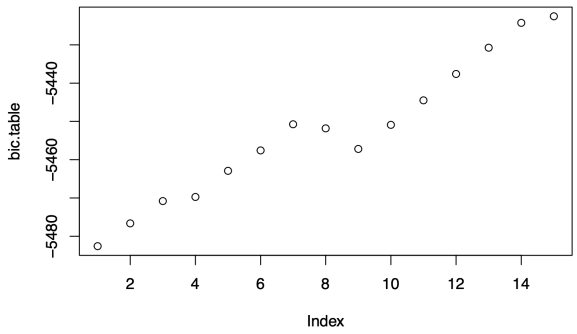
The primary objective of this paper is to analyze Walmart's (WMT) stock price data and its volatility dynamics from January 2020 to December 2023, in order to create a 10 step ahead forecast for the daily log returns. The analysis began with transforming the data and testing for stationarity to ensure that appropriate models can be applied. After confirming that the log returns are stationary, three GARCH(1,1) models with different parameters were fit to capture volatility clustering and non-constant variance in the returns. They were evaluated based on the AIC and BIC, with the best model being used to create the forecast.

To start the process, the time series was examined to check for stationarity. Obviously, the stock price alone is highly unlikely to be stationary, and this was tested through the use of an Ljung-Box test, an ADF test, and a KPSS test. All three confirmed the hypothesis that log differencing was required, so it was performed, and then the log returns were tested again. The same tests were performed, which now showed that the data was stationary, which allowed the modeling process to proceed. Below are the charts of the Walmart stock price time series and the log return time series for reference.

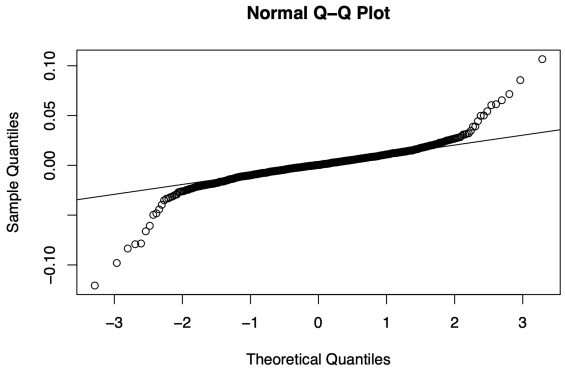


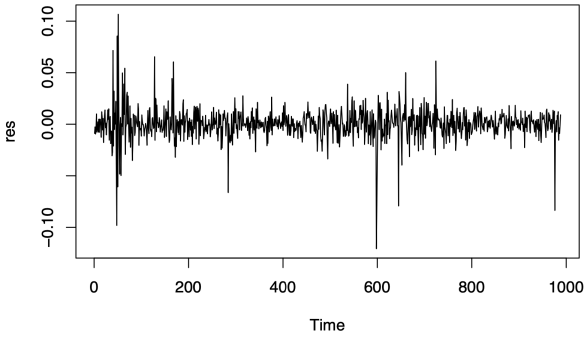
The next step was to attempt to determine whether the log returns had either moving average or autoregressive components. To test for the autoregressive component, the ACF and PACF graphs were computed, which showed essentially no identifiable autoregressive relationship. Next, to examine possible moving average relationships, MA(q) models of order one through 15 were fit, and then the AIC and BIC were computed for each, to find the minimum values. The minimum AIC value was found at the MA(9) model, while the minimum BIC value was found at the MA(1) model. As it is known that BIC tends to penalize larger models more than AIC, this finding aligns with expectations. The graphs showing the progression of AIC and BIC values are included below.





At this point, the ARIMA(0,0,1) model, which is equivalent to the MA(1) process, was identified as a baseline model for the log returns. However, upon examination of the residuals from the fitted model, it was obvious that this model was lacking in various ways. First of all, the Q-Q plot showed clear fat tails, which is common in asset returns. Next, simple ARIMA or MA models fail to account for one key feature observed in the log returns: volatility clustering. This phenomenon, where large changes in returns tend to cluster together, suggests that the variance of the series is not constant over time. This can easily be observed in the log returns chart above, where we see clusters of high returns around February 2020. Furthermore, the plotted residuals from the MA(1) model showed clear heteroskedasticity. To capture this behavior, the analysis proceeded with the implementation of GARCH models, one of which included an MA(1) term.

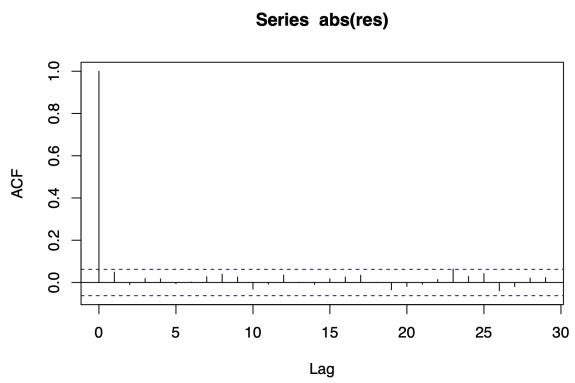


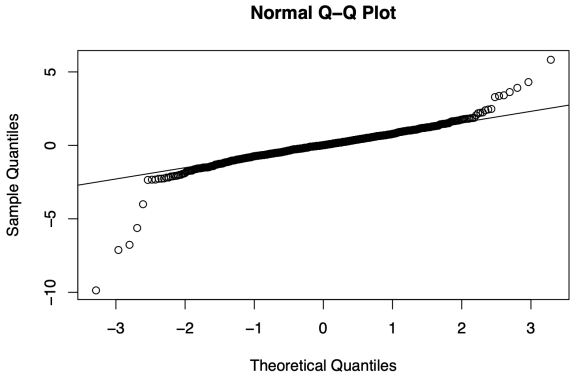


The first step of the GARCH modeling was to examine the relationship that was discovered in the volatility. First, the absolute log returns were computed. As stated before, the ACF and PACF of the log returns showed no autoregressive relationship, but upon examining the ACF of the absolute log returns, it was clear there was a significant autoregressive relationship. This is where the main idea behind GARCH comes in. Log returns may be serially uncorrelated, but the size of the log returns are serially correlated.

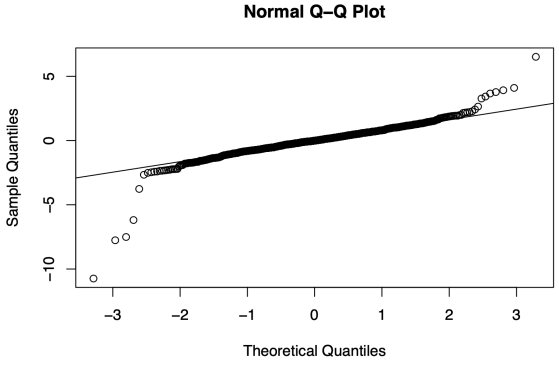
This relationship was also observed by looking at the squared log returns, and confirmed by performing Ljung-Box tests, which showed that the absolute log returns are not white noise.

The first GARCH model fit was a GARCH(1,1) + ARMA(0,1). This resulted in an AIC of -5.772179 and a BIC of -5.747403. That model was then compared to a regular GARCH(1,1). The regular GARCH(1,1) posted an AIC of -5.772053 and a BIC of -5.752233. Due to the similar performance of the models, the ARMA(0,1) was dropped for simplicity. Upon examining the standardized residuals to confirm the fit of the model, the absolute and squared residuals were found to be uncorrelated, which was the goal of incorporating the GARCH. However, the Q-Q plot still showed some fat tails, which indicated that a non-normal distribution must be used. A Kolmogorov-Smirnov test confirmed that the underlying distribution was not normal.





Calculating excess kurtosis gave a value of 13.91941, which confirms significantly fat tails. To address these observed fat tails in the standardized residuals, the conditional distribution for the GARCH(1,1) model was modified to the Student's t Distribution. The GARCH(1,1) model with a Student's t distribution was fit, and the results showed a significant improvement in both AIC (-6.042258) and BIC (-6.017482) values, as well as a better fit in the Q-Q plot.



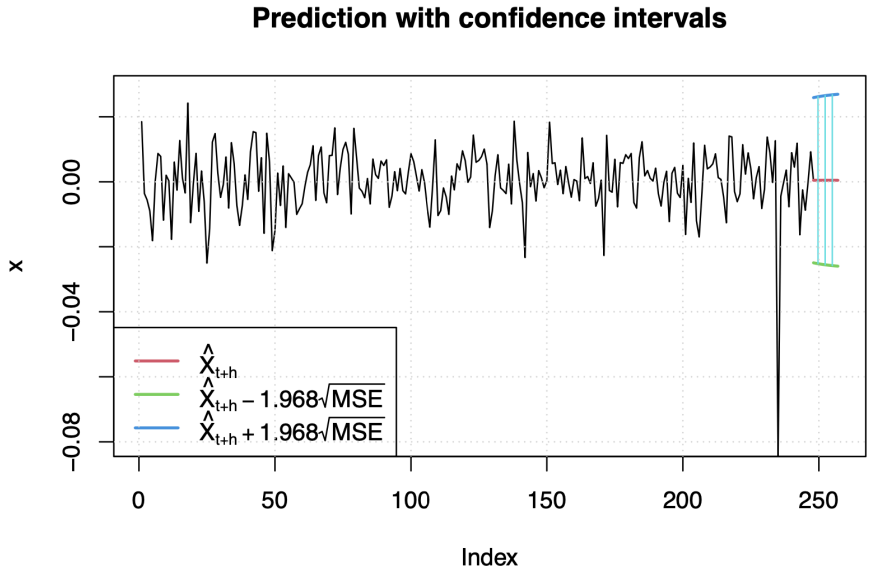
Finally, the last thing to check was the skew of the distribution. The calculated skew value was -0.4016063, which led to creating a skewed Student's t Distribution GARCH(1,1). However, the AIC and BIC of this model were actually worse than the model without the skewed distribution. Because of the results, the GARCH(1,1) with the Student's t Distribution and no skew term was selected as the best model for the predictions. A summary of all of the models is below.

Model:	AIC:	BIC:
GARCH(1,1)	-5.772055	-5.752234
GARCH(1,1) STD	-6.042259	-6.017483
GARCH(1,1) STD, Skew	-6.040979	-6.011248

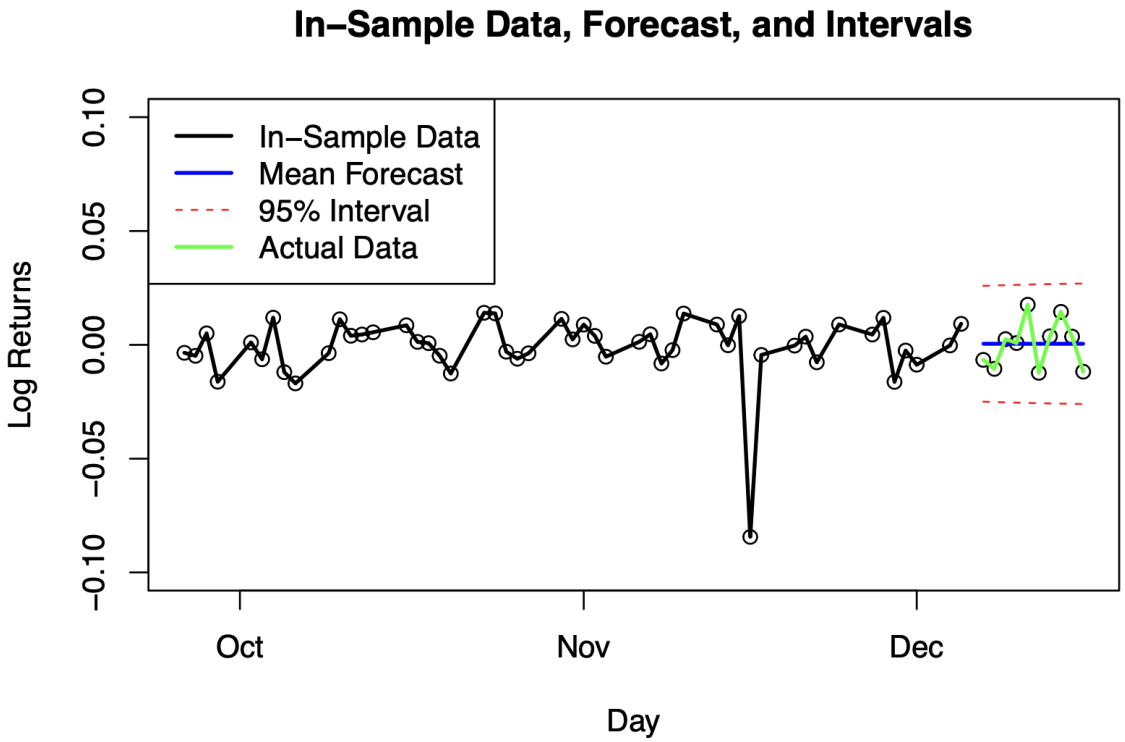
The details of the GARCH(1,1) with the Student's t Distribution are also below:

Mu	Omega	alpha1	beta1	shape
4.7597e-04	1.6354e-05	1.3325e-01	7.8155e-01	4.0942e+00

Finally, the predictions. Since all models were GARCH models, showing all 3 forecasts seems redundant as they look very similar. However, all code used for this paper is attached below, and there are forecast graphs for each model. The forecast chart displayed below is the one predicted by the highlighted model above.



As it turned out, the 10 out-of-sample days all fell within the forecast interval. The selected model was accurate and a good forecaster for the Walmart log returns.



This paper analyzed Walmart's stock price and log returns from January 2020 to December 2023, focusing on daily log returns. Stationarity was achieved through log differencing, and ARIMA models were initially tested but failed to capture volatility clustering. To address this, GARCH(1,1) models were applied with different conditional distributions. The GARCH(1,1) with a Student's t distribution was selected as the best model, as it effectively captured the fat tails observed in the data, confirmed by lower AIC and BIC values. The 10-step-ahead forecast generated by the model successfully contained all actual out-of-sample values within the forecast intervals, demonstrating its accuracy and reliability in predicting Walmart's log returns.

Appendix (Code):

```
---
output:
  pdf_document: default
  html_document: default
---

```{r include=FALSE}
rm(list = ls())

# install and load packages
if (!require("quantmod")) { # quantmod for downloading online financial data
  install.packages("quantmod") # install it only once
  stopifnot(require("quantmod")) # load it every time
}

if (!require("fBasics")) { # fBasics for STAT461 calculations
  install.packages("fBasics") # install it only once
  stopifnot(require("fBasics")) # load it every time
}

if (!require("tseries")) { # tseries for time series analysis
  install.packages("tseries") # install it only once
  stopifnot(require("tseries")) # load it every time
}

if (!require("forecast")) { # forecast for time series analysis
  install.packages("forecast") # install it only once
  stopifnot(require("forecast")) # load it every time
}

if (!require("fGarch")) { # fGarch for GARCH
  install.packages("fGarch") # install it only once
  stopifnot(require("fGarch")) # load it every time
}
```

```{r}
getSymbols("WMT", from = "2020-01-01", to = "2023-12-06")
adj.close = Ad(WMT) # Adjusted Close Prices
chart_Series(adj.close, name = "Walmart: Adjusted Close Prices")
```

```{r}
acf(adj.close) #Highly autocorrelated
pacf(adj.close)
Box.test(adj.close, type = "Ljung") #P-value < 0.05 --> not white noise
adf.test(adj.close) #P-value > 0.05 --> non-stationary
kpss.test(adj.close) #P-value < 0.05 --> non-stationary
#All results expected due to the obvious autocorrelation of stock prices
```

```{r}
log.ret = diff(log(adj.close))[-1]
chart_Series(log.ret, name = "WMT: Log Returns")
```
```

```

```{r}
acf(log.ret) # kind of white noise
pacf(log.ret)
Box.test(log.ret, type = "Ljung") # Box-Ljung test: p-value < 0.05 --> not white noise
adf.test(log.ret) # ADF test: stationary
kpss.test(log.ret) # KPSS test: stationary
```

```{r}
aic.order.MA = function(ts, order.max = 15, plot = FALSE){ # find the order of MA model with the minimal AIC
  aic.table = numeric(order.max)
  for (i in 1:order.max) {
    model = arima(ts, order = c(0,0,i))
    aic.table[i] = AIC(model)
  }
  if (plot) plot(aic.table)
  return(which.min(aic.table))
}

bic.order.MA = function(ts, order.max = 15, plot = FALSE){ # find the order of MA model with the minimal BIC
  bic.table = numeric(order.max)
  for (i in 1:order.max) {
    model = arima(ts, order = c(0,0,i))
    bic.table[i] = BIC(model)
  }
  if (plot) plot(bic.table)
  return(which.min(bic.table))
}
```

```{r}
aic.order.MA(log.ret, plot = TRUE) #Selects MA(9)
bic.order.MA(log.ret, plot = TRUE) #Selects MA(1)
```

```{r}
#ARIMA(p, d, q)
model = auto.arima(log.ret)
model
# p = 0, d = 0, q = 1
```

```{r}
(pred = forecast(model, h = 10, level = c(95))) # predict 10 more values with CI
plot(pred)
```

```{r}
plot(model) # unit root test, omega = 1/x strictly inside unit circle, not x
res = resid(model)
plot(res) # Not equal var, i.e., homoscedasticity, suggests GARCH may be better
acf(res) # white noise
Box.test(res, type = "Ljung") # white noise
qqnorm(res); qqline(res) #Appears to have potentially have fat tails
```
```

```

```{r}
log.ret.abs = abs(log.ret)
acf(log.ret.abs) # not like white noise
Box.test(log.ret.abs, type = "Ljung") # Box-Ljung test: p-value < 0.05 => conclusion: not white noise, still dependent

log.ret.sqr = log.ret^2
acf(log.ret.sqr) # not like white noise
Box.test(log.ret.sqr, type = "Ljung") # Box-Ljung test: p-value < 0.05 => conclusion: not white noise, still dependent
```

```{r}
model = garchFit(formula = ~ arma(0, 1) + garch(1, 1), data = log.ret, trace = FALSE)
model # mu: drift, close to zero, the true value, MA(1) not significant, maybe use without
model.sum = summary(model) # Standardised Residuals Tests: most of p-values > 0.05
model.sum$ics["AIC"] # AIC
model.sum$ics["BIC"] # BIC
```

```{r}
# verify the model by checking the standardized residuals
res = residuals(model, standardize = T)
plot(res)
acf(res)
acf(res^2)
acf(abs(res)) # all look good => strong white noise
qqnorm(res); qqline(res) # normality test: not normal => maybe a different cond.dist is better
ks.test(res, "pnorm") # normality test by Kolmogorov-Smirnov (KS) test: p-value < 0.05 => not normal
```

```{r}
predict(model, n.ahead = 10, plot = TRUE)
```

```{r}
model = garchFit(formula = ~ garch(1, 1), data = log.ret, trace = FALSE)
model # mu: drift, close to zero, the true value
model.sum = summary(model) # Standardised Residuals Tests: most of p-values > 0.05
model.sum$ics["AIC"] # AIC
model.sum$ics["BIC"] # BIC # almost the same AIC and BIC, BIC slightly lower than the MA + GARCH
```

```{r}
# verify the model by checking the standardized residuals
res = residuals(model, standardize = T)
acf(res)
acf(res^2)
acf(abs(res)) # all look good => strong white noise
qqnorm(res); qqline(res) # normality test: not normal => maybe a different cond.dist is better
ks.test(res, "pnorm") # normality test by Kolmogorov-Smirnov (KS) test: p-value < 0.05 => not normal
```

```{r}
predict(model, n.ahead = 10, plot = TRUE)
```
```

```

```{r}
kurtosis(log.ret) #13.91941, extremely high so extremely fat tailed
```

```{r}
model = garchFit(formula = ~ garch(1, 1), data = log.ret, trace = FALSE, cond.dist = "std") # standardized student's t dist = std, shape parameter is significant
model # mu: drift, close to zero, the true value; shape = degrees of freedom
model.sum = summary(model)
model.sum$ics["AIC"] # AIC
model.sum$ics["BIC"] # BIC # even smaller AIC and BIC, much better
```

```{r}
# verify the model by checking the standardized residuals
res = residuals(model, standardize = T)
acf(res)
acf(res^2)
acf(abs(res)) # all look good => strong white noise
qqnorm(res); qqline(res)

x = rstd(1000, mean = 0, sd = 1, nu = 4.0942) # make a standardized student t distribution for comparison
ks.test(res, x) # compare residual's distribution with x's by Kolmogorov-Smirnov (KS) test: p-value > 0.05 => the same
# the result matches the original cond.dist setup
```

```{r}
predict(model, n.ahead = 10, plot = TRUE)
```

```{r}
skewness(log.ret)
```

```{r}
model = garchFit(formula = ~ garch(1, 1), data = log.ret, trace = FALSE, cond.dist = "sstd") # standardized student's t dist = std
model # mu: drift, close to zero, the true value; shape = degrees of freedom; skew significant

model.sum = summary(model)
model.sum$ics["AIC"] # AIC
model.sum$ics["BIC"] # BIC # worse BIC and AIC than the GARCH without skew

# verify the model by checking the standardized residuals
res = residuals(model, standardize = T)
acf(res)
acf(res^2)
acf(abs(res)) # all look good => strong white noise
qqnorm(res); qqline(res)
```

```{r}

x = rsstd(1000, mean = 0, sd = 1, nu = 3.4952, xi = 0.96) # make a standardized skew student t distribution for comparison
ks.test(res, x)
```

```{r}
predict(model, n.ahead = 10, plot = TRUE)
```

```{r}
model = garchFit(formula = ~ garch(1, 1), data = log.ret, trace = FALSE, cond.dist = "std")
predict(model, n.ahead = 10, plot = TRUE)$lowerInterval
```

```{r}
getSymbols("WMT", from = "2023-12-07", to = "2023-12-22")
oos_adj.close = Ad(WMT) # Adjusted Close Prices
chart_Series(oos_adj.close, name = "Walmart: Adjusted Close Prices")
```

```{r}
oos_log.ret = diff(log(adj.close))[-1]
chart_Series(oos_log.ret, name = "WMT: Log Returns")
```

```{r}
# Example in-sample data (replace with your actual in-sample values)
in_sample_days <- index(log.ret[939: 988])  # Assuming 100 in-sample days
in_sample_values <- log.ret[939: 988]  # Replace with actual in-sample log returns
in_sample_values <- as.numeric(in_sample_values)

# Out-of-sample data
forecast_days <- as.Date("2023-12-07") + 0:9
forecast_data <- data.frame(
  Day = forecast_days,
  Mean_Forecast = rep(0.0004759675, 10),
  Lower_Interval = c(-0.02495163, -0.02511310, -0.02525993, -0.02539352, -0.02551512,
                     -0.02562587, -0.02572677, -0.02581873, -0.02590258, -0.02597905),
  Upper_Interval = c(0.02590355, 0.02606502, 0.02621185, 0.02634544, 0.02646704,
                     0.02657779, 0.02667869, 0.02677065, 0.02685450, 0.02693097),
  Actual = c(-0.0066028540, -0.0105499970, 0.0025157535, 0.0007931595, 0.0176159122,
             -0.0122788198, 0.0037388557, 0.0144943896, 0.0036071662, -0.0117708982)
)

# Create an empty plot
plot(c(in_sample_days, forecast_data$Day),
     c(in_sample_values, forecast_data$Actual),
     ylim = range(-.1,.1),
     xlab = "Day", ylab = "Log Returns",
     main = "In-Sample Data, Forecast, and Intervals")

# Plot in-sample data
lines(in_sample_days, in_sample_values, col = "black", lwd = 2, lty = 1)

# Plot mean forecast

```
lines(forecast_data$Day, forecast_data$Mean_Forecast, col = "blue", lwd = 2, lty = 1)

# Plot forecast intervals
lines(forecast_data$Day, forecast_data$Lower_Interval, col = "red", lty = 2)
lines(forecast_data$Day, forecast_data$Upper_Interval, col = "red", lty = 2)

# Plot actual out-of-sample values
lines(forecast_data$Day, forecast_data$Actual, col = "green", lwd = 2, lty = 1)

# Add a legend
legend("topleft", legend = c("In-Sample Data", "Mean Forecast", "95% Interval", "Actual Data"),
       col = c("black", "blue", "red", "green"), lty = c(1, 1, 2, 1), lwd = c(2, 2, 1, 2))
```