Analysis of Market Volatility

Ariel Khait

Fall 2024

Table of Contents

Analysis of VIX and VOX Indexes

What is Implied Volatility

Implied Volatility is the market's forecast of the underlying equity or index's future volatility, derived from the prices of options contracts that expire at a future date. Using the Black-Sholes model to price options, an options price depends on its strike price, time to maturity, underlying asset price, risk free rate, and volatility. This is considered arbitrage free pricing since it is a risk neutral way to price options. To calculate the value of a European option, all components of Black-Sholes model are known to the trader today with the expectation of volatility. With this in mind, we can use market prices of options in combination with the other components of the Black-Sholes model to determine an implied volatility.

The VIX is the 30-day implied volatility of the S&P500 index created by the Chicago Board Options Exchange (CBOE). Calculated using out-of-the-money, and in-the-money options with differing strike prices and maturities, the VIX is representative of the markets outlook on future volatility of the S&P 500. This is often known as the “fear” index since times of high volatility are known for their fear of market collapse meaning a high VIX value indicated to trader a large amount of “fear” in the market caused by some type of shock.

Its precursor, the VXO [also calculated by the CBOE] was calculated using at-the-money options on the S&P100. In 2003, CBOE switched over their calculation from strictly at-the-money options of the top 100 US companies [S&P100] to both out-of-the-money options and in-the-money options on the top 500 US companies [S&P500].

Evolution of VIX vs. VXO

Figure 1 is the VIX index from 2004 until the present day and Figure 2 is the VOX calculated from 2004 until CBOE stop calculating the index in September of 2021. We first observe that Figures 1 and 2 are very similar, in shape and structure as they are both calculated on options where the underlying assets are high market cap companies that usually behave similarly. It is important to note that there are large spikes in 2008, and 2020 corresponding to the housing bubble and the COVID-19 crisis respectively. These were times of large uncertainty in the economy and high volatility as seen by both indexes.

The only noticable differences we see between the two figures are the noticably consistent higher values of the VXO. This can be attributed to the less diversity in the index, looking at only a subset of options for the top 100 US companies as compared to a wide variety of options of the top 500 US companies.

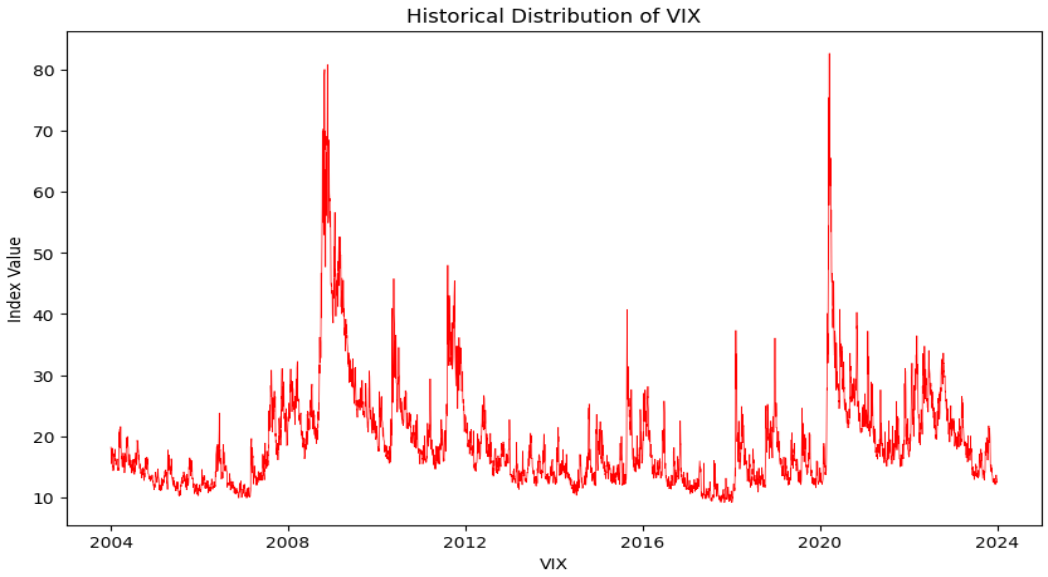


Figure 1(VIX)

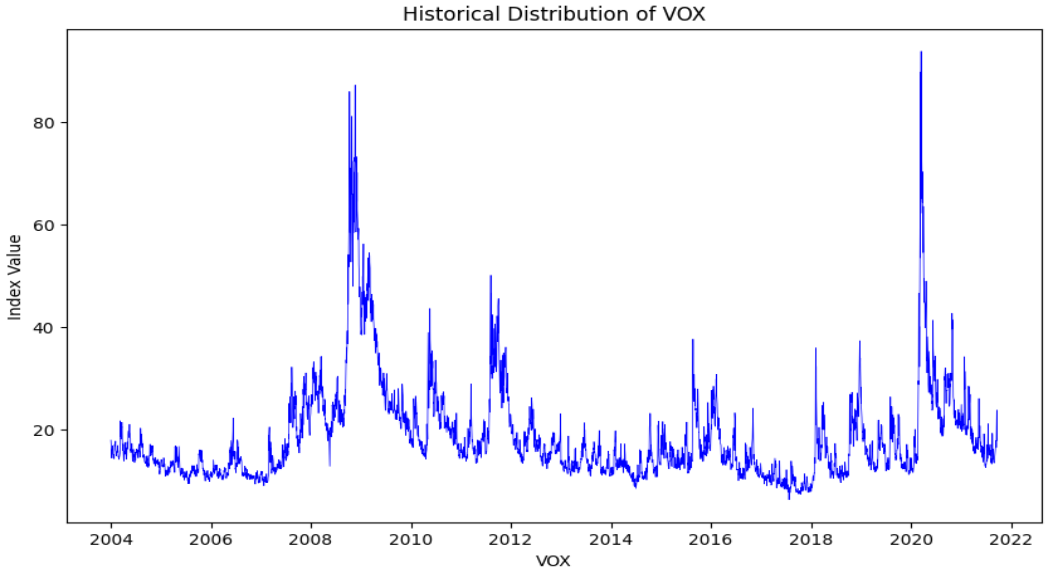


Figure 2 (VOX)

Historical Distribution of VIX vs. VXO

Firstly, when looking at the historical distributions of both the VIX and VXO, we must determine if they are stationary processes. Using an augmented Dickey-Fuller (ADF) test, we can determine if both processes have a unit root. The null hypothesis is that both processes have a unit root. Using the ADF test on both the VIX and VOX, we get an ADF of statistic of -5.422 and -4.684 with p values of 3.04e-06 and 9.01e-05 respectively. This means we can reject the null hypothesis and conclude both processes are stationary.

Calculating the moments of each distribution we get to Table 1:

	Mean	Variance	Skew	Kurtosis
VIX	*19.15*	*76.77*	*2.53*	*9.471*
VOX	*18.59*	*94.92*	*2.76*	*10.91*

Examining Table 1, we immediately notice the high skew and kurtosis in the data. Considering the standard normal values of skew and kurtosis are 1 and 3 respectively, we predict this distribution to be heavily skewed to the left with fat tails. Comparing this with the kernel estimate of our distribution in Figures 3,4 we see such a pattern appear. Furthermore, the average value of both indices around 18.75 with a higher variance as expected from the VOX due to its diversification compared to the VIX.

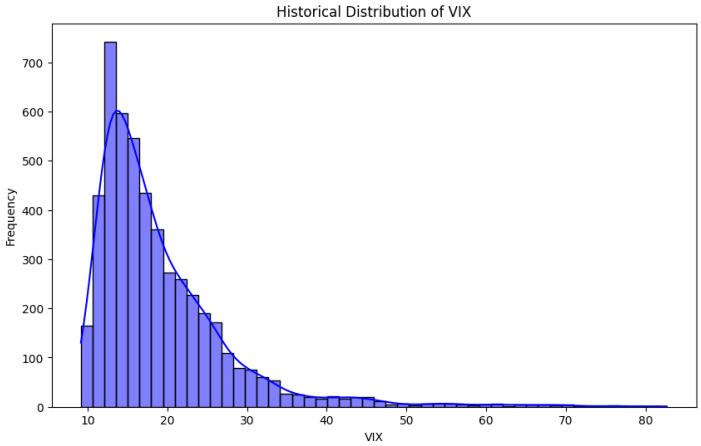


Figure 3: VIX Historical Distribution

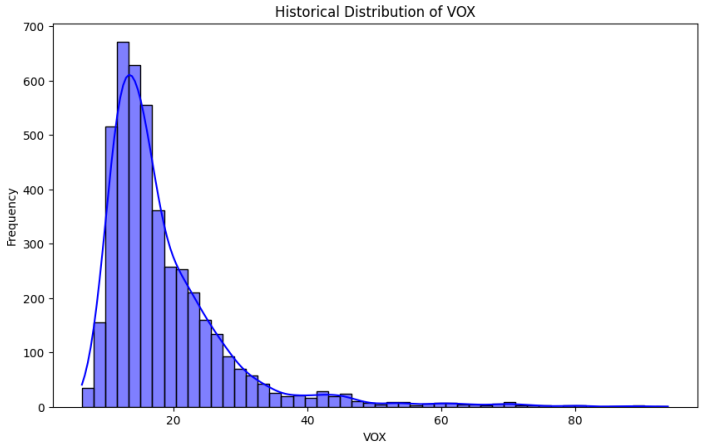


Figure 4: VOX Historical Distribution

Figures 5 through 8 represent the different Q-Q plots for both distributions, it's clear that the distributions are not Gaussian but instead exhibit characteristics of a lognormal distribution. Neither of the Q-Q plots aligns along the 45-degree line expected for a Gaussian distribution. Instead, there is a noticeable curvature in the middle, with significant deviations at both ends. This curvature indicates fat tails, meaning the distributions have high kurtosis and exhibit skewness as seen in Table 1.

Looking at Figures 5 through 8, we see that both the VIX and VOX data better fit the 45-degree line in Figures 6 and 8 as compared to Figures 5 and 7. This further suggest that both time series follow a lognormal distribution with one fat tail towards the top right of the plot rather than fat tails on both sides of the 45-degree line.

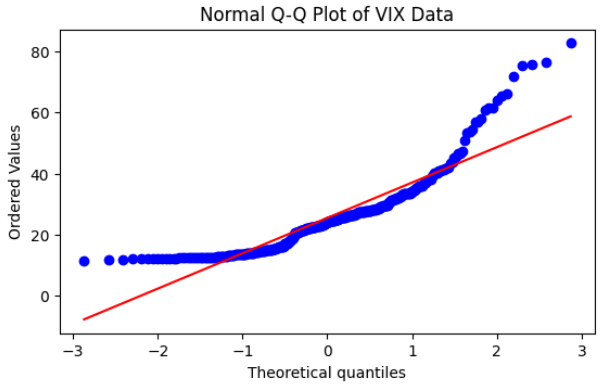


Figure 5: Normal Q-Q Plot of VIX

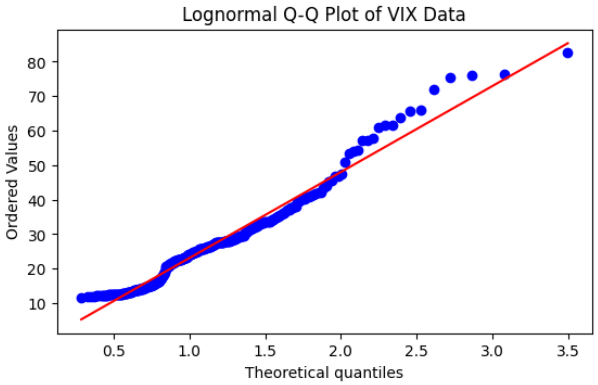


Figure 6: Lognormal Q-Q Plot of VIX

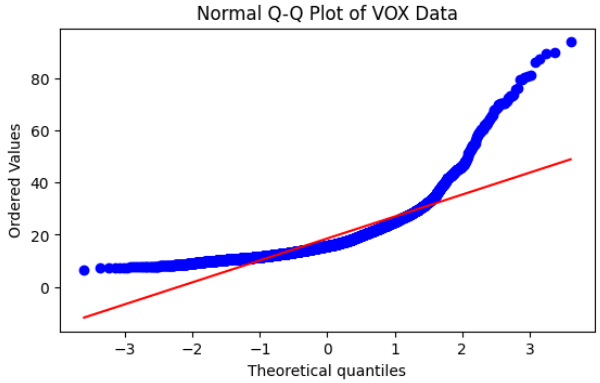


Figure 7: Normal Q-Q Plot of VOX

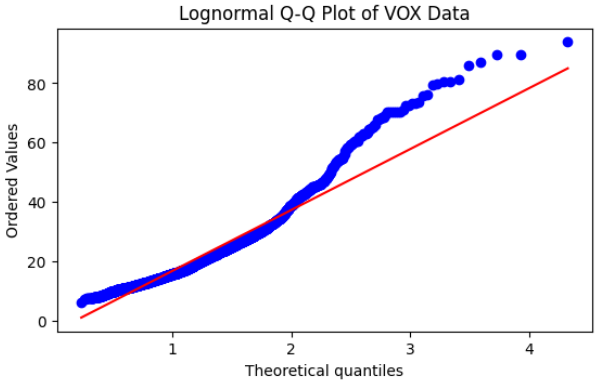


Figure 8: Lognormal Q-Q Plot of VOX

Rolling Window Computations

The market for the volatility of volatility consists of volatility derivatives, such as VIX futures and VIX options. These derivatives do not have a stock or index as their underlying asset; instead, they are based on the implied volatility of the market, specifically the volatility of the S&P 500 index as measured by the VIX. This allows traders to speculate on or hedge against changes in market sentiment rather than price movements of a specific index or stock. Investors use these products to manage risk in periods of high uncertainty, such as during economic downturns or significant market events, where volatility tends to spike. We see this in Figures 9 through 12 where during the 2010 depression, the rolling standard deviation had a very similar shape to the rolling median and VaRs. This allows traders to trade options on the VIX and VOX to hedge themselves against market downturns such as the 2008 housing market bubble. Furthermore, we see more recently the same activity where investors could use the variance of the VIX and VOX indexes to hedge against the COVID-19 crisis.

In general, we see that the rolling variance spikes earlier than the rolling mean and VaR's. This implies that the rolling variance of the VIX can be used as an early indicator of a market collapse like we saw in 2008 and like we are now seeing after COVID-19.

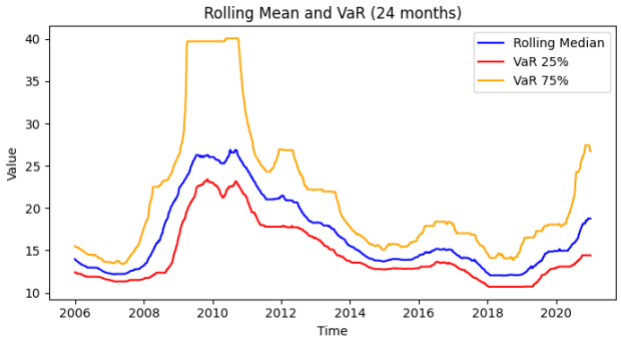


Figure 9: VIX Rolling Mean and VaR

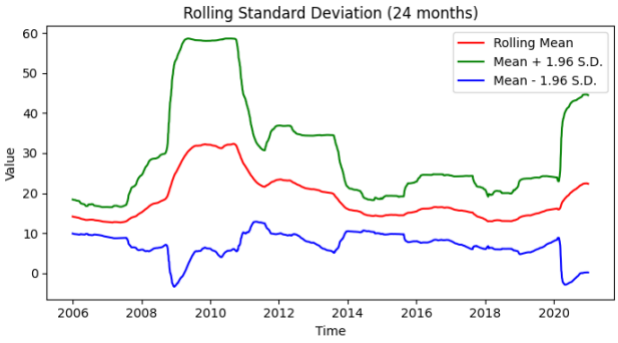


Figure 10: VIX Rolling Variance

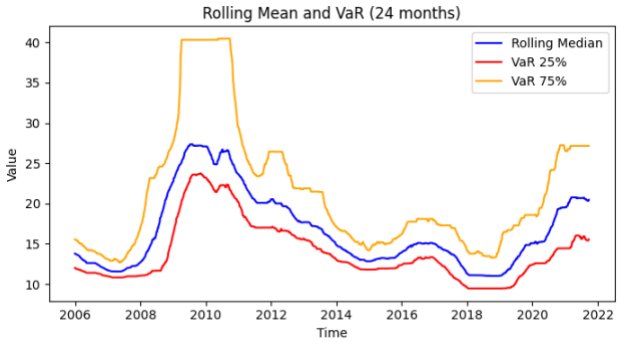


Figure 11: VOX Rolling Mean and VaR

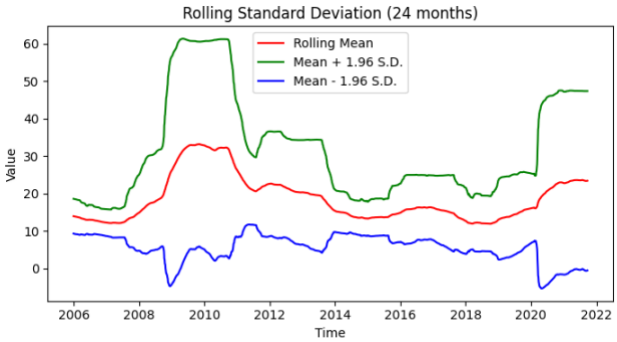


Figure 12: VOX Rolling Variance

Change In Methodology

Since the change in methodology, the VIX has become more comprehensive, covering options for 500 of the largest-cap companies in the S&P 500, rather than focusing solely on at-the-money options for the top 100 companies (S&P 100). The VXO is still calculated because it provides traders with insight into the sentiment difference between the "too big to fail" companies in the S&P 100 and the broader market represented by the other large-cap companies in the S&P 500.

This diversification is useful during market downturns. In times of crisis, the Federal Reserve may view the bankruptcy of any S&P 100 company as detrimental to the economy and consequently, it is more inclined to act if the situation arises. This safety net results in less implied volatility for these companies. However, the 400 companies in the S&P 500 but not in the S&P 100 may not receive the same federal support. As a result, the fear of collapse could be higher for these companies, leading to a divergence between the values

of the VIX and the VXO. This divergence helps traders gauge the severity of implied volatility between the top 100 companies and the remaining 400.

COVID-19 Pandemic

Focusing on the COVID-19 pandemic, we notice that the rolling mean and VaR's begin to increase towards the beginning of the pandemic but more importantly, the rolling variance of the VIX index blows up from a value of around 10 to 125. This is very similar to the 2008 crisis when the rolling variance shot up while the rolling mean and VaR's gradually increased. If we use the variance of the VIX as an indicator for panic in the market leading to an eventual depression, we can then quantify the COVID-19 pandemic one such potential event.

Predictability of the VIX

Literature Review of Efficient Market Hypothesis

According to Clark (1973), if we let X_t denote the price of an equity at time t, examining the data shows that X_t exhibits a random walk, which is characterized by the equation $X_t = X_{t-1} + \varepsilon_t$ where ε_t is random white noise (i.e., with an expectation of 0 and uncorrelated with ε_j for $t \neq j$). He also claims that $\forall t \epsilon \mathbb{N}, \Delta X_t$ are mutually independent but not normally distributed; instead, they are leptokurtic, for any process with stationary independent increments, as described previously. In general, the random walk assumes that the price tomorrow will be equal to the price today plus a white noise adjustment, rather than depending on previous iterations, which later defines it as a Markov process.

ARMA Specification on Logged Values

We begin by analyzing the series $y_t = \ln(VIX_t)$, representing the log returns of the VIX index, over the period from 2004 to the present, as shown in Figure 13. Next, we examine the Autocorrelation Function (ACF) for up to 300 lags, as illustrated in Figure 14, trying to estimate an autoregressive model for y_t.

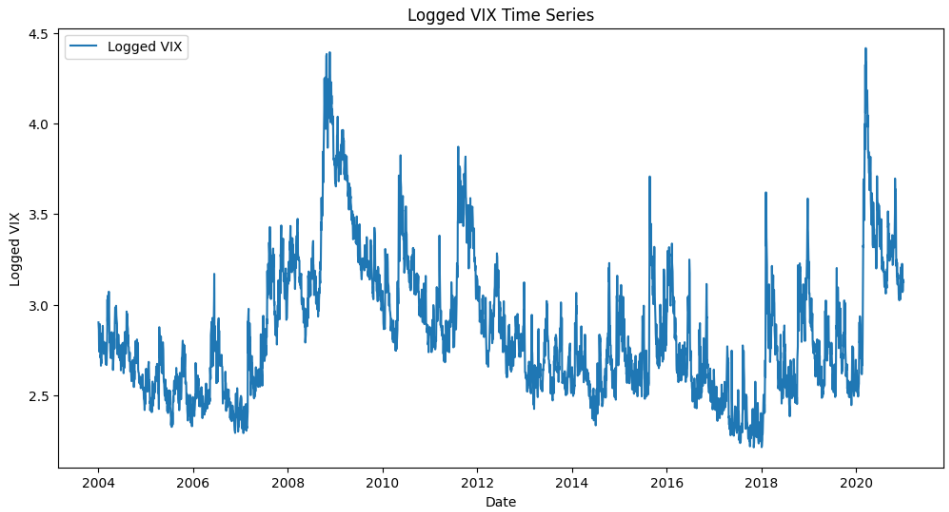


Figure 13 (Logged VIX Returns)

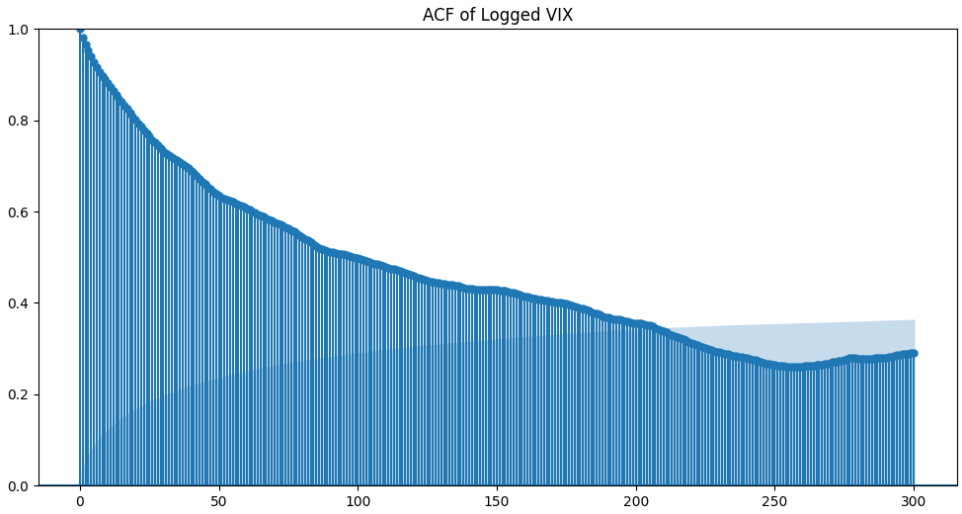


Figure 14 (ACF of Logged VIX Returns)

Upon examining the autocorrelation function (ACF) of the logged VIX returns shown in Figure 14, we observe that autocorrelation remains present up to approximately the 210th lag, at which point it becomes statistically insignificant (indicated by the light blue shaded confidence interval). Initially, the autocorrelation is close to 1.0, gradually decaying to around 0.2 over the course of approximately 260 lags, which corresponds to a full trading year.

To fit an autoregressive model to this data, we employed a grid search algorithm to identify the model with the lowest Akaike Information Criterion (AIC). This analysis points to the ARMA(2,1) model as the optimal choice for modeling y_t, with the corresponding coefficients shown in Table 2 below:

	Coefficient	S.E.	P Value
ϕ_1	*1.7048*	*(0.044)*	*0.000*
ϕ_2	*-0.7074*	*(0.043)*	*0.000*
θ_1	*-0.8018*	*(0.038)*	*0.000*

Revisiting the ADF test from Section 1.2 of our analysis, we rejected the possibility of a unit root in the VIX data. When applying the ADF test to the logged data, we obtained an ADF statistic of -4.52 with a p-value of 0.00018, allowing us to reject the null hypothesis that the logged data has a unit root (i.e., the logged data is stationary). This result contrasts with our ARMA(2,1) model, where the sum of the coefficients $\phi_1 + \phi_2 = 0.9974$, is very close to 1, suggesting non-stationarity.

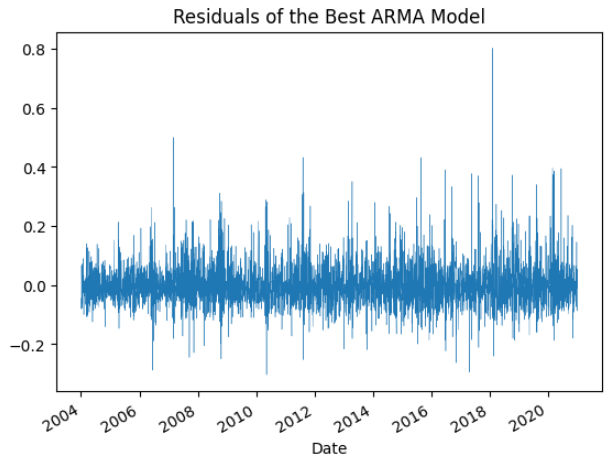


Figure 15 (Residuals of ARMA(2,1) Model)

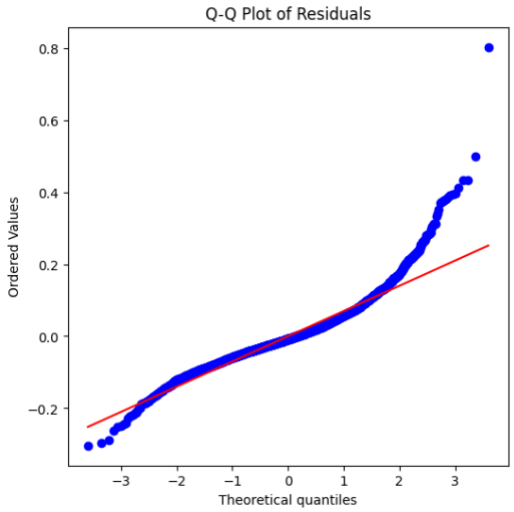


Figure 16 (Q-Q Plot of Residuals)

Focusing on the residuals of the model (Figure 15), we observe that they appear to resemble a white noise process, as one would expect from an ARMA model. However, a closer examination of the Q-Q plot in Figure 16 reveals that the residuals do not follow a normal distribution. Additionally, the results of the Ljung-Box Test yield a p-statistic of 0.31 implying series correlation amongst the residuals and the Jarque-Bera test yields a significant statistic of11592.8996, allow us to reject the null hypothesis of normality. These findings suggest that the model may be mis-specified.

ARMA Specification on Logged Differences

We now shift the focus of our analysis to the change in the logged VIX data: $\Delta y_t = y_t - y_{t-1} = \ln(VIX_t) - \ln(VIX_{t-1})$. After applying this transformation to the data, we observe that it appears to be stationary, as shown in Figure 17. The autocorrelation function (ACF) of this differenced data differs significantly from that of the original logged data. In Figure 18, we see that only the first two lag exhibits significant autocorrelation, suggesting an AR(2) process contrary to y_t model where the ACF was significant until the 210th lag.

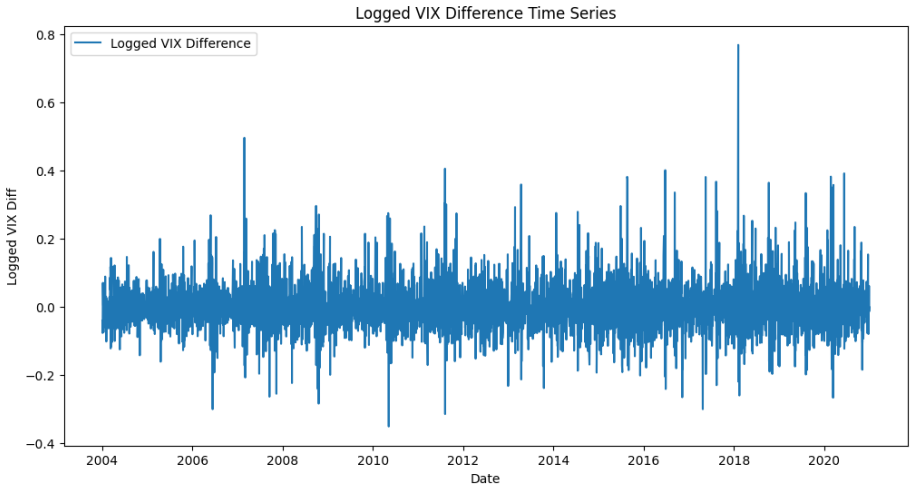


Figure 17 (Logged VIX Difference Over Time)

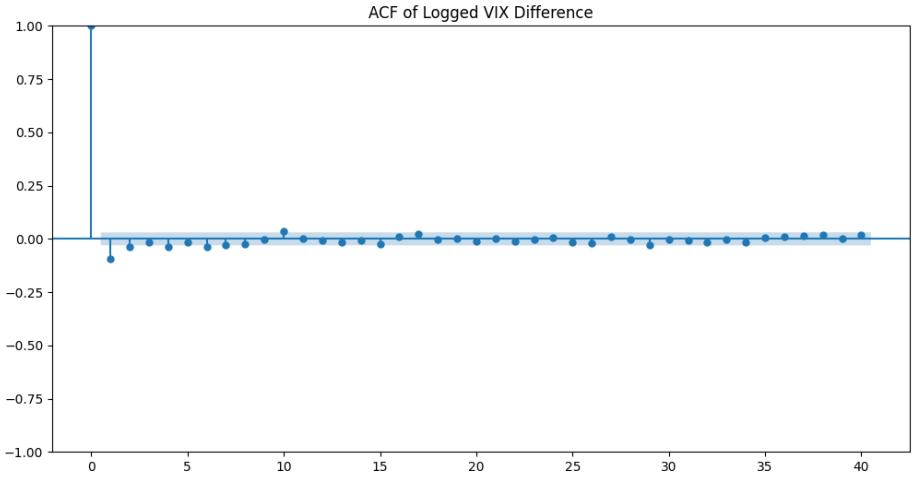


Figure 18 (ACF of Logged VIX Differenced Data)

Again, using the AIC to find the best fitting model, we find that the ARMA(3,1) with parameters seen in Table 3 below, best fit the data.

	Coefficient	S.E.	P-Value
ϕ_1	*0.8768*	*(0.011)*	*0.000*
ϕ_2	*0.0454*	*(0.014)*	*0.002*
ϕ_3	*0.0293*	*(0.011)*	*0.011*
θ_1	*-0.9861*	*(0.005)*	*0.000*

We find that the ADF statistic is a significant -23.4, once again rejecting the null hypothesis that the data is non-stationary. From Table 3, the sum of the coefficients is still slightly lower than 1, but higher than in the y_t model. Continuing with our comparison, we examine the residuals and the resulting Q-Q plot, as shown in Figures 19 and 20.

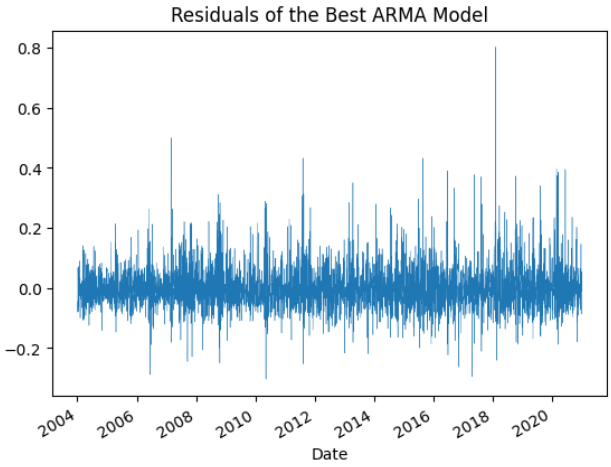


Figure 19 (Residuals of ARMA(3,1) Model)

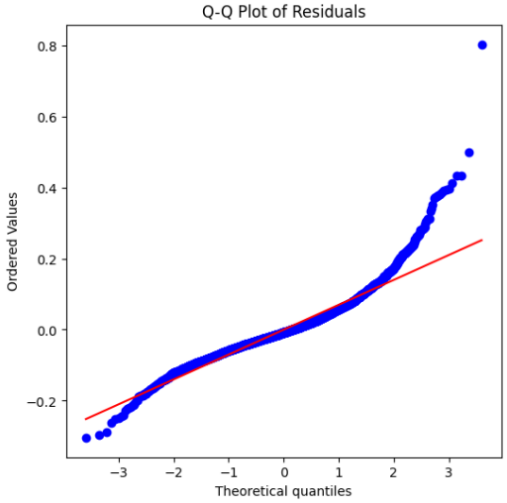


Figure 20 (Q-Q Plot of Residuals)

Similarly to the y_t model, the Δy_t model appears to have residuals that are random, as shown in Figure 19. However, this assumption is disproven upon closer inspection. The Q-Q plot reveals that the residuals are not normally distributed, and the Ljung-Box and Jarque-Bera tests further confirm that the residuals are both serially correlated and not normally distributed. Overall, this ARMA model for Δy_t can be seen as an integrated version of the y_t model. However, due to the serial correlation present in the residuals, the ARMA model does not adequately fit our data, so it seems that the VIX does not exhibit a linear structure, making the ARMA model unsuitable for modeling the VIX.

Applying Efficient Market Hypothesis to VIX

The Efficient Market Hypothesis (EMH) states that prices follow a random walk with no long-term "steady-state" or average level. Applying the EMH to our analysis of the VIX implies that its returns should exhibit a random walk, represented by the following equation:

$$y_t = \alpha + y_{t-1} + \varepsilon$$

where y_t is the log returns of the VIX. Furthermore, notice that by subtracting y_{t-1} from both sides and substituting for Δy_t we get:

$$\Delta y_t = \alpha + \varepsilon.$$

This intuitively means the difference in log returns resembles a random walk ε up to a drift of α. Turning our attention back to the VIX data, we conduct a regression to test for the presence of the Efficient Market Hypothesis (EMH):

$$\Delta y_t = \alpha + \beta_1 \Delta y_{t-1} + \beta_2 \Delta y_{t-1}^2 + \varepsilon$$

Table 4	α	β_1	β_2	R^2
Reg 1: Δy_t	***6.641e-05**** *(0.001)*	***-0.950**** *(0.015)*	-	***0.009***
Reg 2: Δy_t	***0.0020**** *(0.001)*	-	***-0.3647**** *(0.071)*	***0.006***
Reg 3: Δy_t	***0.0014*** *(0.001)*	***-0.0769**** *(0.016)*	***-0.2369**** *(0.076)*	***0.011***

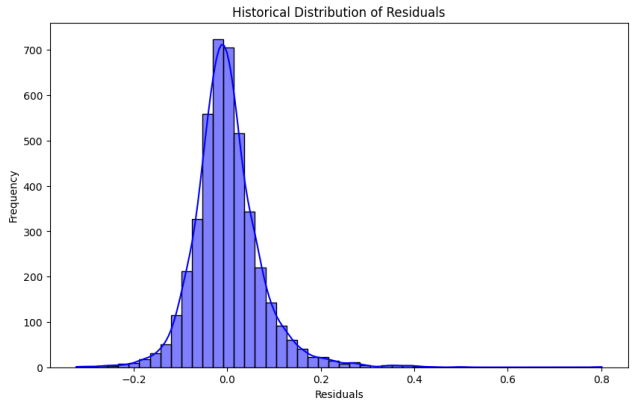
****: Statistical Significance at a 5% level*
***: Statistical Significance at a 10% Level*

Upon examining the regression results, we initially expected that both $\beta_{1,2}$ would equal zero to confirm the Efficient Market Hypothesis (EMH) in our data. However, contrary to this expectation, both $\beta_{1,2}$ are statistically significant at the 5% level across all three regressions. Focusing on Regression 3, we observe a significant, non-zero correlation between both the lagged differenced log returns (Δy_t), its previous iteration (Δy_{t-1}) and the squared lagged differenced log returns (Δy_{t-1}^2), directly contradicting the EMH as we defined it.

Furthermore, the significant negative correlation between the current change in log returns (Δy_t) and its lagged value (Δy_{t-1}) suggests that the VIX can serve as a hedge against short-term market volatility, as discussed in Chapter 1. Additionally, the strong negative correlation between Δy_{t-1}^2 and Δy_t, in Regression 3 indicates that the VIX exhibits short-term predictability based on previous levels of volatility. This result further refutes the EMH in the short term, implying that VIX movements can be anticipated to some extent using past volatility patterns.

Residual Analysis

Turning our attention to the residuals of our regression, we plot the historical distribution (Figure 20) and associated Q-Q plot (Figure 21) to determine normality.



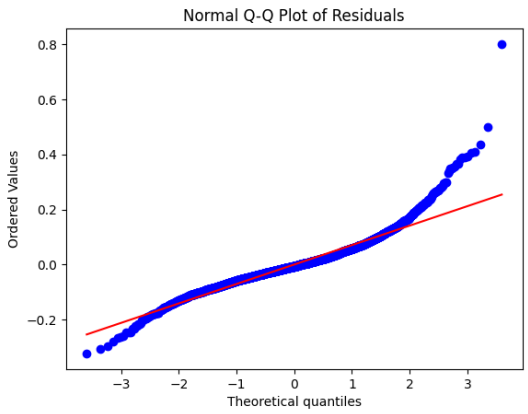


Figure 20 (Historical Distribution of Residuals) Figure 21 (Normal Q-Q Plot of Residuals)

Based on the Efficient Market Hypothesis (EMH), we assume that our model is influenced by random white noise with a constant drift. In Figure 20, the residuals visually appear to follow a Gaussian distribution, as expected. However, a closer examination in Figure 21 reveals that the distribution actually exhibits fat tails, or high kurtosis. This observation is confirmed by the moments presented in Table 5:

	Mean	Variance	Skew	Kurtosis
Residuals	-1.34e-18	0.005	1.177	7.198

Table 5 reveals a mean that is close to zero, aligning with what we would expect in a Gaussian distribution. However, the variance is smaller than the Gaussian benchmark of 1, indicating that the data is more tightly clustered around the mean than anticipated. Examining the third and fourth moments, we observe a slight positive skew and a kurtosis significantly higher than what would be expected in a normal distribution. These findings raise questions about potential model misspecifications, such as the assumption that each residual is independently drawn. To investigate further, we turn to the ACF of the residuals and residuals squared (Figure 22) with this potential issue in mind.

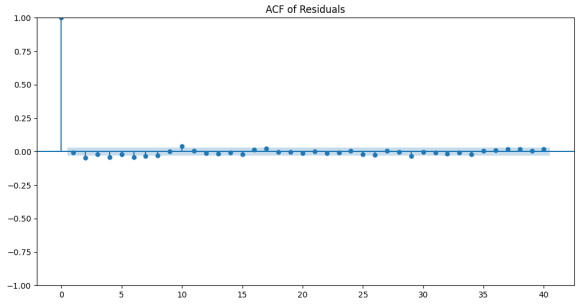


Figure 22 (ACF of Residuals)

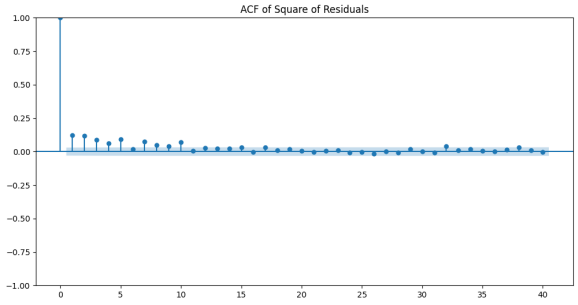


Figure 23 (ACF of Residuals Squared)

In Figure 22, we observe a significant autocorrelation at lag 1, while the remaining lags fail to reject the null hypothesis of zero autocorrelation. This figure indicates a short-term autocorrelation in the residuals, leading to serial correlation between consecutive residual draws. Figure 23 further shows a significant autocorrelation for approximately the first 10 lags, suggesting short-term correlation in the volatility of the residuals. Overall, our regression residuals do not follow a Gaussian distribution, and the presence of serial correlation contradicts the Efficient Market Hypothesis (EMH) in the short term.

Predictability Puzzle

Extending our analysis in the previous section, we look to at the predictability of the VIX over h periods. We firstly define $\Delta y_{t+1,t+h} = \Delta y_{t+1} + \Delta y_{t+2} + \cdots + \Delta y_{t+h} = \sum_{i=1}^{h} \Delta y_{t+i}$ as the sum of the logged returns h periods into the future with Δy_t defined as it has been in

previous chapters. Secondly, we define $\Delta y_{t-k+1,t} = \Delta y_t + \Delta y_{t-1} + \cdots + \Delta y_{t-k+1} = \sum_{i=1}^{k} \Delta y_{t-i+1}$ as the sum of logged returns h periods into the past, where once again, Δy_t is defined similarly to previous chapters. We then define the regression:

$$\Delta y_{t+1,t+h} = \beta_h \Delta y_{t-k+1,t} + w_{t,h}$$

Let *h* represent the time horizon at which we aim to predict the VIX, using the sum of logged returns from *h* periods ago along a white noise component. We focus specifically on β_h, which reflects the relationship between past returns over *h* periods and future returns over the same period. To extend our analysis, we examine the R^2 of our regression, which represents the variance in the sum of future logged returns explained by the sum of past logged returns over *h* periods. In this context, we interpret R^2as a measure of predictability.

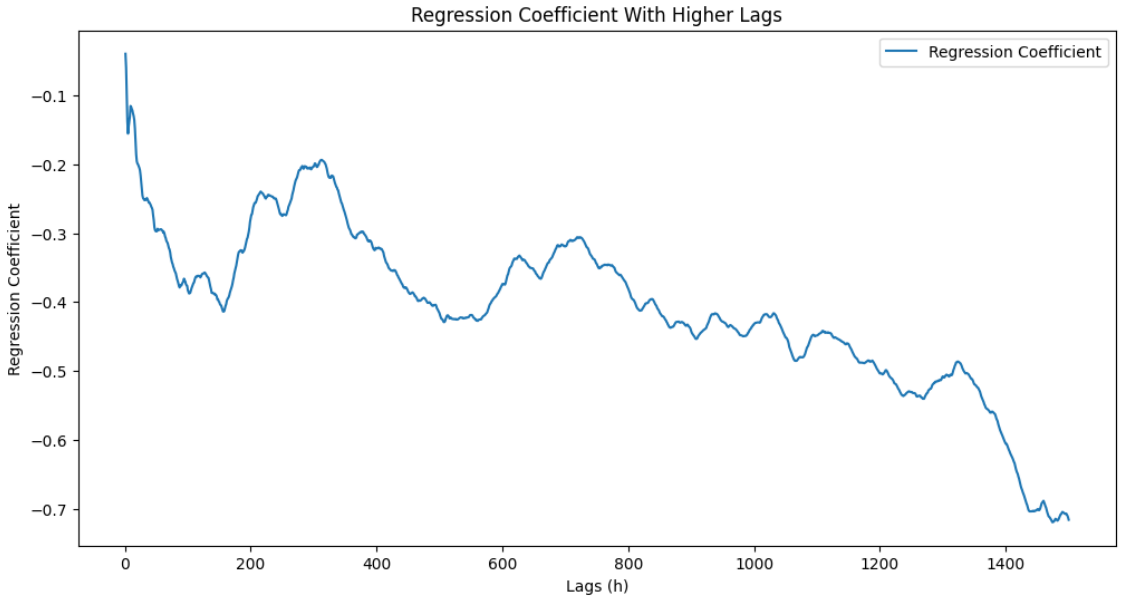


Figure 24 (Regression Coefficient)

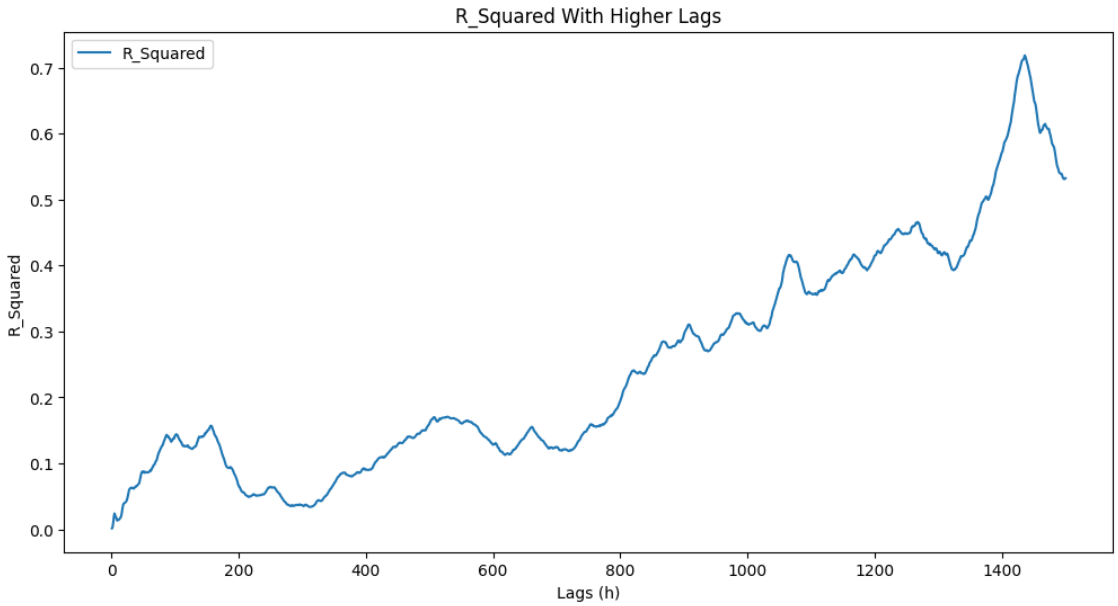


Figure 25 (R Squared of Regression)

Figures 24 and 25 reveal a trend that contradicts the Efficient Market Hypothesis (EMH), showing that predictability increases up to around the 1400th lag, with the regression coefficient also trending downward. This implies that the VIX becomes more predictable as the forecast horizon extends. This finding mirrors the conclusion of Fama and French (1988), who observed that 'autocorrelations become negative for 2-year returns and reach their lowest points for 3-5-year returns' when predicting stock returns.

References

Cboe Exchange, Inc. (2019). CBOE Volatility Index®. In *White Paper*. https://www.sfu.ca/~poitras/419_VIX.pdf

Clark, P. K. (1973). A Subordinated Stochastic Process Model with Finite Variance for Speculative Prices. *Econometrica*, *41*(1), 135. https://doi.org/10.2307/1913889

Fama, E. F., & French, K. R. (1988). Permanent and temporary components of stock prices. *Journal of Political Economy*, *96*(2), 246–273. https://doi.org/10.1086/261535

Saha, A., Malkiel, B. G., & Rinaudo, A. (2018). Has the VIX index been manipulated? *Journal of Asset Management*, *20*(1), 1–14. https://doi.org/10.1057/s41260-018-00102-4

Wikipedia contributors. (2024, September 3). *Implied volatility*. Wikipedia. https://en.wikipedia.org/wiki/Implied_volatility

Appendix

```
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from statsmodels.tsa.stattools import adfuller
import warnings
warnings.filterwarnings("ignore", message="DataFrame is highly fragmented")
warnings.filterwarnings("ignore", message=".*A date index has been provided,
but it has no associated frequency information.*")
```

Assignment 1:

```
vix = yf.download('^VIX', start='2004-01-01', end='2024-01-01')
vox = yf.download('^VXO', start='2004-01-01', end='2024-01-01')

# Step 2: Extract the 'Close' prices of VIX for analysis
vix_data = vix['Close'].dropna()
vox_data = vox['Close'].dropna()
```

```
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
```

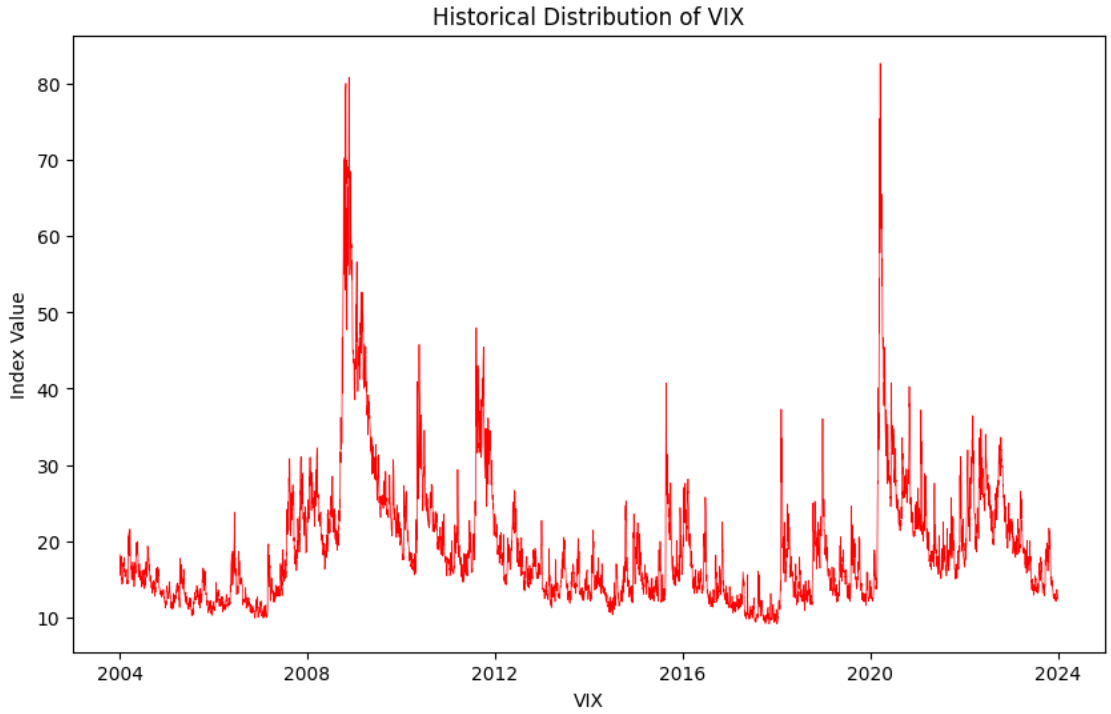
```
# Perform the ADF test
result = adfuller(vix['Close'])

print('ADF Statistic:', result[0])
print('p-value:', result[1])
if result[1] < 0.05:
    print("Reject the null hypothesis: the series is stationary.")
else:
    print("Fail to reject the null hypothesis: the series has a unit root.")
```

```
ADF Statistic: -5.422248613196907
p-value: 3.041176121007876e-06
Reject the null hypothesis: the series is stationary.
```

```
# Perform the ADF test
result = adfuller(vox['Close'])

print('ADF Statistic:', result[0])
print('p-value:', result[1])
if result[1] < 0.05:
    print("Reject the null hypothesis: the series is stationary.")
else:
    print("Fail to reject the null hypothesis: the series has a unit root.")
```

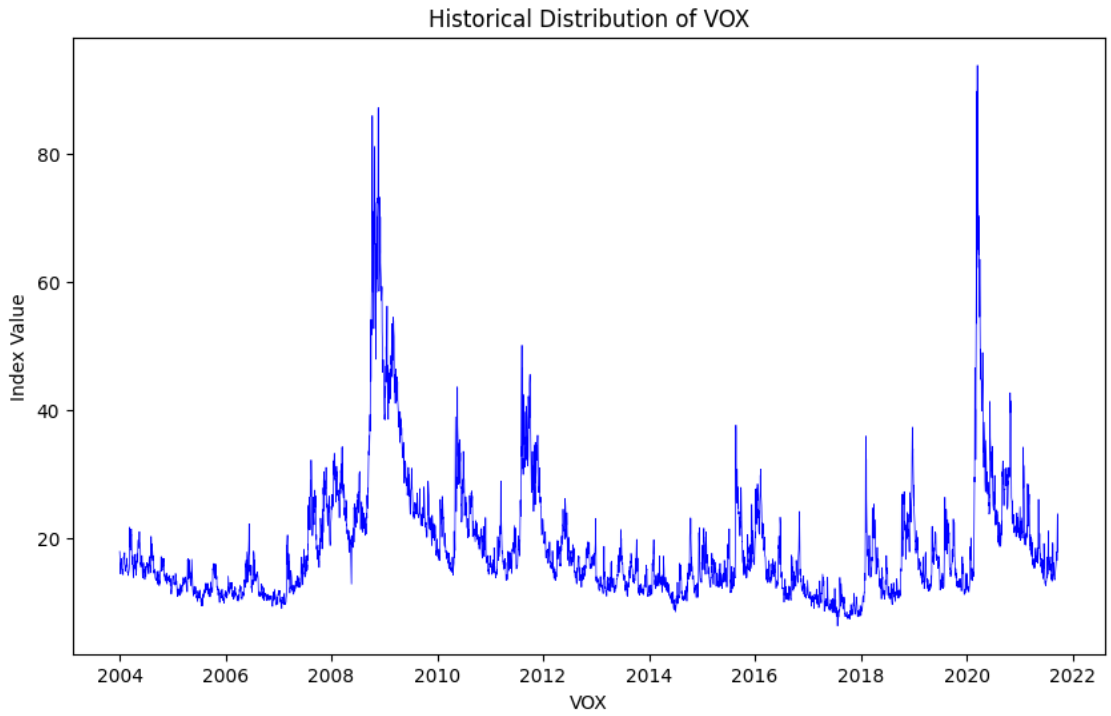
```
ADF Statistic: -4.684101957014764
p-value: 9.013409502008162e-05
Reject the null hypothesis: the series is stationary.
```

```
plt.figure(figsize=(10, 6))
plt.plot(vix_data.index, vix_data, label='Vox Series Data', color='red', line
width = 0.5)
plt.title("Historical Distribution of VIX")
plt.xlabel("VIX")
plt.ylabel("Index Value")
plt.show()
```



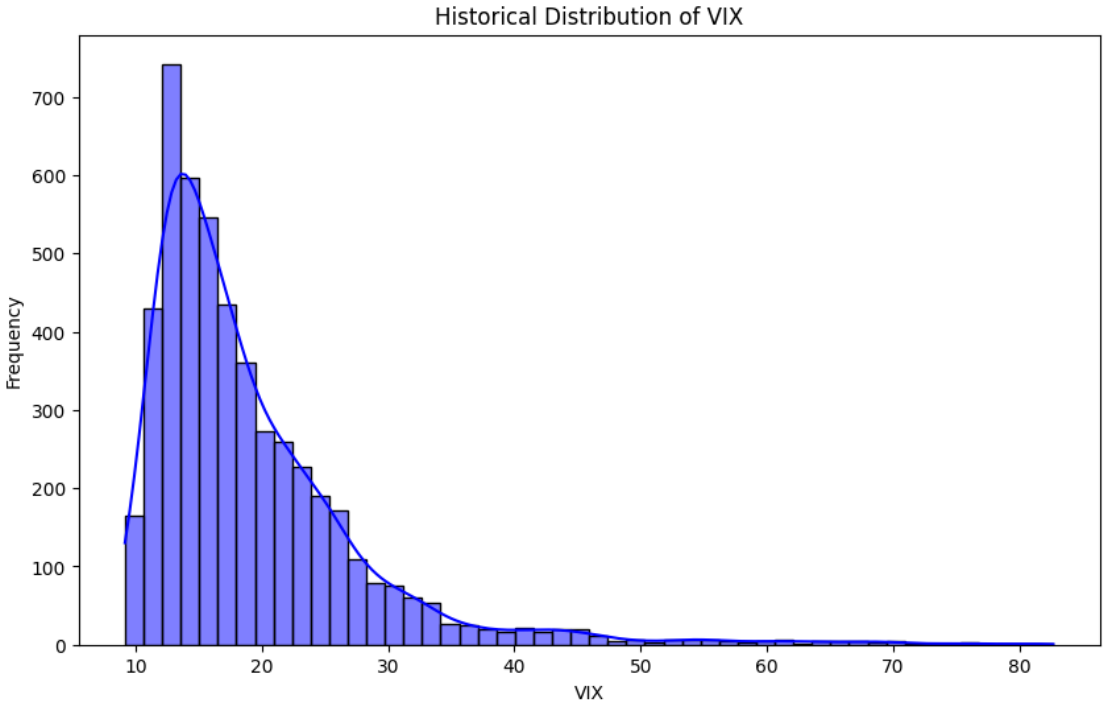
png

```
plt.figure(figsize=(10, 6))
plt.plot(vox_data.index, vox_data, label='Vox Series Data', color='blue', lin
ewidth = 0.5)
plt.title("Historical Distribution of VOX")
plt.xlabel("VOX")
plt.ylabel("Index Value")
plt.show()
```



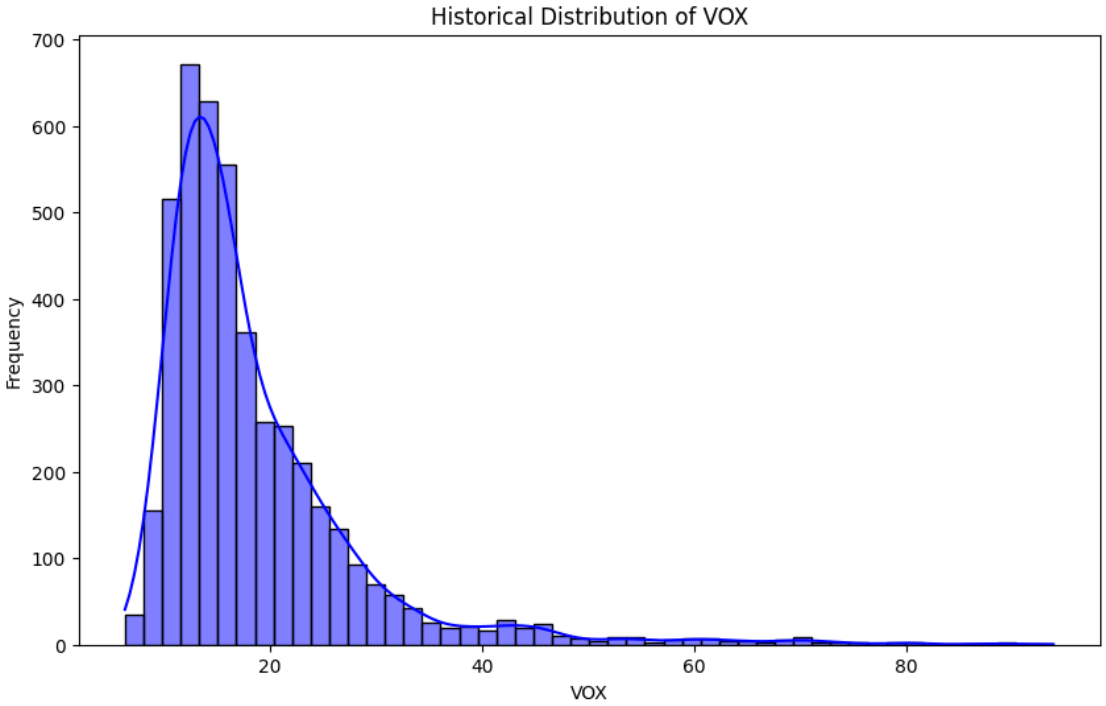
png

```
plt.figure(figsize=(10, 6))
sns.histplot(vix_data, kde=True, bins=50, color='blue')
plt.title("Historical Distribution of VIX")
plt.xlabel("VIX")
plt.ylabel("Frequency")
plt.show()
```



png

```
plt.figure(figsize=(10, 6))
sns.histplot(vox_data, kde=True, bins=50, color='blue')
plt.title("Historical Distribution of VOX")
plt.xlabel("VOX")
plt.ylabel("Frequency")
plt.show()
```



png

```
mean_vix = np.mean(vix_data)
variance_vix = np.var(vix_data)
skewness_vix = stats.skew(vix_data)
kurtosis_vix = stats.kurtosis(vix_data)

print(f"Mean: {mean_vix}")
print(f"Variance: {variance_vix}")
print(f"Skewness: {skewness_vix}")
print(f"Kurtosis: {kurtosis_vix}")
```

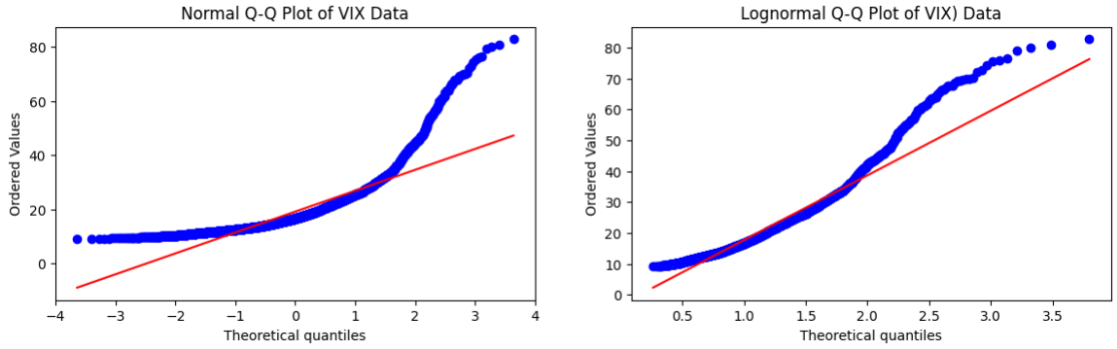
```
Mean: 19.147275976601307
Variance: 76.77095148725881
Skewness: 2.5348579151213344
Kurtosis: 9.471483240240307
```

```
mean_vox = np.mean(vox_data)
variance_vox = np.var(vox_data)
skewness_vox = stats.skew(vox_data)
kurtosis_vox = stats.kurtosis(vox_data)

print(f"Mean: {mean_vox}")
print(f"Variance: {variance_vox}")
print(f"Skewness: {skewness_vox}")
print(f"Kurtosis: {kurtosis_vox}")
```

```
Mean: 18.59028506509115
Variance: 94.91898528036018
```

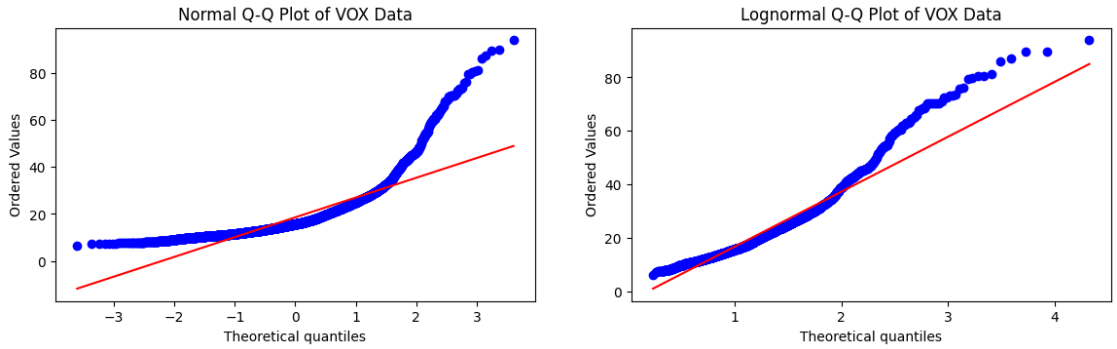
```
Skewness: 2.7644457259622586
Kurtosis: 10.913302212897303
```

```
plt.figure(figsize=(14, 8))
shape, loc, scale = stats.lognorm.fit(vix_data, floc=0)
plt.subplot(2, 2, 1)
stats.probplot(vix_data, dist="norm", plot=plt)
plt.title("Normal Q-Q Plot of VIX Data")

plt.subplot(2, 2, 2)
stats.probplot(vix_data, dist="lognorm",sparams=(shape,), plot = plt)
plt.title("Lognormal Q-Q Plot of VIX) Data")
plt.show()
```



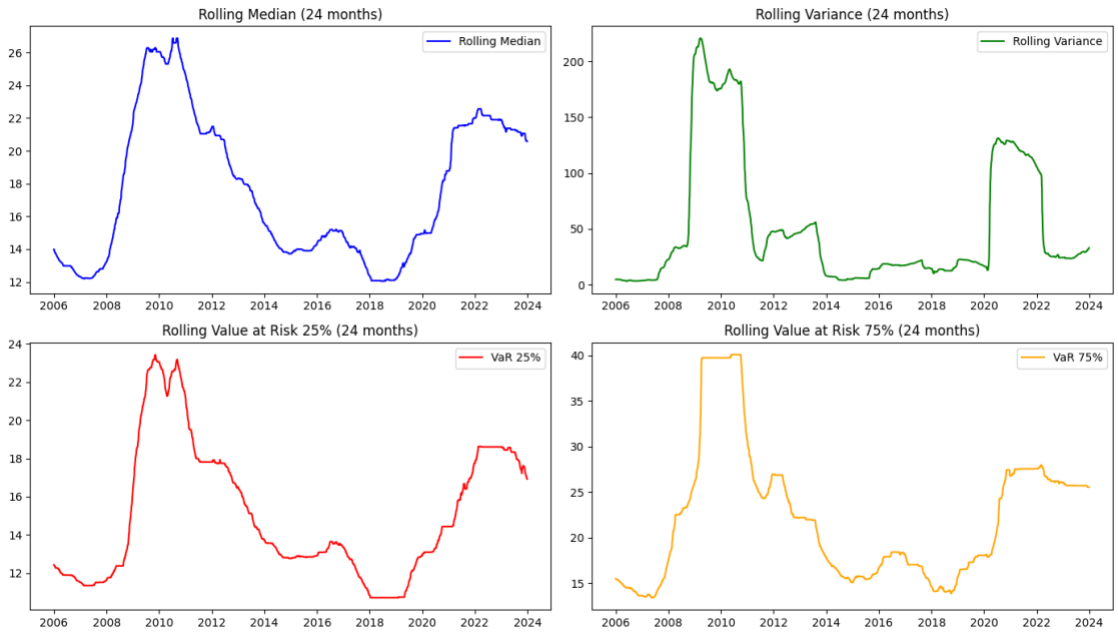
png

```
plt.figure(figsize=(14, 8))
shape, loc, scale = stats.lognorm.fit(vox_data, floc=0)
plt.subplot(2, 2, 1)
stats.probplot(vox_data, dist="norm", plot=plt)
plt.title("Normal Q-Q Plot of VOX Data")

plt.subplot(2, 2, 2)
stats.probplot(vox_data, dist="lognorm",sparams=(shape,), plot = plt)
plt.title("Lognormal Q-Q Plot of VOX Data")
plt.show()
```



png

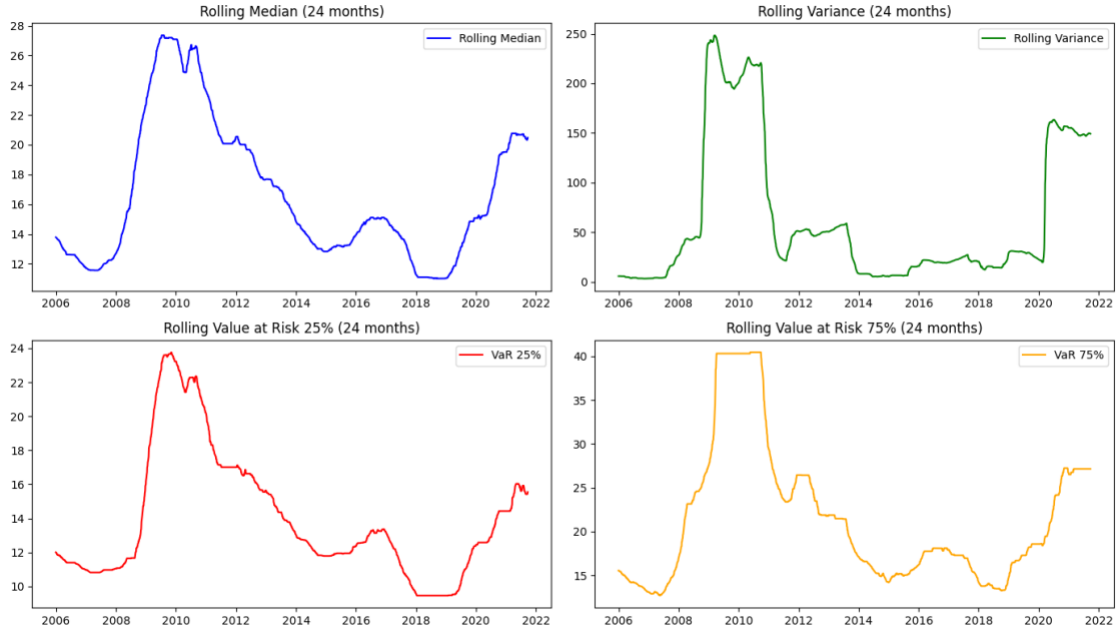
```
rolling_window = 504
rolling_VaR_50 = vix_data.rolling(window=rolling_window).quantile(0.5)
rolling_variance = vix_data.rolling(window=rolling_window).var()
rolling_VaR_25 = vix_data.rolling(window=rolling_window).quantile(0.25)
rolling_VaR_75 = vix_data.rolling(window=rolling_window).quantile(0.75)

plt.figure(figsize=(14, 8))

# Plot rolling mean
plt.subplot(2, 2, 1)
plt.plot(rolling_VaR_50, label='Rolling Median', color='blue')
plt.title('Rolling Median (24 months)')
plt.legend()

# Plot rolling variance
plt.subplot(2, 2, 2)
plt.plot(rolling_variance, label='Rolling Variance', color='green')
plt.title('Rolling Variance (24 months)')
plt.legend()

# Plot VaR at 25%
plt.subplot(2, 2, 3)
plt.plot(rolling_VaR_25, label='VaR 25%', color='red')
plt.title('Rolling Value at Risk 25% (24 months)')
plt.legend()

# Plot VaR at 75%
plt.subplot(2, 2, 4)
plt.plot(rolling_VaR_75, label='VaR 75%', color='orange')
plt.title('Rolling Value at Risk 75% (24 months)')
plt.legend()

plt.tight_layout()
plt.show()
```



png

```
rolling_window = 504
rolling_mean = vox_data.rolling(window=rolling_window).mean()
rolling_VaR_50 = vox_data.rolling(window=rolling_window).quantile(0.5)
rolling_variance = vox_data.rolling(window=rolling_window).var()
rolling_VaR_25 = vox_data.rolling(window=rolling_window).quantile(0.25)
rolling_VaR_75 = vox_data.rolling(window=rolling_window).quantile(0.75)
rolling_sd = np.sqrt(rolling_variance)

plt.figure(figsize=(14, 8))

# Plot rolling mean
plt.subplot(2, 2, 1)
plt.plot(rolling_VaR_50, label='Rolling Median', color='blue')
plt.title('Rolling Median (24 months)')
plt.legend()

# Plot rolling variance
plt.subplot(2, 2, 2)
plt.plot(rolling_variance, label='Rolling Variance', color='green')
plt.title('Rolling Variance (24 months)')
plt.legend()

# Plot VaR at 25%
plt.subplot(2, 2, 3)
plt.plot(rolling_VaR_25, label='VaR 25%', color='red')
plt.title('Rolling Value at Risk 25% (24 months)')
plt.legend()

# Plot VaR at 75%
```

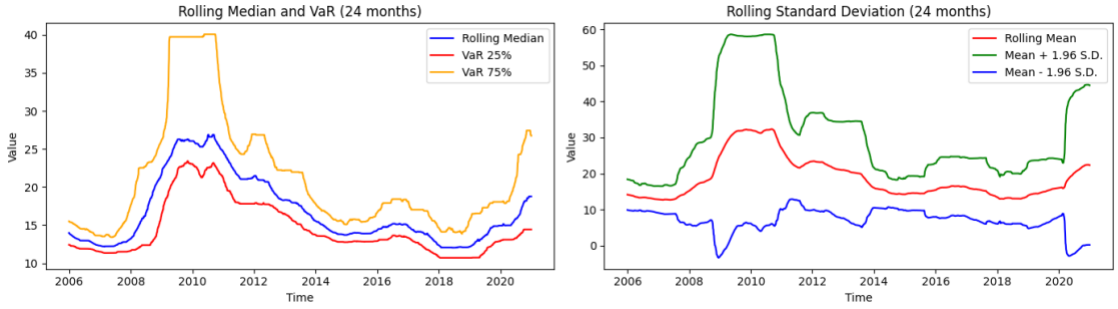
```
plt.subplot(2, 2, 4)
plt.plot(rolling_VaR_75, label='VaR 75%', color='orange')
plt.title('Rolling Value at Risk 75% (24 months)')
plt.legend()

plt.tight_layout()
plt.show()
```



png

```
plt.figure(figsize=(14, 4))

# First subplot: Rolling Mean and both VaRs on the same graph
plt.subplot(1, 2, 1)  # 1 row, 2 columns, 1st subplot
plt.plot(rolling_VaR_50, label='Rolling Median', color='blue')
plt.plot(rolling_VaR_25, label='VaR 25%', color='red')
plt.plot(rolling_VaR_75, label='VaR 75%', color='orange')
plt.title('Rolling Median and VaR (24 months)')
plt.xlabel('Time')
plt.ylabel('Value')
plt.legend()

# Second subplot: Rolling Variance
plt.subplot(1, 2, 2)  # 1 row, 2 columns, 2nd subplot
plt.plot(rolling_mean, label = 'Rolling Mean', color = 'red')
plt.plot(rolling_mean + 1.96 * rolling_sd, label='Mean + 1.96 S.D.', color='g
reen')
plt.plot(rolling_mean - 1.96 * rolling_sd, label='Mean - 1.96 S.D.', color='b
lue')
plt.title('Rolling Standard Deviation (24 months)')
plt.xlabel('Time')
```

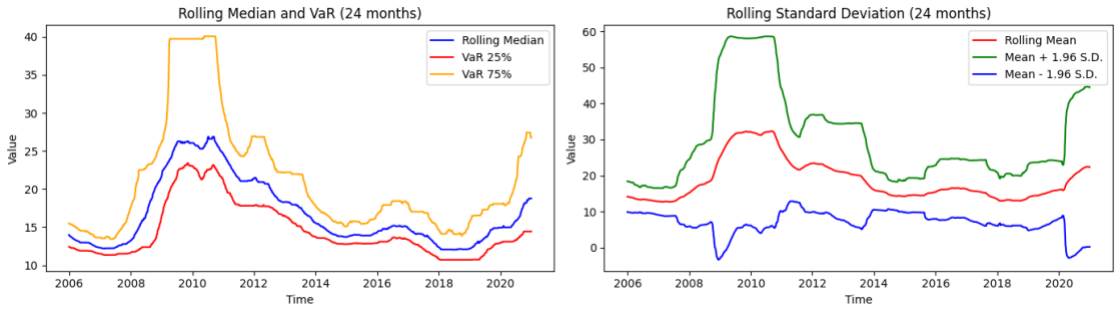
```
plt.ylabel('Value')
plt.legend()

# Adjust layout
plt.tight_layout()

# Show plot
plt.show()
```



png

```
vix = yf.download('^VIX', start='2004-01-01', end='2021-01-01')
vix_data = vix['Close'].dropna()
rolling_window = 504
rolling_mean = vix_data.rolling(window=rolling_window).mean()
rolling_VaR_50 = vix_data.rolling(window=rolling_window).quantile(0.5)
rolling_variance = vix_data.rolling(window=rolling_window).var()
rolling_sd = np.sqrt(rolling_variance)
rolling_VaR_25 = vix_data.rolling(window=rolling_window).quantile(0.25)
rolling_VaR_75 = vix_data.rolling(window=rolling_window).quantile(0.75)

[**********************100%**********************]  1 of 1 completed

plt.figure(figsize=(14, 4))

# First subplot: Rolling Mean and both VaRs on the same graph
plt.subplot(1, 2, 1)  # 1 row, 2 columns, 1st subplot
plt.plot(rolling_VaR_50, label='Rolling Median', color='blue')
plt.plot(rolling_VaR_25, label='VaR 25%', color='red')
plt.plot(rolling_VaR_75, label='VaR 75%', color='orange')
plt.title('Rolling Median and VaR (24 months)')
plt.xlabel('Time')
plt.ylabel('Value')
plt.legend()

# Second subplot: Rolling Variance
plt.subplot(1, 2, 2)  # 1 row, 2 columns, 2nd subplot
plt.plot(rolling_mean, label = 'Rolling Mean', color = 'red')
plt.plot(rolling_mean + 1.96 * rolling_sd, label='Mean + 1.96 S.D.', color='g
reen')
plt.plot(rolling_mean - 1.96 * rolling_sd, label='Mean - 1.96 S.D.', color='b
```

```
lue')
plt.title('Rolling Standard Deviation (24 months)')
plt.xlabel('Time')
plt.ylabel('Value')
plt.legend()

# Adjust layout
plt.tight_layout()

# Show plot
plt.show()
```



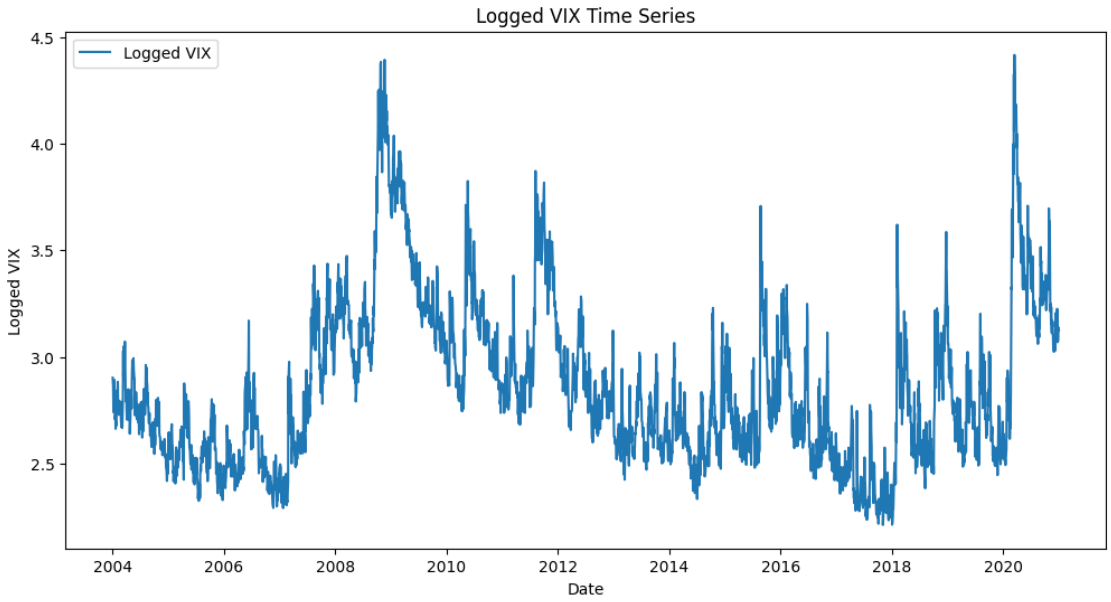
png

Assigment 2:

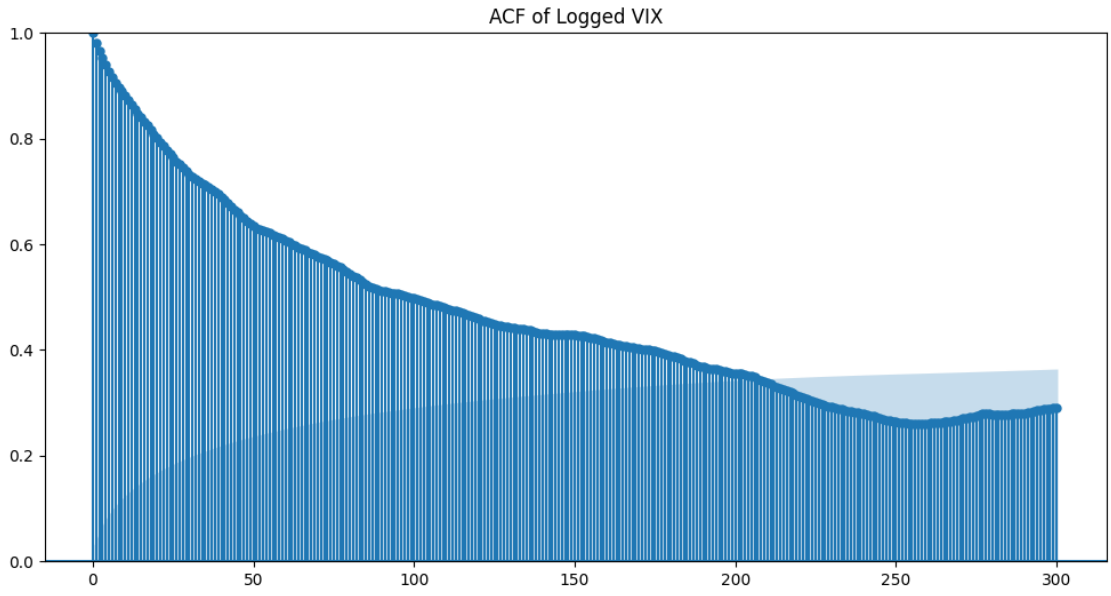
```
vix_df = pd.DataFrame(vix_data, index = vix_data.index)
vix_df['Logged_VIX'] = np.log(vix_df['Close'])
vix_df
```

4280 rows × 2 columns

```
from statsmodels.graphics.tsaplots import plot_acf
plt.figure(figsize=(12, 6))
plt.plot(vix_df['Logged_VIX'], label='Logged VIX')
plt.title('Logged VIX Time Series')
plt.xlabel('Date')
plt.ylabel('Logged VIX')
plt.legend()
plt.show()

# Plotting the ACF of logged VIX values
fig, ax = plt.subplots(figsize=(12, 6))
plot_acf(vix_df['Logged_VIX'], lags = 300, ax=ax)
ax.set_ylim(0, 1)
plt.title('ACF of Logged VIX')
plt.show()
```



png



png

```
result = adfuller(vix_df['Logged_VIX'])

print('ADF Statistic:', result[0])
print('p-value:', result[1])
if result[1] < 0.05:
    print("Reject the null hypothesis: the series is stationary.")
else:
    print("Fail to reject the null hypothesis: the series has a unit root.")
```

```
ADF Statistic: -4.524404196002887
p-value: 0.00017760960542141253
Reject the null hypothesis: the series is stationary.
```

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.tsa.arima.model import ARIMA

# Assuming your time series data is in a pandas Series named `data`
# Replace `data` with the name of your series.

# Set up range for AR and MA orders to test
p_values = range(0, 5)  # Adjust range as needed
q_values = range(0, 5)  # Adjust range as needed

# Initialize variables to store the best model information
best_aic = np.inf
best_order = None
best_model = None

# Grid search over p, d, q combinations for ARMA models
for p in p_values:
    for q in q_values:
        try:
            model = ARIMA(vix_df['Logged_VIX'], order=(p, 0, q))
            model_fit = model.fit()
            aic = model_fit.aic
            print(f"ARMA model order: {(p, 0, q)}")
            print(f"ARMA AIC: {aic}")
            if aic < best_aic:
                best_aic = aic
                best_order = (p, 0, q)
                best_model = model_fit
        except Exception as e:
            print(f"ARMA({p},0,{q}) model failed to fit: {e}")
            continue

print(f"Best ARMA model order: {best_order}")
print(f"Best AIC: {best_aic}")
print(best_model.summary())
```

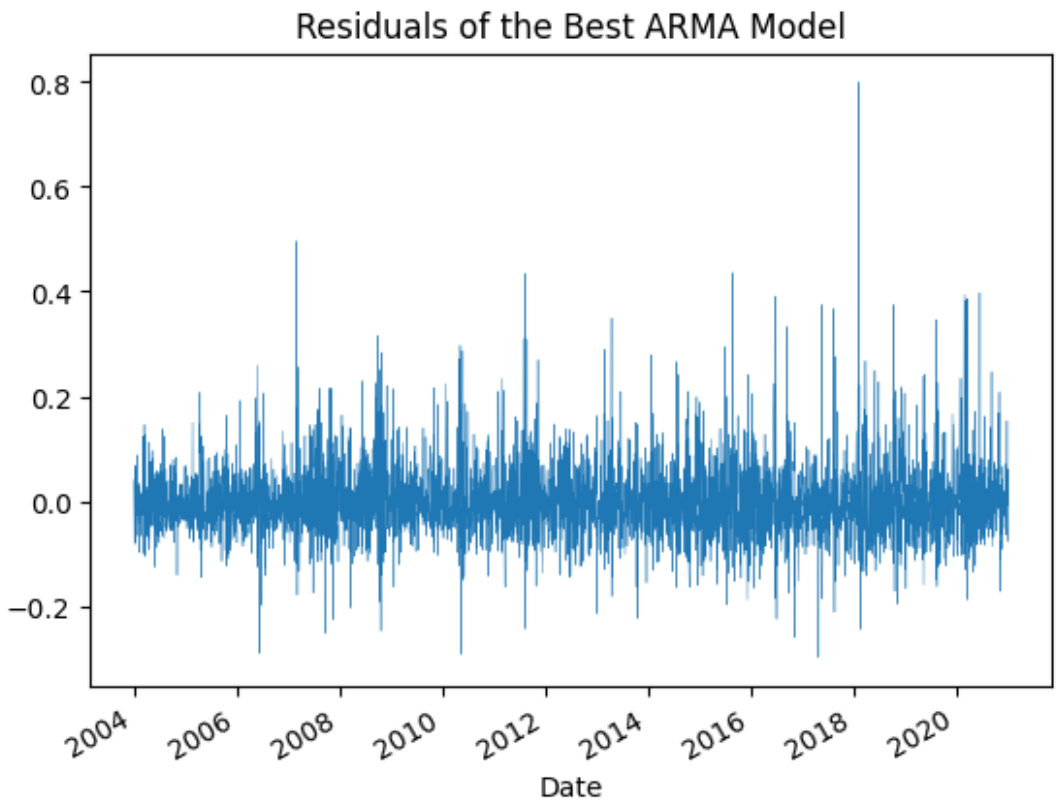
```
ARMA model order: (0, 0, 0)
ARMA AIC: 3873.0346350376176
ARMA model order: (0, 0, 1)
ARMA AIC: -894.5829174985115
ARMA model order: (0, 0, 2)
ARMA AIC: -3765.4075521913364
ARMA model order: (0, 0, 3)
ARMA AIC: -5554.4189356296065
```

```
ARMA model order: (0, 0, 4)
ARMA AIC: -6673.141345624118
ARMA model order: (1, 0, 0)
ARMA AIC: -10200.10503534276
ARMA model order: (1, 0, 1)
ARMA AIC: -10233.213261684376
ARMA model order: (1, 0, 2)
ARMA AIC: -10237.723023123493
ARMA model order: (1, 0, 3)
ARMA AIC: -10237.508228940002
ARMA model order: (1, 0, 4)
ARMA AIC: -10243.750379463123
ARMA model order: (2, 0, 0)
ARMA AIC: -10230.37087591475
ARMA model order: (2, 0, 1)
ARMA AIC: -10252.059974133797
ARMA model order: (2, 0, 2)
ARMA AIC: -10250.198966413445
ARMA model order: (2, 0, 3)
ARMA AIC: -10251.38583964701
ARMA model order: (2, 0, 4)
ARMA AIC: -10250.765541218574
ARMA model order: (3, 0, 0)
ARMA AIC: -10235.92589553074
ARMA model order: (3, 0, 1)
ARMA AIC: -10232.616970822735
ARMA model order: (3, 0, 2)
ARMA AIC: -10241.440689339965
ARMA model order: (3, 0, 3)
ARMA AIC: -10227.35948868106
ARMA model order: (3, 0, 4)
ARMA AIC: -10232.436929410294
ARMA model order: (4, 0, 0)
ARMA AIC: -10235.176202254119
ARMA model order: (4, 0, 1)
ARMA AIC: -10232.114544274958
ARMA model order: (4, 0, 2)
ARMA AIC: -10224.947025174362
ARMA model order: (4, 0, 3)
ARMA AIC: -10245.759795973936
ARMA model order: (4, 0, 4)
ARMA AIC: -10227.977386258282
Best ARMA model order: (2, 0, 1)
Best AIC: -10252.059974133797
                               SARIMAX Results
==============================================================================
Dep. Variable:             Logged_VIX   No. Observations:                 4278
Model:                 ARIMA(2, 0, 1)   Log Likelihood                5131.030
Date:                Tue, 19 Nov 2024   AIC                         -10252.060
Time:                        11:28:30   BIC                         -10220.254
```

```
Sample:                             0   HQIC                       -10240.824
                                - 4278
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.8591      0.099     28.911      0.000       2.665       3.053
ar.L1          1.7048      0.044     38.952      0.000       1.619       1.791
ar.L2         -0.7074      0.043    -16.438      0.000      -0.792      -0.623
ma.L1         -0.8018      0.038    -21.090      0.000      -0.876      -0.727
sigma2         0.0053   6.11e-05     86.918      0.000       0.005       0.005
==============================================================================
=====
Ljung-Box (L1) (Q):                   1.04   Jarque-Bera (JB):             115
13.57
Prob(Q):                              0.31   Prob(JB):
0.00
Heteroskedasticity (H):               1.71   Skew:
1.37
Prob(H) (two-sided):                  0.00   Kurtosis:
10.55
==============================================================================
=====

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex
-step).
```

```
best_model.resid.plot(title="Residuals of the Best ARMA Model", linewidth=0.3
)
```

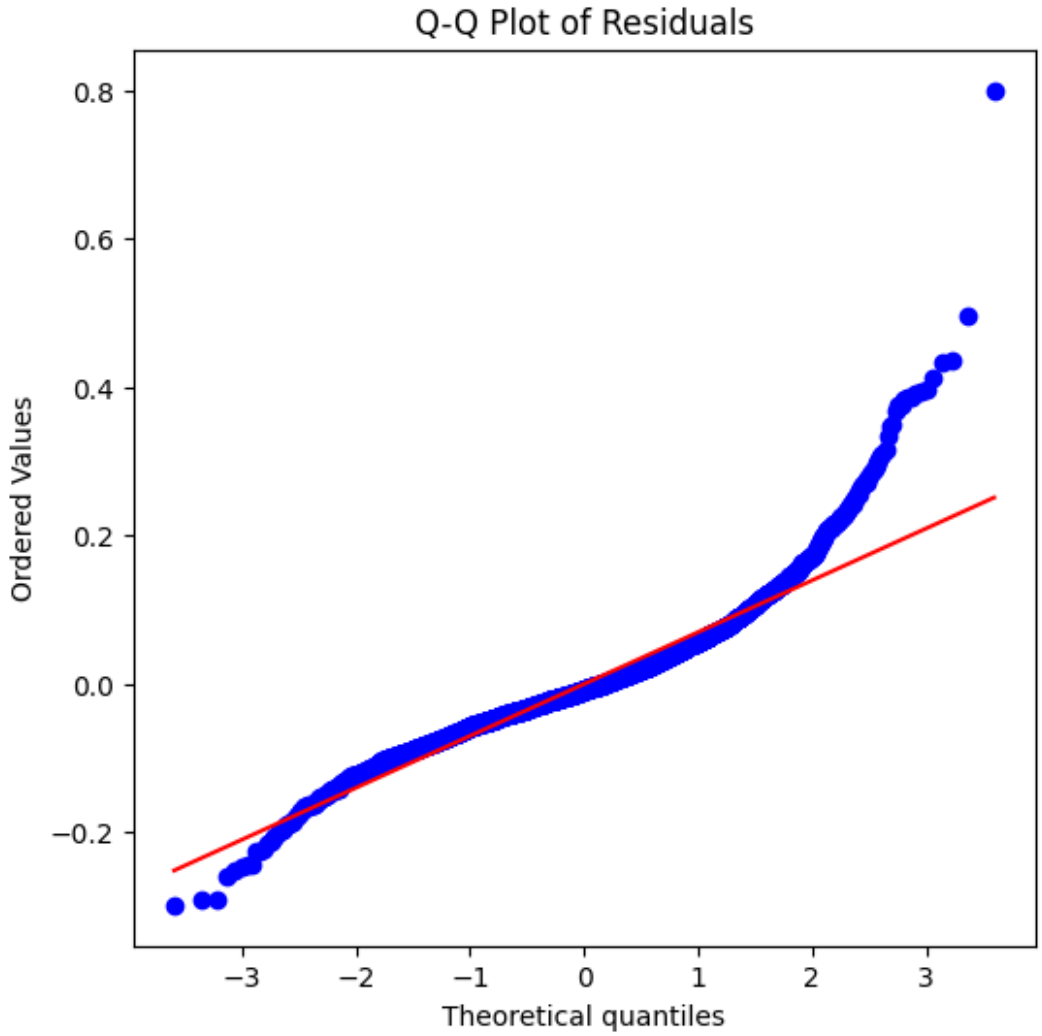
```
<Axes: title={'center': 'Residuals of the Best ARMA Model'}, xlabel='Date'>
```



png

```
residuals = best_model.resid

plt.figure(figsize=(6, 6))
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Q-Q Plot of Residuals')
plt.show()

# Shapiro-Wilk Test (tests for normality)
shapiro_stat, shapiro_p_value = stats.shapiro(residuals)
print(f"Shapiro-Wilk Test: Statistic = {shapiro_stat:.4f}, p-value = {shapiro
_p_value:.4f}")

# Jarque-Bera Test (tests for skewness and kurtosis, deviations from normalit
y)
jb_stat, jb_p_value = stats.jarque_bera(residuals)
print(f"Jarque-Bera Test: Statistic = {jb_stat:.4f}, p-value = {jb_p_value:.4
f}")

# Interpretation of p-values from tests:
alpha = 0.05
print("\nNormality Test Results:")
if shapiro_p_value > alpha:
    print("Shapiro-Wilk: Residuals are normally distributed (fail to reject H
```

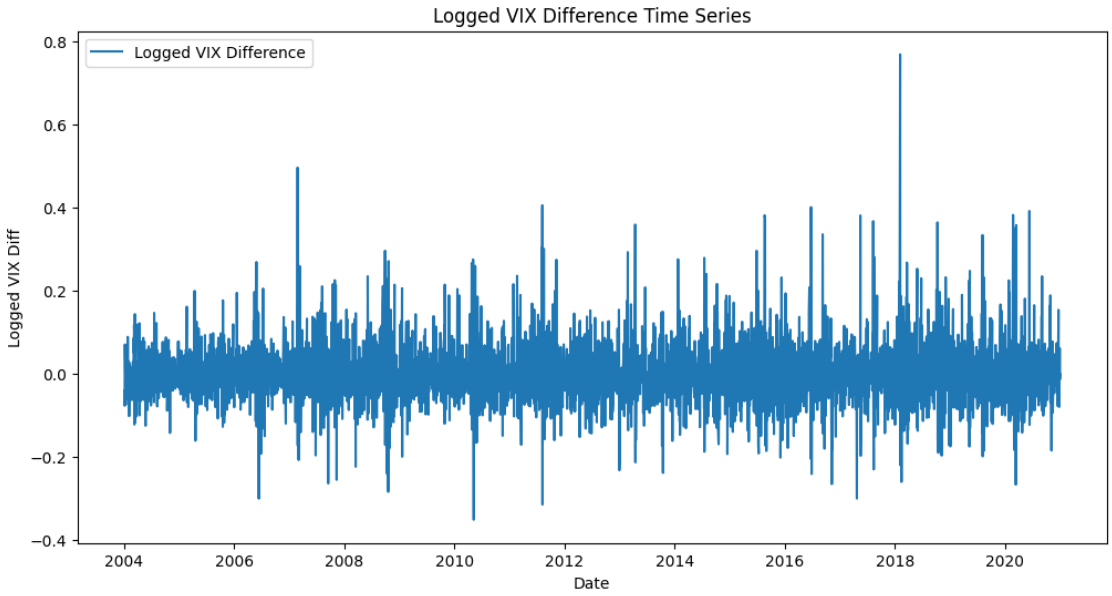
```
0).")
else:
    print("Shapiro-Wilk: Residuals are not normally distributed (reject H0)."
)

if jb_p_value > alpha:
    print("Jarque-Bera: Residuals are normally distributed (fail to reject H0
).")
else:
    print("Jarque-Bera: Residuals are not normally distributed (reject H0).")
```



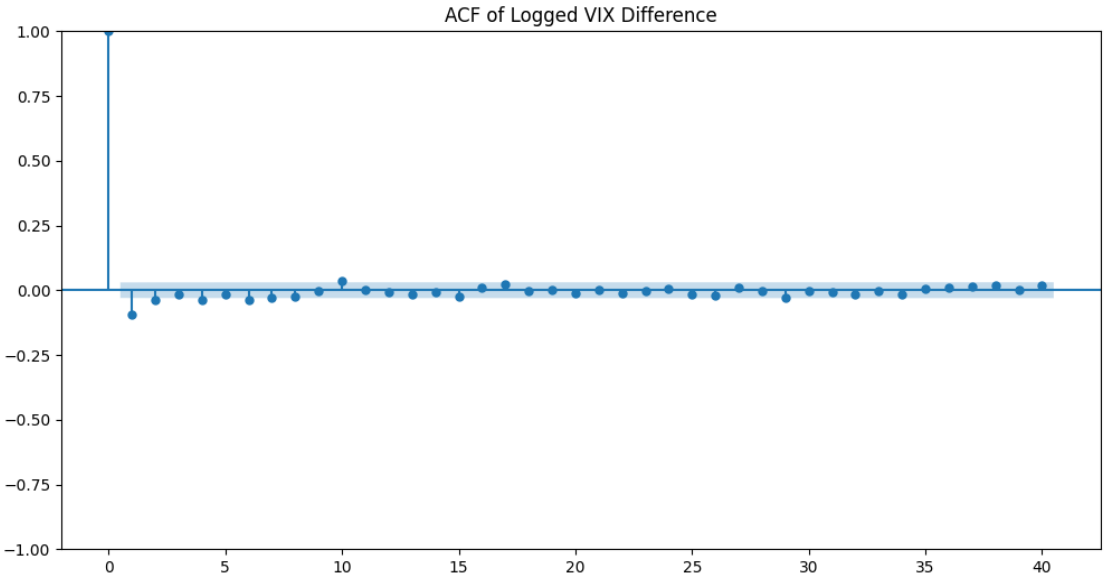
png

```
Shapiro-Wilk Test: Statistic = 0.9188, p-value = 0.0000
Jarque-Bera Test: Statistic = 11722.9742, p-value = 0.0000

Normality Test Results:
Shapiro-Wilk: Residuals are not normally distributed (reject H0).
Jarque-Bera: Residuals are not normally distributed (reject H0).
```

```
vix_df['Logged_VIX_Diff'] = vix_df['Logged_VIX'].diff()
vix_df = vix_df.dropna()

plt.figure(figsize=(12, 6))
plt.plot(vix_df['Logged_VIX_Diff'], label='Logged VIX Difference')
plt.title('Logged VIX Difference Time Series')
plt.xlabel('Date')
plt.ylabel('Logged VIX Diff')
plt.legend()
plt.show()

# Plotting the ACF of logged VIX values
fig, ax = plt.subplots(figsize=(12, 6))
plot_acf(vix_df['Logged_VIX_Diff'], lags = 40, ax=ax)
plt.title('ACF of Logged VIX Difference')
plt.show()
```



png



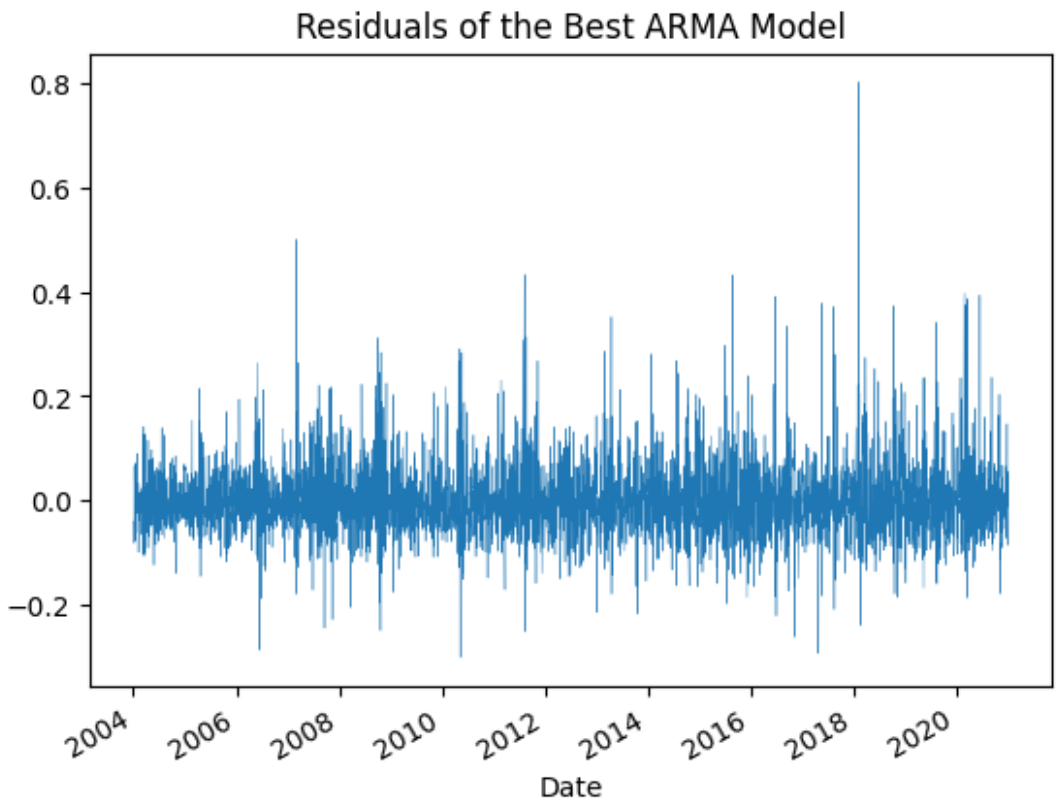
png

```
p_values = range(0, 5)  # Adjust range as needed
q_values = range(0, 5)
i_values = range(0, 2) # Adjust range as needed

# Initialize variables to store the best model information
best_aic = np.inf
best_order = None
best_model = None

# Grid search over p, d, q combinations for ARMA models
for p in p_values:
    for q in q_values:
        for i in i_values:
            try:
                model = ARIMA(vix_df['Logged_VIX_Diff'], order=(p, i, q))
                model_fit = model.fit()
                aic = model_fit.aic
                if aic < best_aic:
                    best_aic = aic
                    best_order = (p, i, q)
                    best_model = model_fit
            except Exception as e:
                print(f"ARMA({p},{i},{q}) model failed to fit: {e}")
                continue

print(f"Best ARMA model order: {best_order}")
print(f"Best AIC: {best_aic}")
print(best_model.summary())
```

```
Best ARMA model order: (3, 0, 1)
Best AIC: -10250.219863786444
                               SARIMAX Results
==============================================================================
Dep. Variable:        Logged_VIX_Diff   No. Observations:                 4279
Model:                 ARIMA(3, 0, 1)   Log Likelihood                5131.110
Date:                Tue, 19 Nov 2024   AIC                         -10250.220
Time:                        11:15:34   BIC                         -10212.051
Sample:                             0   HQIC                        -10236.736
                                - 4279
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       9.032e-05      0.000      0.212      0.832      -0.001       0.001
ar.L1          0.8749      0.012     73.876      0.000       0.852       0.898
ar.L2          0.0437      0.014      3.032      0.002       0.015       0.072
ar.L3          0.0299      0.012      2.573      0.010       0.007       0.053
ma.L1         -0.9838      0.006   -172.663      0.000      -0.995      -0.973
sigma2         0.0053   6.11e-05     87.445      0.000       0.005       0.005
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               116
26.50
Prob(Q):                              0.92   Prob(JB):
0.00
Heteroskedasticity (H):               1.71   Skew:
1.36
Prob(H) (two-sided):                  0.00   Kurtosis:
10.60
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex
-step).
```

```
best_model.resid.plot(title="Residuals of the Best ARMA Model", linewidth=0.3
)
```

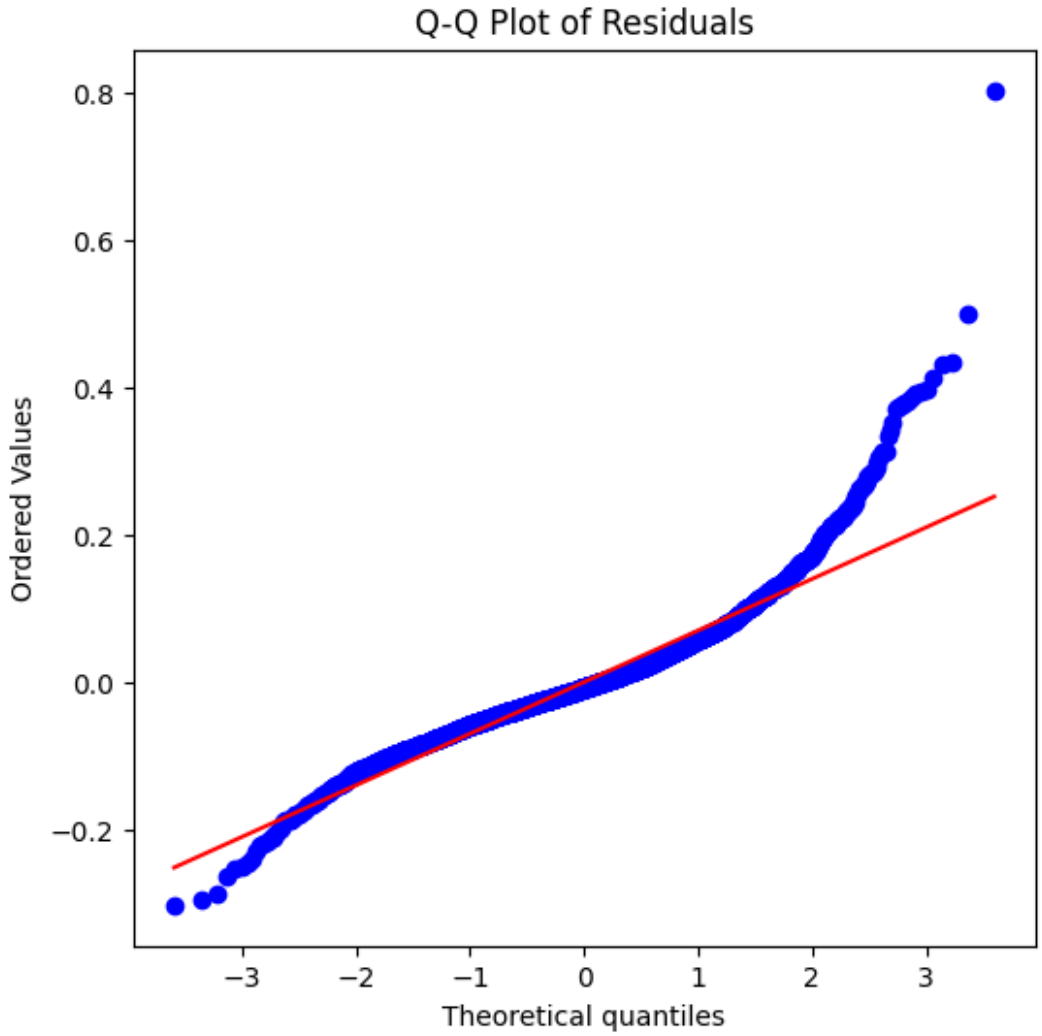
```
<Axes: title={'center': 'Residuals of the Best ARMA Model'}, xlabel='Date'>
```



png

```
residuals = best_model.resid

plt.figure(figsize=(6, 6))
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Q-Q Plot of Residuals')
plt.show()

# Shapiro-Wilk Test (tests for normality)
shapiro_stat, shapiro_p_value = stats.shapiro(residuals)
print(f"Shapiro-Wilk Test: Statistic = {shapiro_stat:.4f}, p-value = {shapiro
_p_value:.4f}")

# Jarque-Bera Test (tests for skewness and kurtosis, deviations from normalit
y)
jb_stat, jb_p_value = stats.jarque_bera(residuals)
print(f"Jarque-Bera Test: Statistic = {jb_stat:.4f}, p-value = {jb_p_value:.4
f}")

# Interpretation of p-values from tests:
alpha = 0.05
print("\nNormality Test Results:")
if shapiro_p_value > alpha:
    print("Shapiro-Wilk: Residuals are normally distributed (fail to reject H
```

```
0).")
else:
    print("Shapiro-Wilk: Residuals are not normally distributed (reject H0)."
)

if jb_p_value > alpha:
    print("Jarque-Bera: Residuals are normally distributed (fail to reject H0
).")
else:
    print("Jarque-Bera: Residuals are not normally distributed (reject H0).")
```



png

```
Shapiro-Wilk Test: Statistic = 0.9199, p-value = 0.0000
Jarque-Bera Test: Statistic = 11624.7841, p-value = 0.0000

Normality Test Results:
Shapiro-Wilk: Residuals are not normally distributed (reject H0).
Jarque-Bera: Residuals are not normally distributed (reject H0).
```

```
result = adfuller(vix_df['Logged_VIX_Diff'])

print('ADF Statistic:', result[0])
print('p-value:', result[1])
if result[1] < 0.05:
    print("Reject the null hypothesis: the series is stationary.")
else:
    print("Fail to reject the null hypothesis: the series has a unit root.")
```

```
ADF Statistic: -23.410048252322007
p-value: 0.0
Reject the null hypothesis: the series is stationary.
```

```
vix_df['Logged_VIX_Diff_Lag1'] = vix_df['Logged_VIX_Diff'].shift(1)
vix_df['Logged_VIX_Diff_Lag1_squared'] = vix_df['Logged_VIX_Diff'].shift(1) *
* 2
vix_df = vix_df.dropna()

print(vix_df)
```

```
                Close  Logged_VIX  Logged_VIX_Diff  Logged_VIX_Diff_Lag1  \
Date
2004-01-06  16.730000    2.817203        -0.044426             -0.040891
2004-01-07  15.500000    2.740840        -0.076363             -0.044426
2004-01-08  15.610000    2.747912         0.007072             -0.076363
2004-01-09  16.750000    2.818398         0.070487              0.007072
2004-01-12  16.820000    2.822569         0.004170              0.070487
...               ...         ...              ...                   ...
2020-12-24  21.530001    3.069447        -0.079435             -0.038709
2020-12-28  21.700001    3.077312         0.007865             -0.079435
2020-12-29  23.080000    3.138966         0.061654              0.007865
2020-12-30  22.770000    3.125444        -0.013523              0.061654
2020-12-31  22.750000    3.124565        -0.000879             -0.013523

            Logged_VIX_Diff_Lag1_squared
Date
2004-01-06                      0.001672
2004-01-07                      0.001974
2004-01-08                      0.005831
2004-01-09                      0.000050
2004-01-12                      0.004968
...                                  ...
2020-12-24                      0.001498
2020-12-28                      0.006310
2020-12-29                      0.000062
2020-12-30                      0.003801
2020-12-31                      0.000183

[4278 rows x 5 columns]
```

```
import statsmodels.api as sm
X = vix_df[['Logged_VIX_Diff_Lag1']]
X = sm.add_constant(X)
Y = vix_df['Logged_VIX_Diff']

model = sm.OLS(Y, X).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        Logged_VIX_Diff   R-squared:                       0.009
Model:                            OLS   Adj. R-squared:                  0.009
Method:                 Least Squares   F-statistic:                     38.97
Date:                Tue, 19 Nov 2024   Prob (F-statistic):           4.71e-10
Time:                        11:15:35   Log-Likelihood:                 5103.9
No. Observations:                4278   AIC:                         -1.020e+04
Df Residuals:                    4276   BIC:                         -1.019e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          6.641e-05      0.001      0.059      0.953      -0.002       0.002
Logged_VIX_Diff_Lag1    -0.0950      0.015     -6.243      0.000      -0.125      -0.065
==============================================================================
Omnibus:                     1207.322   Durbin-Watson:                   2.009
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             9488.258
Skew:                           1.125   Prob(JB):                         0.00
Kurtosis:                       9.940   Cond. No.                         13.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```
X = vix_df[['Logged_VIX_Diff_Lag1_squared']]
X = sm.add_constant(X)
Y = vix_df['Logged_VIX_Diff']

model = sm.OLS(Y, X).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        Logged_VIX_Diff   R-squared:                       0.006
```

```
Model:                            OLS   Adj. R-squared:                  0.006
Method:                 Least Squares   F-statistic:                     26.37
Date:                Tue, 19 Nov 2024   Prob (F-statistic):           2.95e-07
Time:                        11:15:35   Log-Likelihood:                 5097.6
No. Observations:                4278   AIC:                        -1.019e+04
Df Residuals:                    4276   BIC:                        -1.018e+04
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                                 coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
const                            0.0020      0.001      1.720      0.086
-0.000       0.004
Logged_VIX_Diff_Lag1_squared    -0.3647      0.071     -5.135      0.000
-0.504      -0.225
==============================================================================
Omnibus:                     1226.656   Durbin-Watson:                   2.140
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             9569.052
Skew:                           1.149   Prob(JB):                         0.00
Kurtosis:                       9.957   Cond. No.                         63.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correct
ly specified.
```

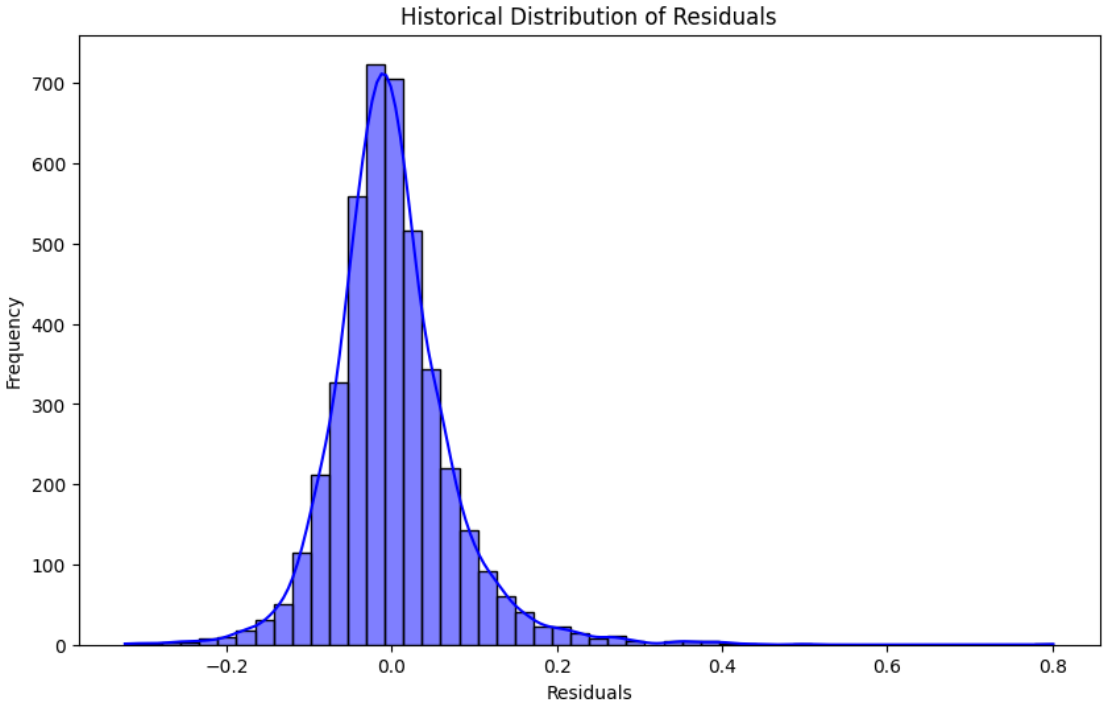
```
X = vix_df[['Logged_VIX_Diff_Lag1', 'Logged_VIX_Diff_Lag1_squared']]
X = sm.add_constant(X)
Y = vix_df['Logged_VIX_Diff']

model = sm.OLS(Y, X).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        Logged_VIX_Diff   R-squared:                       0.011
Model:                            OLS   Adj. R-squared:                  0.011
Method:                 Least Squares   F-statistic:                     24.41
Date:                Tue, 19 Nov 2024   Prob (F-statistic):           2.89e-11
Time:                        11:15:35   Log-Likelihood:                 5108.7
No. Observations:                4278   AIC:                        -1.021e+04
Df Residuals:                    4275   BIC:                        -1.019e+04
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
                                 coef    std err          t      P>|t|
```

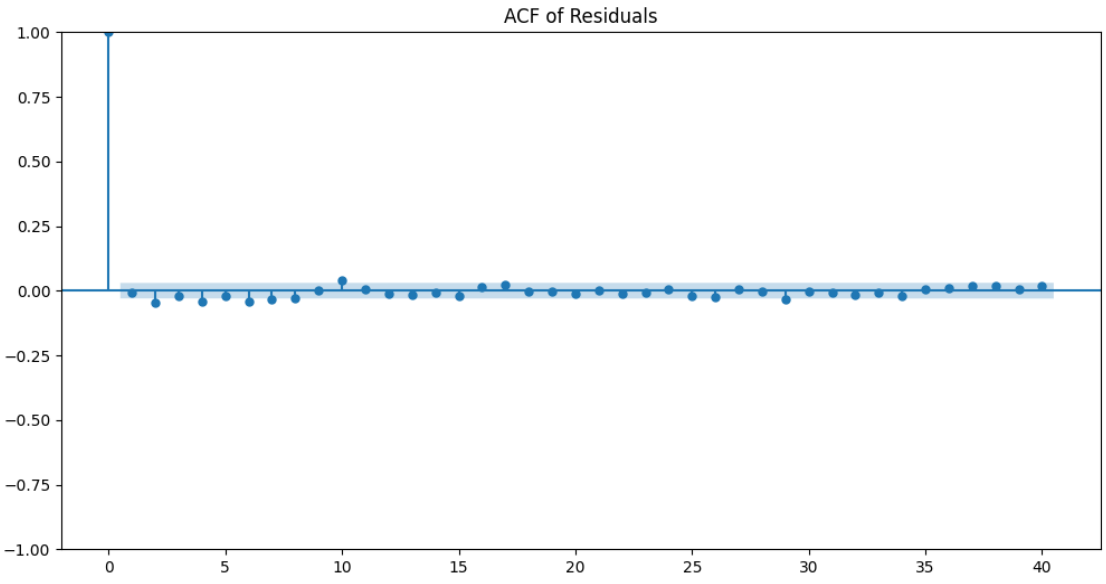
```
[0.025      0.975]
------------------------------------------------------------------------------
------------------
const                          0.0014      0.001      1.133      0.257
-0.001       0.004
Logged_VIX_Diff_Lag1           -0.0769      0.016     -4.724      0.000
-0.109      -0.045
Logged_VIX_Diff_Lag1_squared   -0.2369      0.076     -3.123      0.002
-0.386      -0.088
==============================================================================
Omnibus:                     1262.422   Durbin-Watson:                   2.011
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            10222.893
Skew:                           1.177   Prob(JB):                         0.00
Kurtosis:                      10.198   Cond. No.                         67.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correct
ly specified.
```

Assigment 2 Part 3:

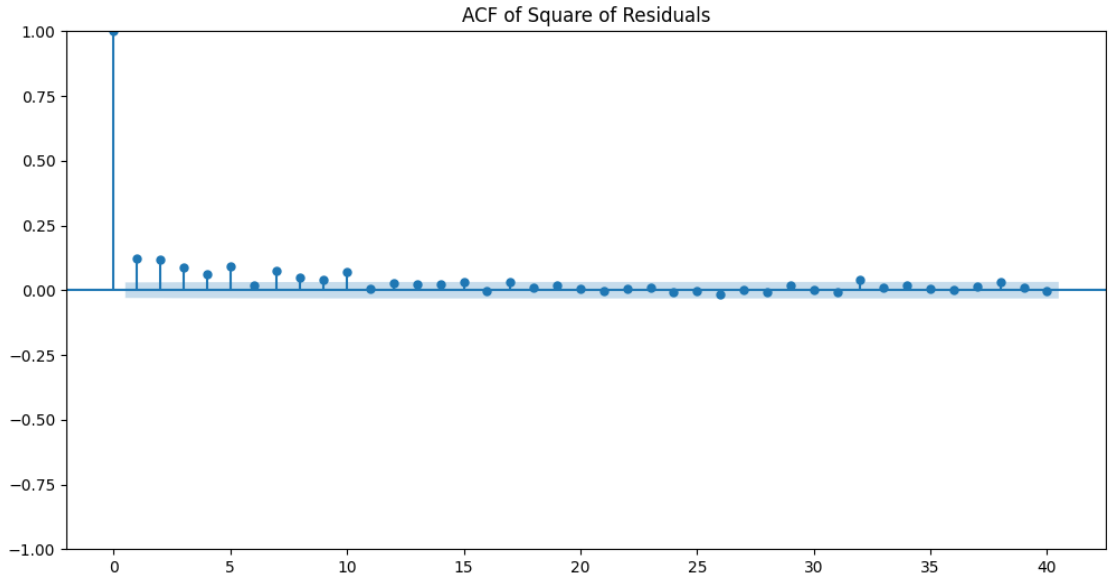
```python
residuals = model.resid
plt.figure(figsize=(10, 6))
sns.histplot(residuals, kde=True, bins=50, color='blue')
plt.title("Historical Distribution of Residuals")
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.show()

# Residuals
fig, ax = plt.subplots(figsize=(12, 6))
plot_acf(residuals, lags = 40, ax=ax)
plt.title('ACF of Residuals')
plt.show()

# Residuals Squared
residuals_squared = residuals**2
fig, ax = plt.subplots(figsize=(12, 6))
plot_acf(residuals_squared, lags = 40, ax=ax)
plt.title('ACF of Square of Residuals')
plt.show()
```


Historical Distribution of Residuals
Frequency
Residuals
0
100
200
300
400
500
600
700
−0.2
0.0
0.2
0.4
0.6
0.8

png


ACF of Residuals
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35
40

png

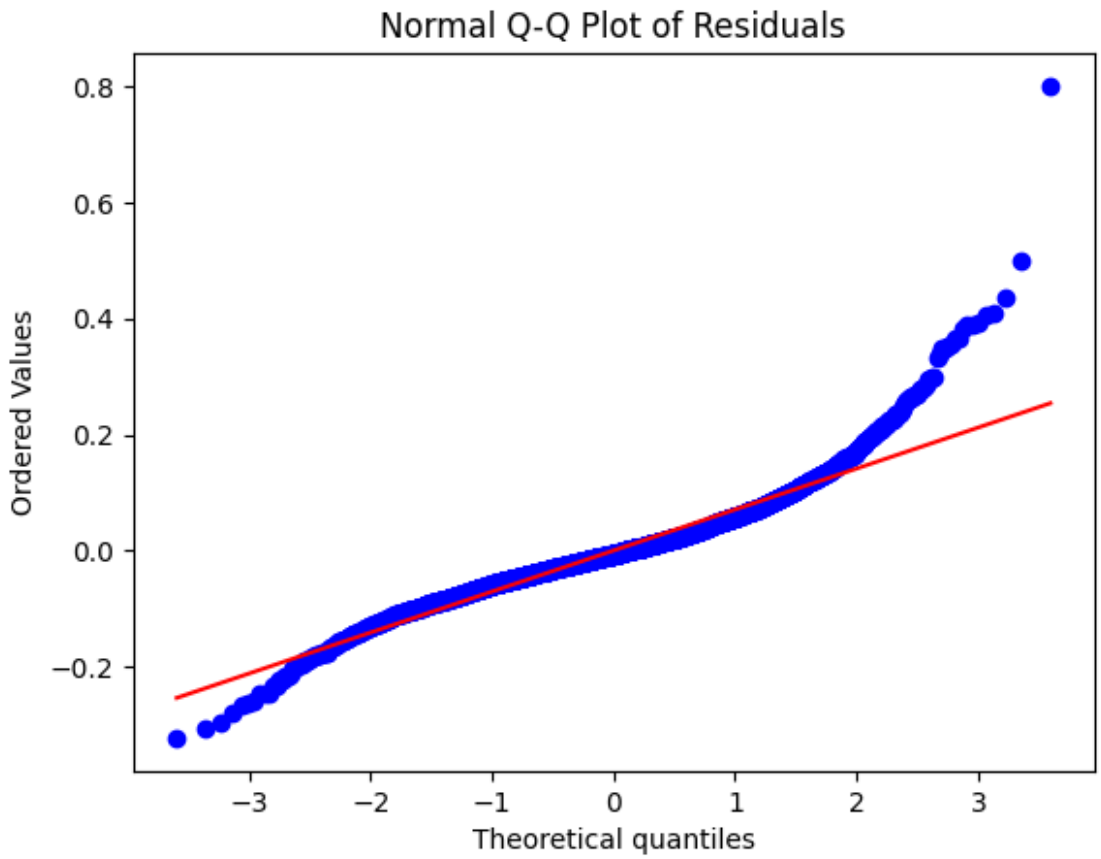


png

```
mean_vix = np.mean(residuals)
variance_vix = np.var(residuals)
skewness_vix = stats.skew(residuals)
kurtosis_vix = stats.kurtosis(residuals)

print(f"Mean: {mean_vix}")
print(f"Variance: {variance_vix}")
print(f"Skewness: {skewness_vix}")
print(f"Kurtosis: {kurtosis_vix}")
```

```
Mean: -1.3365237439970912e-18
Variance: 0.005373602405539862
Skewness: 1.1769693519844666
Kurtosis: 7.197944394772065
```

```
stats.probplot(residuals, dist="norm", plot=plt)
plt.title("Normal Q-Q Plot of Residuals")
plt.show()
```



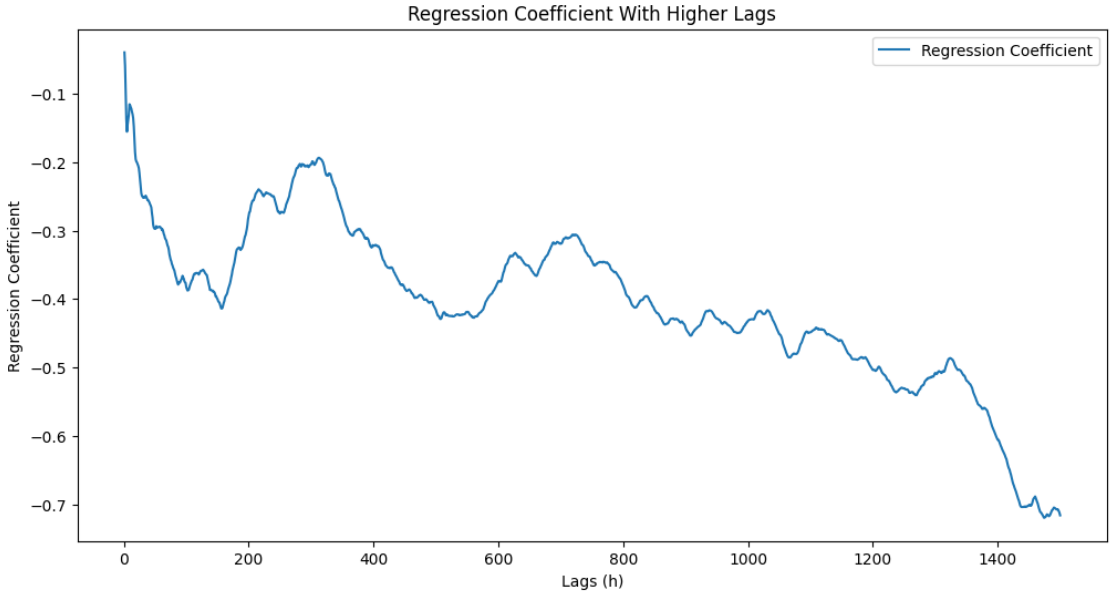
png

```
final = pd.DataFrame()
h = 1500
future_cols = []
lagged_cols = []
df = pd.DataFrame(vix_df[['Logged_VIX_Diff']], index = vix_df.index)

for i in range(1,h+1):
    df[f'Logged_VIX_Future{i}'] = vix_df['Logged_VIX_Diff'].shift(-i)
    future_cols.append(f'Logged_VIX_Future{i}')
    df[f'Logged_VIX_Lag{i}']= vix_df['Logged_VIX_Diff'].shift(i)
    lagged_cols.append(f'Logged_VIX_Lag{i}')
    df = df.dropna()

    df['Logged_VIX_Future_Sum'] = df[future_cols].sum(axis=1)
    df['Logged_VIX_Lagged_Sum'] = df[lagged_cols].sum(axis=1)

    X = df[['Logged_VIX_Lagged_Sum']]
    X = sm.add_constant(X)
    Y = df['Logged_VIX_Future_Sum']

    model = sm.OLS(Y, X).fit()
    intercept, slope = model.params
```

```
    final.loc[i,'Slope'] = slope
    final.loc[i,'R_Squared'] = model.rsquared
    final.loc[i, 'Model_P_Value'] = model.f_pvalue
```

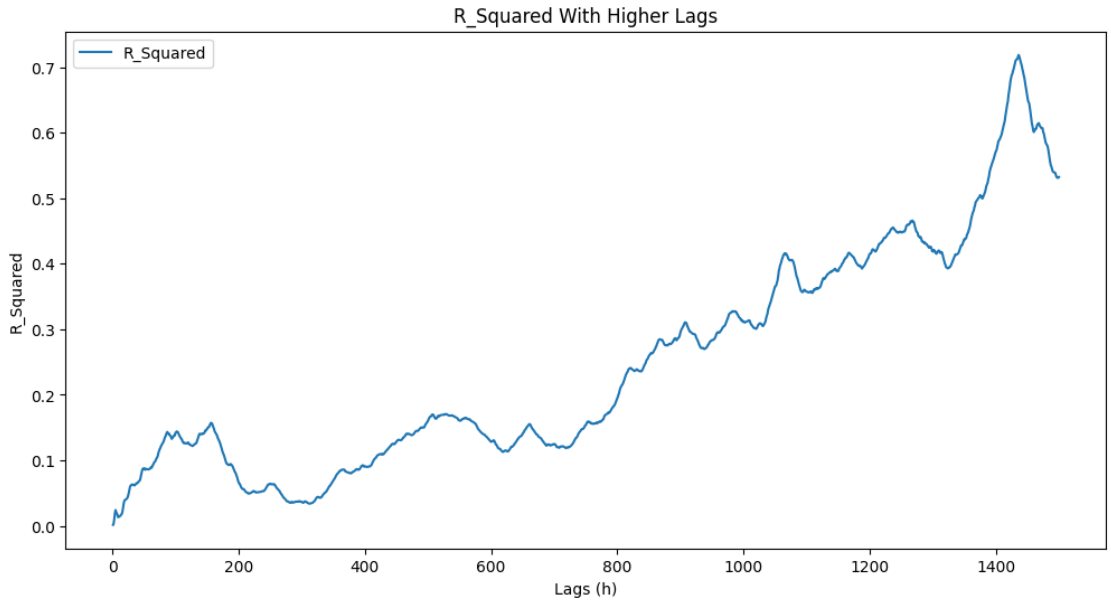
1500 rows × 3 columns

```
plt.figure(figsize=(12, 6))
plt.plot(final['Slope'], label='Regression Coefficient')
plt.title('Regression Coefficient With Higher Lags')
plt.xlabel('Lags (h)')
plt.ylabel('Regression Coefficient')
plt.legend()
plt.show()
```



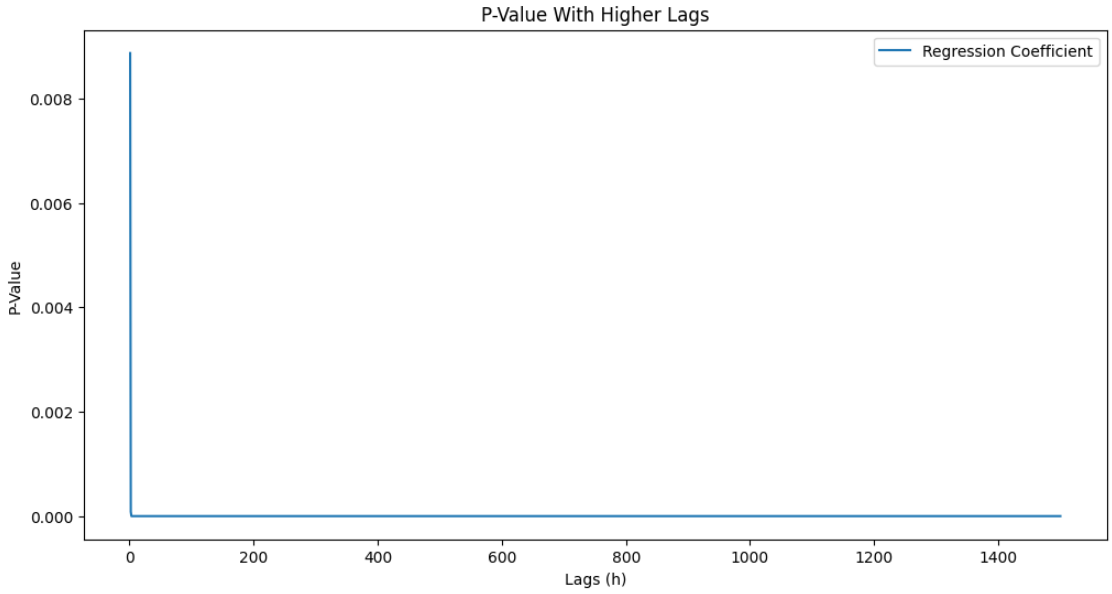
png

```
plt.figure(figsize=(12, 6))
plt.plot(final['R_Squared'], label='R_Squared')
plt.title('R_Squared With Higher Lags')
plt.xlabel('Lags (h)')
plt.ylabel('R_Squared')
plt.legend()
plt.show()
```



png

```
plt.figure(figsize=(12, 6))
plt.plot(final['Model_P_Value'], label='Regression Coefficient')
plt.title('P-Value With Higher Lags')
plt.xlabel('Lags (h)')
plt.ylabel('P-Value')
plt.legend()
plt.show()
```



png