

Diving Into the Deep

How Dark Pools Effect Price Volatility in Financial Markets

FIN11 Term Paper

Candidate: 115 & 163

Executive Summary

This study investigates the relationship between dark pool activity and price variation in financial markets, using an adaptation of Mao Ye's model for price discovery to explore the interaction between opaque and transparent trading venues. Dark pools offer anonymity and discretion for large institutional traders, potentially impacting price discovery and market stability in lit exchanges. Our analysis seeks to determine whether increased dark pool market share correlates with heightened transitory price volatility in lit markets. The dataset, spanning November 2019 to June 2024, incorporates weekly trading volume data across lit and dark pool venues, alongside normalized price variations for multiple assets. Through linear regression models, we explore the role of the *k parameter* as a proxy for dark pool activity. However, diagnostic analyses reveal challenges in meeting the assumptions of ordinary least squares (OLS) regression, including heteroskedasticity and residual autocorrelation, which highlight the complexity of the relationships under study.

The results show a statistically significant but weak correlation between dark pool activity and price variation, with adjusted R^2 values consistently low across all models. While iterations involving log transformations and additional explanatory variables, such as index variance and normalized total volume, improve explanatory power, they still fail to fully account for the multifaceted drivers of price variation. These findings call for further research into non-linear relationships, asset-specific effects through fixed-effects models, and trading simulations to isolate the effects of dark pool activity.

Table of Content

Executive Summary *2*

Table of Figures: *4*

Table of Tables: *4*

1. Introduction *5*

2. Implication of Dark Pools *5*

3. Method *7*
3.1 The Market Model 7
3.2 Price Informativeness 8
3.3 Variable Transformation 10

4. Data *10*
4.1 Data Sampling 10
4.2 Accounting for Dividends 11
4.3 Descriptive Statistics 11

6. Regression Method *14*

7. Results *14*
7.1 Evaluating Model 1 15
7.2 Evaluating Model 2 17
7.3 Evaluating Model 3 19
7.4 Cross-Sectional Regression 19

8. Discussion *21*
8.1 Discussing Results 21
8.2 Data & Model Limitations 21
8.3 Market Participants 22

9. Conclusion *22*

10. Further Research *22*
10.1 Further Model Development 22
10.2 Live Trading Session 23

Bibliografi **24**

Table of Figures:

Figure 1: Correlation matrix of variables depicting relationships across metrics 12
Figure 2: Scatter plots depicting relationships between relPriceVar and explanatory variables before transformation, illustrating non-linear patterns. 13
Figure 3: Scatter plots depicting relationships between relPriceVar and explanatory variables after log-transforming the explanatory variables 13
Figure 4: Scatter plots depicting relationships between relPriceVar and explanatory variables after log-transforming both the response variable and the explanatory variables 13
Figure 5: QQ-Plot for model 1.1 16
Figure 6: Autocorrelation Plot for model 1.1 16
Figure 7: Residual Plot for model 1.1 16
Figure 8: Fitted vs. Actual Plot for model 1.1 16
Figure 9: QQ-Plot for model 1.2 16
Figure 10: Autocorrelation Plot 1.2 16
Figure 11: Residual Plot for model 1.2 16
Figure 12: Fitted vs. Actual Plot for model 1.2 16
Figure 13: QQ-Plot for model 2 18
Figure 14: Histogram of Residuals for model 2 18
Figure 15: Residual Plot for model 2 18
Figure 16: Fitted vs. Actual Plot for model 2 18
Figure 17: QQ-Plot for model 3 18
Figure 18: Histogram of Resdiuals for model 3 18
Figure 19: Residual Plot for model 3 18
Figure 20: Fitted vs. Actual Plot for model 3 18
Figure 21: Dark Volume Coefficient Histogram 20
Figure 22: Index Variance Coefficient Histogram 20
Figure 23: Normalized Total Volume Histogram 20
Figure 24: Correlation matrix for volmue metrics 21

Table of Tables:

Table 1: Regression results for model 1.1 & 1.2 15
Table 2: Regression results for model 2.1 17
Table 3: Regression results for model 3.1 19

1. Introduction

The increasing prevalence of dark pools and other alternative trading systems (ATS) in financial markets has sparked significant debate among academics, regulators, and market participants. Advocates argue that dark pools serve a crucial role by allowing institutional investors, such as pension funds, to execute large trades discreetly, reducing market impact and transaction costs. Parkes, Thorpe, and Li (2015) highlight that dark pools are specifically designed to minimize price impacts and ensure anonymity, making them an efficient tool for institutional trading strategies. However, regulatory concerns highlight the potential downsides of dark pools, including increased volatility and market instability due to the reduction in order book transparency. This issue has drawn attention, particularly in Europe, where regulative directives such as MiFID II and MiFIR have been implemented to address concerns around dark pools. These regulations aimed to reduce information asymmetry, improve transparency, and foster fair competition between traditional exchanges and alternative trading systems (Ntourou & Mallios, 2024). However, despite these efforts, the academic community remains divided on the overall impact of dark pools, with no clear consensus on whether their benefits outweigh their risks.

Since their rise in popularity, dark pools have rapidly expanded, steadily capturing an increasing share of trading activity from traditional lit markets. Their adoption has surged from a modest 4% of market activity in 2005, as reported by the U.S. Securities and Exchange Commission (2014), to over 43% according to Rosenblatt Securities' 2023 estimates (2024). This widespread growth underscores their appeal to market participants and highlights the need to understand how this trend affects market efficiency, price discovery, and volatility dynamics.

In this paper, we build upon the theoretical framework introduced by Ye (2011) to investigate the relationship between dark pool market share and intraweek price volatility. Specifically, we explore how the integration of dark pools into the market mix impacts transitory volatility in lit markets and will discuss whether alternative market structures can mitigate this effect while accommodating large institutional trades. To support our analysis, we utilize a comprehensive dataset covering trading activity from both lit and dark venues for Nasdaq-100 stocks, with weekly aggregated volumes sourced from the SEC and FINRA, alongside daily stock prices from Yahoo Finance. Our findings aim to contribute to the ongoing discourse on balancing market stability with efficiency in modern financial systems.

2. Implication of Dark Pools

The model described by Albert Sidney Kyle (1985) has long been a cornerstone for understanding price discovery in financial markets. In this framework, informed traders, noise traders, and a market maker interact in a single lit market, where the market price gradually incorporates private information held by informed traders. With each trade, half of the private information held by the informed trader is revealed. This process reflects the gradual nature of price discovery, where the market price converges toward the true fundamental value as trades progress. However, the full revelation of

private information theoretically requires an infinite number of trades, as each trade only reveals half of the remaining information. The emergence of crossing networks, often referred to as dark pools, challenges this paradigm by introducing alternative venues where trading occurs without public visibility.

Mao Ye (2011) argues that crossing networks, such as dark pools, fundamentally undermine price discovery. By diverting informed and liquidity trading activity from lit exchanges, crossing networks reduce the information content of observed order flow, resulting in less informative prices. Ye's stance is particularly critical of the adverse effects of dark pools on market efficiency, emphasizing that the reduction in price discovery is especially pronounced for stocks with higher fundamental value uncertainty. More recent research by Buti, Rindi, and Werner (2022) expands the understanding of dark pools, showing that their effects on market quality have evolved over time. In 2020, higher dark pool market share for large-cap stocks led to wider spreads and increased short-term volatility, suggesting that dark trading can exacerbate market instability under certain conditions. These findings highlight that, beyond undermining price discovery, dark pools now also negatively affect key metrics like market liquidity and volatility, particularly in periods of market stress.

This highlights a key trade-off between private benefits and public costs. Dark pools provide significant private benefits to large institutional traders by allowing them to execute large orders discreetly, reducing market impact and transaction costs. However, these benefits often come at the expense of public market efficiency. The diminished transparency in dark pools can impair price formation and reduce the informational efficiency of lit markets, which affects all market participants.

On the contrary, Haoxiang Zhu (2013) offers an alternative perspective, suggesting that dark pools can enhance price discovery under certain conditions. Zhu's model predicts that informed traders are more likely to trade on lit exchanges to avoid execution risk in dark pools, while liquidity traders gravitate toward dark pools for cost-saving purposes. This separation reduces noise in the lit market, enabling the market maker to set prices more effectively. In addition, recent research at the University of Utah and Pennsylvania State University by Brogaard and Pan (2022) supports the notion that dark pools can enhance price discovery under certain conditions. Their findings indicate that increased dark pool activity facilitates greater information acquisition by market participants, measured through stock price dynamics around earnings announcements. However, while these results highlight the potential benefits of dark pools in enhancing informational efficiency, they remain contingent on specific market structures and do not fully resolve broader concerns about transparency and market fairness.

We believe, however, that Zhu's interpretation of the time horizon underestimates the role of information acquisition in real-world trading. Brogaard and Pan demonstrate that informed traders benefit from the ability to act on private information in a more gradual manner through dark pools, which delays immediate market-wide price impacts. This challenges the

assumption that informed traders require instant execution and highlights the importance of dark pools in fostering incremental information incorporation. Therefore, we choose to build upon Ye's market model, which better aligns with these findings.

3. Method

3.1 The Market Model

We use Ye's market model to analyze how the redistribution of trading activity between a lit exchange and a crossing network affects price informativeness. Price discovery, measured through the informativeness of the price e, quantifies how much private information about an asset's fundamental value $\tilde{v}$ is incorporated into its market price $\tilde{p}$ (Ye, 2011, p. 26-28). The measure is defined as:

$$e = 1 - \frac{Var(\tilde{v}|\tilde{p})}{Var(\tilde{v})}$$

This captures the reduction in uncertainty about $\tilde{v}$ due to observing $\tilde{p}$. Using Bayes' rule, the posterior variance of $\tilde{v}$, given the observed price, is expressed as:

$$Var(\tilde{v}|\tilde{p}) = \frac{\sigma_v^2}{1 + \beta^{*2}\left(\frac{\sigma_v^2}{\sigma_e^2}\right)}$$

where σ_v^2 is the variance of the asset's fundamental value, σ_e^2 is the noise variance in the lit market, and β^* represents the informed trader's sensitivity to private information. In the Kyle model, β^* determines the extent to which private information is incorporated into trading volumes, directly influencing price informativeness.

Mao Ye's model modifies this framework by introducing a crossing network, which siphons a portion of informed and liquidity trading activity away from the lit market. This redistribution decreases β^* thereby increasing the posterior variance of $\tilde{v}$ and reducing e. The resulting expression for price informativeness in the presence of a crossing network is:

$$e = 1 - \frac{1}{1+\beta^{*2}\left(\frac{\sigma_v^2}{\sigma_e^2}\right)}$$

To account for the scale of liquidity trading in the crossing network, Ye introduces the parameter $K = \frac{k}{c}$, where k represents the scale of liquidity trading and c captures trading costs. A larger K reflects an increase in the attractiveness of the crossing network relative to the exchange. Substituting K into the model yields:

$$e = 1 - \frac{1}{1 + R(1 - 2\lambda K)}$$

where $R = \frac{\sigma_v^2}{\sigma_e^2}$ measures the ratio of fundamental value uncertainty to noise variance, and λ is the price impact coefficient. Higher values of λ indicate a market more sensitive to order flow, which increases the cost of large trades for informed traders.

Ye's findings emphasize that as K increases, price informativeness e declines due to the reduction in observable informed trading activity on the lit exchange. This decline in price discovery is exacerbated by higher fundamental value uncertainty σ_v^2, which amplifies the importance of private information. The theoretical

implications are as follows: by diverting order flow, crossing networks hinder the lit market's ability to aggregate and reflect private information, thus impairing its role in price discovery.

Ye's critique of crossing networks centers on their broader implications for market efficiency. While these venues offer certain advantages, such as minimizing market impact for large orders, their impact on price formation raises concerns. The model predicts that the lit market becomes increasingly less informative as crossing networks gain market share, challenging the traditional mechanisms of price discovery described in Kyle's framework. This analysis lays the groundwork for assessing whether these theoretical predictions align with real-world market dynamics.

3.2 Price Informativeness

We adopt the same postulation as Ye (2011), which suggests that the introduction of crossing networks (dark pools) reduces price discovery in lit markets by diverting informed and liquidity trading activity, thereby lowering the informativeness of prices. To facilitate this analysis, we will use the model to account for real-world market conditions and the structure of our dataset. This approach will allow us to assess the validity of our postulation under practical scenarios and evaluate its implications for price discovery.

According to Ye, the variance of the fundamental value $Var(\tilde{v})$ can be decomposed into two components: the variance explained by the price $Var(\tilde{v})$ and the variance unexplained by the price $Var(\tilde{v}|\tilde{p})$. This decomposition is expressed as:

$$\mathrm{Var}(\tilde{v}) = \mathrm{Var}(\tilde{\mathrm{p}}) + \mathrm{Var}(\,\tilde{v} \mid \tilde{\mathrm{p}}\,),$$

where a more informative price reduces $\mathrm{Var}(\,\tilde{v} \mid \tilde{p}\,)$ and increases $\mathrm{Var}(\tilde{p})$, reflecting efficient price discovery. However, the introduction of dark pools reduces the amount of informed trading on the exchange, lowering the informativeness of the price. This effect translates into a smaller $\mathrm{Var}(\tilde{p})$, leaving a larger share of the variance in $\tilde{v}$ unexplained.

Ye (2011) derived the following expression for the variance of the price in the presence of a crossing network:

$$\mathrm{Var}(\hat{p}) = \lambda^2 \left(2 - 2\lambda\frac{k}{c}\right)\sigma_e^2,$$

We want to test whether the price informativeness has the same relationship in the real-world data as described in the market model. To do this we take a start with the variance in observed price.

We need to make a few assumptions. Firstly λ representing the price impact reflects how much the market maker reacts to the order flow, and this can change over time and across different assets. We chose however for the simplicity of the model to assume that this is constant for all assets and across time. We represent λ as some arbitrary constant a_1. We rewrite the variance in observed price as.

$$\mathrm{Var}(\hat{p}) = \mathrm{a}_1^2 \left(2 - 2\mathrm{a}_1\frac{k}{c}\right)\sigma_e^2$$

We further look at σ_e^2 that represents the variation in liquidity demand. This, as everything else in the real-world, is dependent on a plethora of variables, we however choose to assume that this

is largely the same, and we choose to substitute it for an arbitrary constant a_2. We can then formulate the price variation as.

$$\mathrm{Var}(\hat{p}) = \mathrm{a}_1^2\left(2 - 2\mathrm{a}_1\frac{k}{c}\right)a_2$$
$$\mathrm{Var}(\hat{p}) = \left(2\mathrm{a}_1^2 - 2\mathrm{a}_1^3\frac{k}{c}\right)a_2$$
$$\mathrm{Var}(\hat{p}) = 2\mathrm{a}_1^2 a_2 - 2\mathrm{a}_1^3 a_2\frac{k}{c}$$

c reflects the trading cost in the crossing network, we assume that this is constant in an equilibrium, such that we substitute it for an arbitrary constant $c = a_3^{-1}$. Such that we have.

$$\mathrm{Var}(\hat{p}) = 2\mathrm{a}_1^2 a_2 - 2\mathrm{a}_1^3 a_2 \mathrm{a}_3 k$$

We can now take a look at k, the scale of liquidity trading in the dark pool. We assume that this scale is given by the relative sizes of the executed volumes in the lit and dark markets.

$$k = \frac{\mu_{Executed}^{Dark}}{\mu_{Executed}^{Lit}}$$

We, much like the market maker, can only observe an aggregated demand, because we do not know who is informed and who is not, in our case we can only observe the weekly aggregated executed volume in the dark pools. We choose to assume that the relationship between total volume in the dark pool is.

$$D_{Executed}^{Dark} = x_{Executed}^{Dark} + \mu_{Executed}^{Dark}$$

And can be described as a linear relationship with some arbitrary scalar $a_4^{-1} \leq 1$ and can therefore be described as.

$$D_{Executed}^{Dark} = a_4^{-1}\mu_{Executed}^{Dark}$$

We are interested in the scale of uninformed liquidity trading, so we invert this equation.

$$\mu_{Executed}^{Dark} = a_4 D_{Executed}^{Dark}$$

For the lit market we also only observe aggregated demand.

$$D_{Executed}^{Lit} = x_{Executed}^{Lit} + \mu_{Executed}^{Lit}$$

We make the same assumption for the lit market as we did for the dark pool, and we introduce a constant $a_5^{-1} \leq 1$.

$$D_{Executed}^{Lit} = a_5^{-1}\mu_{Executed}^{Lit}$$

$$\mu_{Executed}^{Lit} = a_5 D_{Executed}^{Lit}$$

Recall.

$$k = \frac{\mu_{Executed}^{Dark}}{\mu_{Executed}^{Lit}}$$

We substitute for lit and dark demand.

$$k = \frac{a_4 D_{Executed}^{Dark}}{a_5 D_{Executed}^{Lit}}$$

$$k = \frac{a_4}{a_5}\frac{D_{Executed}^{Dark}}{D_{Executed}^{Lit}}$$

Such that we now have.

$$\mathrm{Var}(\hat{p}) = 2\mathrm{a}_1^2 a_2 - 2\mathrm{a}_1^3 a_2 \mathrm{a}_3 \frac{a_4}{a_5}\frac{D_{Executed}^{Dark}}{D_{Executed}^{Lit}}$$

We now introduce the following constants $\beta_0 = 2a_1^2 a_2$ and $\beta_1 = -2\mathrm{a}_1^3 a_2 \mathrm{a}_3 \frac{a_4}{a_5}$ such that we have.

$$Var(\hat{p}) = \beta_0 + \beta_1\frac{D_{Executed}^{Dark}}{D_{Executed}^{Lit}}$$

3.3 Variable Transformation

We want to fit our regression model across different stocks. That leads to the problem of different stock prices. With no transformation in the price of a stock a 10% change will be drastically different between a stock that trades at 100 dollars vs. a stock that trades at 1 dollar. We choose therefore to transform $Std(\hat{p})$ to $\frac{Std(\hat{p})}{\overline{p}}$ for the intraweek time-period with daily observations per stock in our dataset. We choose specifically not to operate with the variance of the returns as we want to examine the uncertainty of the price and not the uncertainty of the change in the price. These measures are inherently different although they have similar characteristics (Zhu H. , 2018). This transformation captures the variance in the observed prices while making it relative to the absolute value of the share such that we can compare these across different assets.

We now have our first regression model.

Model 1:

$$\frac{Std(\hat{p})}{p_0} = \beta_0 + \beta_1 \frac{D^{Dark}_{Executed}}{D^{Lit}_{Executed}}$$

Regression model 1 is at heart what the mathematical model predicts should dictate the variance in the price. We know however that there also exists systematic volatility, e.g. geopolitical risks. We could try to control for this by incorporating the index volatility into the regression model (Chen, 2024). We do this by taking the mean of the normalized price variation of all assets. This does mean that the dependent variable exists in the explanatory variables, although quite abstracted.

Model 2:

$$\frac{Std(\hat{p})}{p_0} = \beta_0 + \beta_1 \frac{D^{Dark}_{Executed}}{D^{Lit}_{Executed}} + \beta_2\{relIndexVar\}$$

We would also like to test a third model. Based on findings by Næs and Skjeltorp (2004), which highlight the relationship between trading volume, participation rates, and volatility, we hypothesize that market events that increase participation rates may lead to higher volatility and executed volume. To further try to control for these market events which spice volatility we include the normalized order flow for the asset. I.e. we take the observed demand for the week $\widehat{D}$ subtract the average order flow for that asset $\overline{D}$ and divide by the standard deviation of the order flow, $Std(D)$.

Model 3:

$$D = D^{Dark} + D^{Lit}$$

$$\frac{Std(\hat{p})}{p_0} = \beta_0 + \beta_1 \frac{D^{Dark}_{Executed}}{D^{Lit}_{Executed}} + \beta_2\{relIndexVar\} + \beta_3 \frac{\widehat{D} - \overline{D}}{Std(D)}$$

4. Data

4.1 Data Sampling

The dataset used for this paper captures trading activity across lit markets and dark pools, alongside stock prices, over the time horizon from November 25, 2019, to the last week of June 2024. The data includes daily and weekly metrics, aggregated and transformed as necessary to facilitate analysis.

Lit market volume, representing the visible order book in lit exchanges, was obtained from the SEC. This dataset includes all publicly traded stocks listed

by the SEC for public access, providing comprehensive information about trading activity in U.S. equity markets. Hidden volume, representing non-displayed trading activity, was also sourced from the SEC within the same data package and shares the same daily frequency. Dark pool trading volume, reported weekly, was retrieved from the FINRA API, which aggregates trading activity across all crossing networks required to report under regulatory obligations (Financial Industry Regulatory Authority, 2024). Stock price data was acquired from Yahoo Finance on a daily basis, offering additional market performance metrics for the securities analyzed.

The primary focus of this analysis is the Nasdaq-100 Index, which comprises 100 of the largest non-financial companies listed on the Nasdaq Stock Market. To construct the dataset, the SEC's publicly available data was used to extract lit and hidden volume information specifically for the stocks within the Nasdaq-100. However, a subset of stocks was excluded due to inconsistencies in reporting standards or unexplained irregularities observed in the data. The excluded stocks include Alphabet (both Class A and Class C), NVIDIA Corporation, Amazon.com, Linde PLC, MercadoLibre, Fortinet, Pinduoduo, ASML Holding, NXP Semiconductors, AstraZeneca, Coca-Cola, GlobalFoundries, and ARM Holdings. These exclusions were made to ensure the reliability and representativeness of the final dataset.

To ensure comparability between datasets, the daily lit market and hidden volume data were aggregated into weekly totals by summing daily volumes for each stock. This transformation was necessary to align with the weekly granularity of the dark pool data provided by FINRA.

The resulting dataset consists of weekly trading data for the remaining stocks within the Nasdaq-100 Index, covering lit markets, hidden volumes, and dark pools. This comprehensive dataset provides a robust foundation for analyzing the interplay between trading activity in different market venues and its implications for price discovery and market quality.

4.2 Accounting for Dividends

When calculating volatility, we account for dividend payouts by adding the dividend amount to the closing price on the dividend payment date. This adjustment ensures the share price reflects the total value, including the dividend (RealVol, 2024). While this temporarily includes some uninvested capital in the share price, we consider this negligible as the effect lasts no more than five days.

4.3 Descriptive Statistics

Before proceeding with our linear regression models, it is important to first gain a better understanding of the data.

4.3.1 Calculating Variables

Normalized Total Volume:

$$\frac{\{TotalVolumeForWeek\ Ticker\} - \{AverageVolumeForWeek\ Ticker\}}{Std(VolumeForWeek\ Ticker)}$$

Log-Transformed k Parameter:

$$\log\left(\frac{D_{Executed}^{Dark}}{D_{Executed}^{Lit}}\right)$$

Log-Transformed Index Variation:

$$\log\left(Std\left({}_1p, {}_2p, \ldots, {}_Np\right)\right)$$

Log-Transformed Relative Price Variation:

$$\log(relPriceVar)$$

4.3.2 Addressing Correlation

We need to be aware of any correlation between our variables as it could impact the regression analysis.

The Correlation Matrix

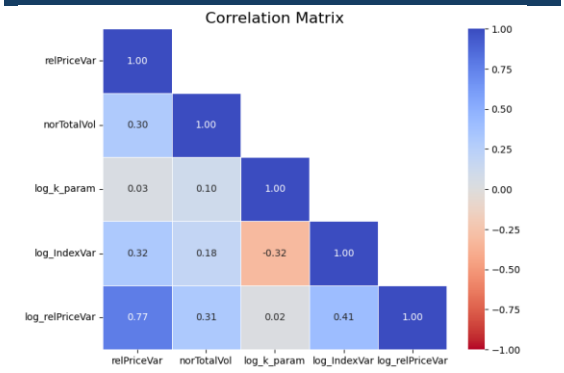


Figure 1: Correlation matrix of variables depicting relationships across metrics

The correlation matrix provides a summary of the relationships between variables in the dataset. Overall, the correlations are relatively low, suggesting that multicollinearity is not an immediate concern. This supports the feasibility of using linear regression for analysis, as high multicollinearity, which can distort coefficient estimates and reduce the interpretability of models, does not appear to be present. However, some aspects of the correlation matrix require further consideration.

One noteworthy observation is the high correlation between *relPriceVar* and *log_relPriceVar*, which shows a strong relationship of 0.77. This is expected, as these variables are transformed aspects of the same concept. Another observation is the negative correlation of -0.32 between *log_k_param* and *log_IndexVar*. Although this is not a particularly strong correlation, it may indicate a potential inverse relationship. The moderate correlation of 0.41 between the log variance of the index (*log_IndexVar*) and the log relative price variance (*log_relPriceVar*) suggests a meaningful connection between systemic market volatility and the relative price variance of individual assets. Notably, *log_relPriceVar* represents the standard deviation of one stock, while *log_IndexVar* represents the standard deviation of the constructed index, which contains the same stock.

It is important to note that correlation does not imply causation. The relationships depicted in the matrix are purely statistical and do not reflect causal mechanisms. While the correlations in the matrix are generally low, the presence of moderate relationships between a few variables suggests the potential for underlying interactions or shared effects that could influence the model's performance.

4.3.3 Evaluating Log-Transformations

This section explores the relationships between the response variable (*relPriceVar*) and the explanatory variables through visual and statistical analysis to evaluate the suitability of log transformations. Scatterplots are used to examine the initial relationships and assess whether log transformations improve linearity and stabilize variance.

Relationships Before Transforming

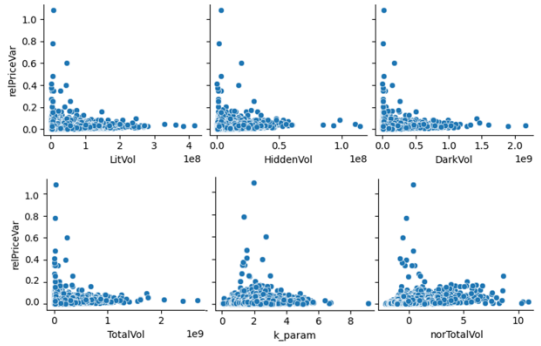


Figure 2: Scatter plots depicting relationships between relPriceVar and explanatory variables before transformation, illustrating non-linear patterns.

Initially, the raw scatter plots reveal clear non-linear patterns, particularly for variables such as *LitVol*, *HiddenVol*, *DarkVol*, and *TotalVol*, where there is a diminishing effect on *relPriceVar* as these volumes increase. This non-linear trend suggests that the raw data might be unsuitable for linear regression. The variable *k_param*, in particular, shows no discernible relationship with *relPriceVar*, indicating its limited relevance as an explanatory variable.

Log-Transforming Explanatory Variables

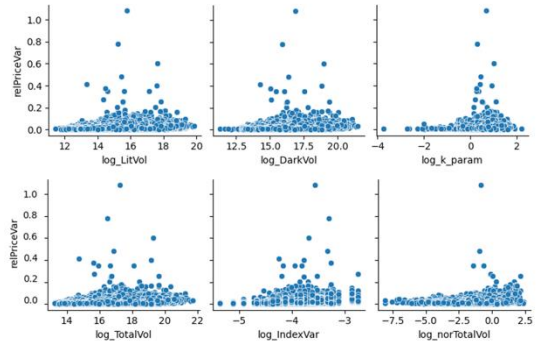


Figure 3: Scatter plots depicting relationships between relPriceVar and explanatory variables after log-transforming the explanatory variables

After log-transforming the explanatory variables, the scatter plots show improved linearity for some variables, such as *log_LitVol* and *log_TotalVol*. However, the patterns observed in variables like *log_IndexVar* and *log_k_param* indicate weak or scattered relationships with *relPriceVar*. In particular, the lack of clear structure for *log_k_param* suggests limited explanatory power, while the clustering in *log_IndexVar* might point to non-linear dynamics or other underlying complexities in the data.

Log-Transforming all Variables

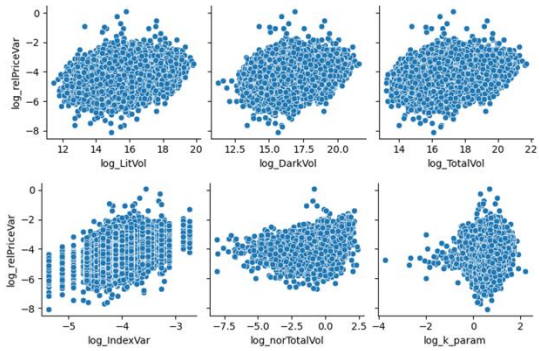


Figure 4: Scatter plots depicting relationships between relPriceVar and explanatory variables after log-transforming both the response variable and the explanatory variables

When both *relPriceVar*, the dependent variable, and the explanatory variables are log-transformed, some relationships, such as those involving *log_LitVol* and *log_TotalVol,* exhibit improved linearity, though noticeable dispersion and outliers remain, potentially affecting regression robustness. The continued lack of discernible structure for *log_k_param* highlights its limited explanatory value, while the patterns in *log_IndexVar* suggest the presence of non-linear dynamics that may not be adequately captured through simple linear modeling. Although log transformations provide some improvements, the observed trends indicate that more advanced analytical methods may be necessary.

The decision to use log-transformed variables, both explanatory and dependent, is based on their ability to improve linear relationships in the data. Logarithmic transformation reduces the

impact of extreme values and scales down non-linear patterns, resulting in more stable and interpretable relationships between variables. By transforming the data, the spread of the variables is compressed, which can mitigate the effect of outliers and facilitate the identification of proportional relationships. Additionally, log transformations often stabilize variance, making the data more suitable for linear regression models. This assumption is supported by the observed improvements in linearity for key variables, such as *log_LitVol* and *log_TotalVol*, after the transformation. While some dispersion remains, the overall trend suggests that log-transformed variables show a more linear relationship.

6. Regression Method

We employed ordinary least squares (OLS) regression to investigate the relationship between price variation and dark pool volume share. This method was chosen for its simplicity and ability to estimate the marginal effect of explanatory variables on the dependent variable under the assumption of linearity. By treating the data as a pooled cross-section, the analysis abstracts away individual stock-level temporal fluctuations, focusing instead on identifying an overarching relationship between the variables.

To validate the stability of our findings, we performed weekly cross-sectional regressions, which allowed us to test the distribution and consistency of coefficients across time. This step was intended to determine whether the relationship between dark pool market share and price variation holds consistently across the sample period. If the weekly coefficients align with the pooled results, it suggests that the relationship is relatively stable and independent of short-term market fluctuations. Significant deviations in weekly coefficients, however, might indicate the presence of time-dependent dynamics or omitted variables, which could limit the robustness of the pooled regression.

The choice to exclude individual uncertainty at the stock level reflects the goal of isolating the relationship between dark pool volume share and price variation in a simplified model. This approach assumes that any inherent variability within individual stocks averages out across the sample. However, we recognize that this assumption may overlook idiosyncratic stock-specific factors that could impact the observed relationships.

While the OLS method provides a straightforward framework for exploring this relationship, its limitations must be acknowledged. The linearity assumption may oversimplify complex dynamics between market volumes and price variation, and the exclusion of certain stocks due to data quality concerns may affect the generalizability of the findings.

7. Results

The results section evaluates and diagnoses the regression models outlined in Chapter 3.3, focusing on their performance, explanatory power, and adherence to key assumptions of OLS regression. Through this analysis, we aim to assess the relationship between dark pool activity and price variation while addressing potential limitations and areas for model refinement.

7.1 Evaluating Model 1

This section evaluates Model 1, which investigates the relationship between dark pool market share *k_param* and relative price variation. Model 1 consists of two versions: 1.1 and 1.2. The distinction between these versions is explained in detail below.

Model 1.1 directly regresses *k_param* on relative price variation, while Model 1.2 uses log-transformed variables to address non-linearities and improve interpretability.

The regression results indicate statistical significance for *k_param* ($p < 0.01$) in both models, however, the explanatory power remains minimal, with adjusted R^2 values of 0.7%. This implies that while *k_param* has a measurable relationship with price variation, it explains only a small fraction of the observed variance, suggesting the presence of other influencing factors beyond dark pool market share.

In Model 1.1, the residual diagnostics highlight several shortcomings. The fitted vs. actual plot reveals limited variability in the predicted values, clustering of actual values near zero, and substantial outliers. Additionally, the residuals show signs of heteroskedasticity, where variance increases with fitted values, violating a key OLS assumption. These issues collectively suggest that Model 1.1 is inadequate in capturing the dynamics between *k_param* and price variation.

Model 1.2, with log-transformed variables, addresses some of these issues. The fitted vs. actual plot demonstrates improved linearity and a more stable variance structure. However, the autocorrelation plot reveals persistent residual autocorrelation, violating the OLS assumption of independent errors. The residual histogram appears approximately normal, but deviations in the Q-Q plot tails indicate potential outliers or non-normality in extreme values. It appears that the transformation has not fully resolved specification issues.

In summary, while Model 1.2 offers marginal improvements, both versions of Model 1 struggle to effectively capture the complexity of the relationship between dark pool market share and price variation. Heteroskedasticity, residual autocorrelation, and low explanatory power limit the utility of these models, indicating that further refinement or alternative methods are needed to better explain the observed dynamics.

Model 1.1: Price Variation vs k parameter
Model 1.2: log Price Variation vs log k parameter

	(1.1)	(1.2)
const	0.014***	-4.314***
	(0.000)	(0.009)
k_param	0.003***	
	(0.000)	
log_k_param		0.195***
		(0.016)
R-squared	0.007	0.007
R-squared Adj.	0.007	0.007
AIC	-100823.352	44656.376
Log Likelihood	50413.676	-22326.188
N	20207	20207

Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01

Table 1: Regression results for model 1.1 & 1.2

Diagnostic Plots for Model 1.1:

QQ-Plot (1.1)

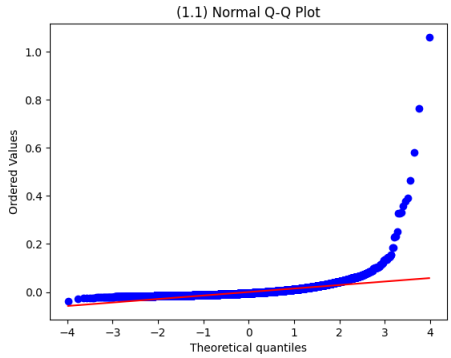


Figure 5: QQ-Plot for model 1.1

Autocorrelation Plot (1.1)

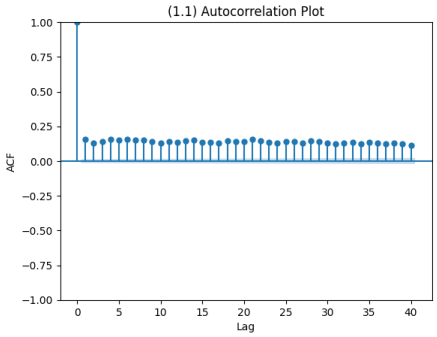


Figure 6: Autocorrelation Plot for model 1.1

Residual Plot (1.1)

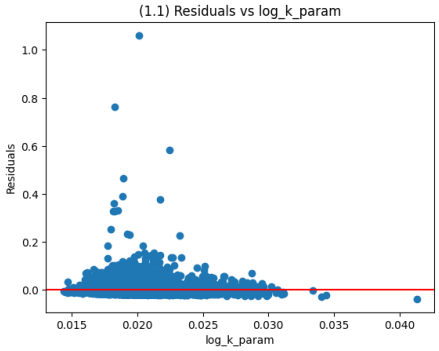


Figure 7: Residual Plot for model 1.1

Fitted vs. Actual Plot (1.1)

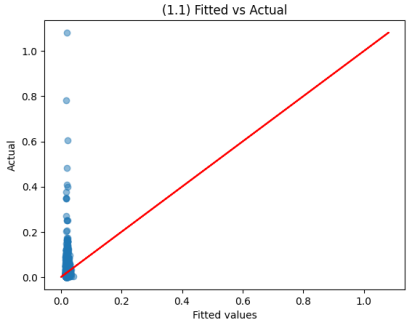


Figure 8: Fitted vs. Actual Plot for model 1.1

Diagnostic Plots for Model 1.2

QQ-Plot (1.2)

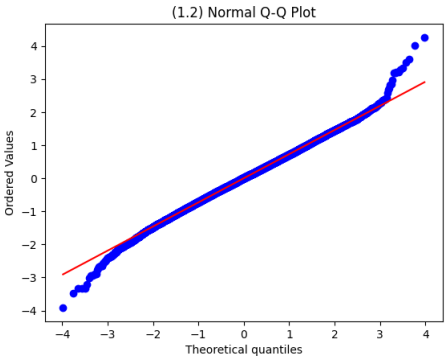


Figure 9: QQ-Plot for model 1.2

Autocorrelation Plot (1.2)

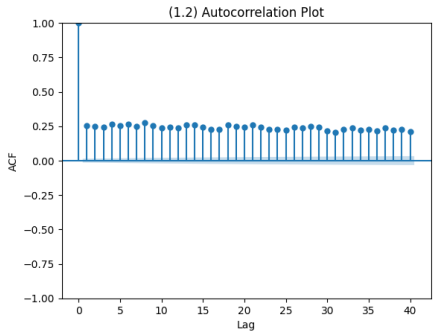


Figure 10: Autocorrelation Plot 1.2

Residual Plot (1.2)

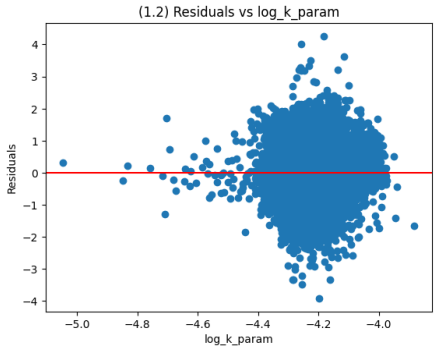


Figure 11: Residual Plot for model 1.2

Fitted vs. Actual Plot (1.2)

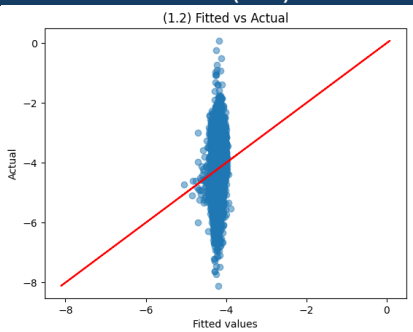


Figure 12: Fitted vs. Actual Plot for model 1.2

7.2 Evaluating Model 2

In diagnosing Model 2 (2.1), we incorporate both the log-transformed k parameter, *log_k_param*, and the log-transformed index variance, *log_IndexVar*, as explanatory variables to assess their relationship with the log-transformed price variation $\log_relPriceVar$. This model corresponds to Model 2 as described in Section 3.3 of the paper.

The results, as presented in Table 2, demonstrate a notable improvement in explanatory power, with the adjusted R^2 increasing to 19.3%. This significant rise suggests that the inclusion of index variance has captured a substantial portion of the variation in price dynamics. However, this improvement must be contextualized with the understanding that the index variance inherently overlaps with the dependent variable due to the shared components of stock movements in the index, introducing a degree of leakage. Despite this, the leakage is considered minimal given the diverse composition of the index.

The regression diagnostics for Model 2.1 provide further insights into the model's adequacy. The residual plot shows a more homogeneous spread, reflecting a reduction in heteroskedasticity compared to previous models. The autocorrelation plot reveals minimal autocorrelation in the residuals, suggesting that this model better adheres to the OLS assumption of residual independence.

The histogram and Q-Q plot of residuals support the assumption of normally distributed errors, with only minor deviations at the tails. These diagnostics suggest that the residuals are reasonably well-behaved, which enhances the validity of the coefficient estimates and their statistical inference. Nonetheless, it is important to recognize that while the model offers a considerable improvement, the remaining unexplained variation and potential non-linear relationships indicate that further refinements may still be necessary.

Overall, Model 2.1 demonstrates the benefits of incorporating additional explanatory variables and transforming the data to stabilize variance and linearize relationships. While not without limitations, this model represents a more robust framework for exploring the dynamics of price variation in relation to dark pool market share and systemic market volatility.

Model 2.1: Log-transformed Price Variation vs. k-parameter and Index Variance

	(2.1)
const	-0.467***
	(0.081)
log_k_param	0.357***
	(0.023)
log_IndexVar	0.939***
	(0.021)
R-squared	0.193
R-squared Adj.	0.193
N	8056
Log Likelihood	-7823.836
AIC	15653.672

Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01

Table 2: Regression results for model 2.1

Diagnostic Plots for Model 2:

QQ-Plot (2.1)

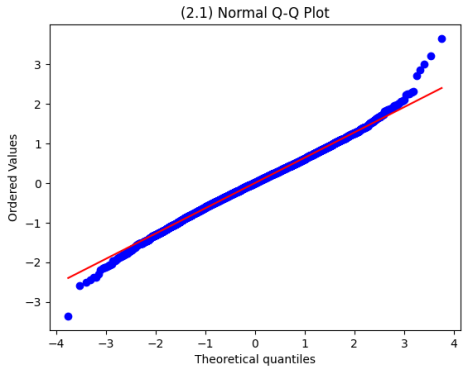


Figure 13: QQ-Plot for model 2

Histogram of Residuals (2.1)

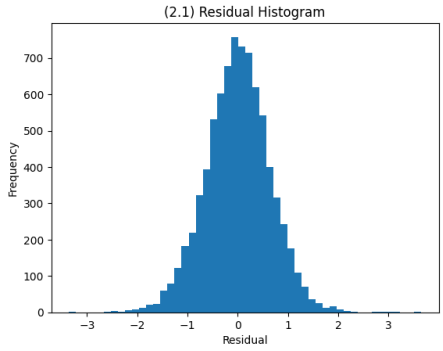


Figure 14: Histogram of Residuals for model 2

Residual Plot (2.1)

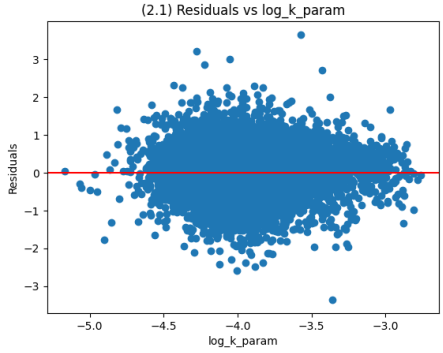


Figure 15: Residual Plot for model 2

Fitted vs. Actual Plot (2.1)

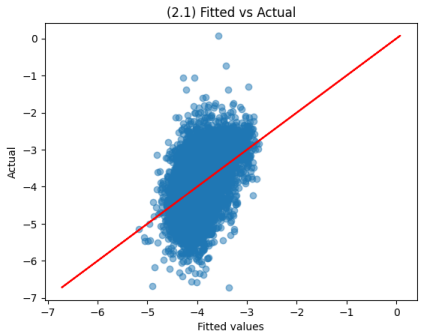


Figure 16: Fitted vs. Actual Plot for model 2

Diagnostic Plots for Model 3:

QQ-Plot (3.1)

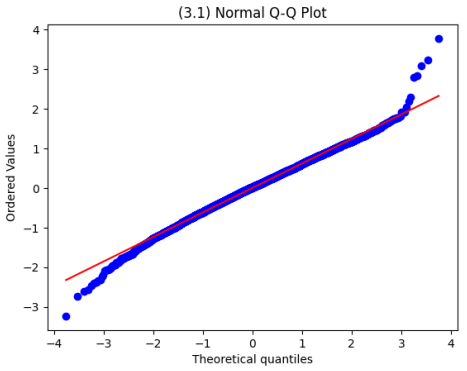


Figure 17: QQ-Plot for model 3

Histogram of Residuals (3.1)

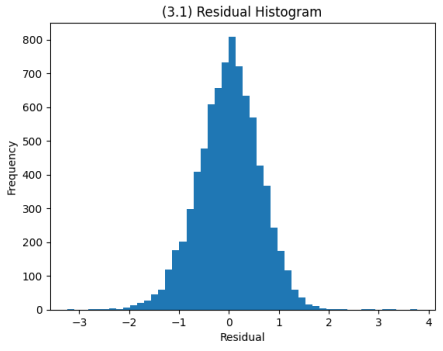


Figure 18: Histogram of Resdiuals for model 3

Residual Plot (3.1)

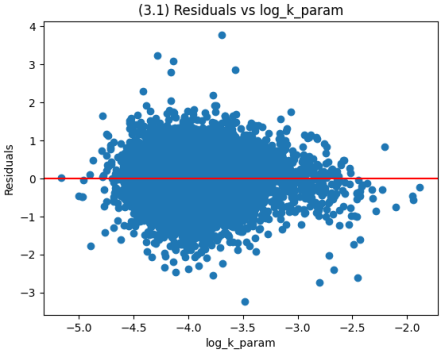


Figure 19: Residual Plot for model 3

Fitted vs. Actual Plot (3.1)

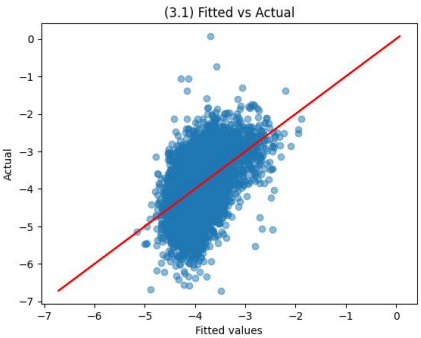


Figure 20: Fitted vs. Actual Plot for model 3

7.3 Evaluating Model 3

In Model 3.1, which includes the normalized total volume as an additional explanatory variable, we observe significant improvements in model performance compared to previous iterations. The R^2 value increases to 24.1%, indicating that this model captures a greater proportion of the variation in the dependent variable, relative price variation. The coefficients for all explanatory variables, including *log_k_param*, *log_IndexVar*, and *norTotalVol*, are statistically significant $(p < 0.01)$, reinforcing their relevance in explaining price variation.

The fitted vs. actual plot shows improved alignment, with less clustering near zero compared to previous models, though some variability remains unaccounted for. The residual plot reveals a more consistent spread, suggesting better model specification, but subtle patterns in the residuals indicate potential heteroskedasticity. The histogram of residuals shows a near-normal distribution, and the Q-Q plot confirms approximate normality, though deviations at the tails highlight potential outliers.

The model demonstrates notable improvements in explanatory power and diagnostic performance; however, it might not fully satisfy the assumptions of OLS regression. Heteroskedasticity remains a concern, as indicated by residual patterns and deviations from normality at the extremes. Despite these issues, we will proceed with the model, recognizing its limitations while acknowledging that further refinement or additional variables could enhance its robustness and predictive capacity.

Model 3.1: Logged Price Variation	
	(3.1)
const	-0.994***
	(0.082)
log_k_param	0.271***
	(0.022)
log_IndexVar	0.829***
	(0.021)
norTotalVol	0.163***
	(0.007)
R-squared	0.241
R-squared Adj.	0.241
N	8056
Log Likelihood	-7577.269
AIC	15162.538

Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01

Table 3: Regression results for model 3.1

7.4 Cross-Sectional Regression

The cross-sectional regression analysis for Model 3.1 provides additional insights into the consistency and distribution of coefficients across the weekly data. By conducting regressions for each time period separately, we can evaluate how stable the coefficients remain over time and identify any patterns or irregularities.

The histograms of the coefficients, figure 21-23, reveal key observations. The coefficient for the dark volume *log_k_param* demonstrates a relatively stable distribution, with most values clustering around a positive range, reinforcing the hypothesized positive relationship between dark pool activity and price variation. However, there is a slight tail extending into the negative range, indicating instances where the relationship between dark pool market share and price variation may reverse. This tail suggests that while the overall relationship holds, there could be temporal or market-specific factors causing deviations.

The coefficients for index variance *log_IndexVar* exhibit a tightly clustered distribution centered around values slightly above 1. This stability aligns with expectations, given that index variance is inherently tied to the variation in individual asset prices due to its aggregate nature.

The normalized total volume *norTotalVol* coefficient shows a slightly broader spread compared to the other variables, with a central clustering around positive values. This broader spread may indicate that the influence of total trading volume on price variation is less consistent, potentially due to variations in market liquidity or trading behavior across different weeks.

One limitation of the analysis is the relatively small number of weekly observations (approximately 250), spanning roughly five years. While this provides a meaningful view of temporal stability, the small sample size introduces variability and limits the statistical precision of the coefficient estimates. Despite this limitation, the overall stability of the coefficients across variables suggests that the relationships identified in Model 3.1 are consistent over time and that the regression model provides a reasonable framework for understanding the drivers of price variation.

Dark Volume Coefficient Histogram

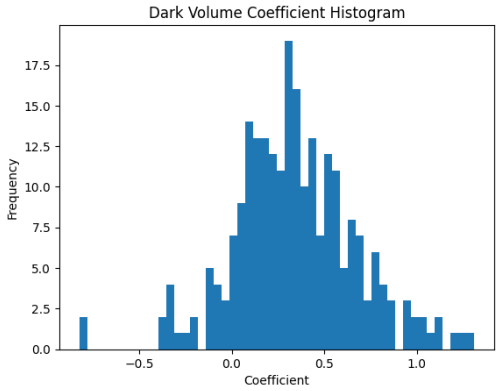


Figure 21: Dark Volume Coefficient Histogram

Index Variance Coefficient Histogram

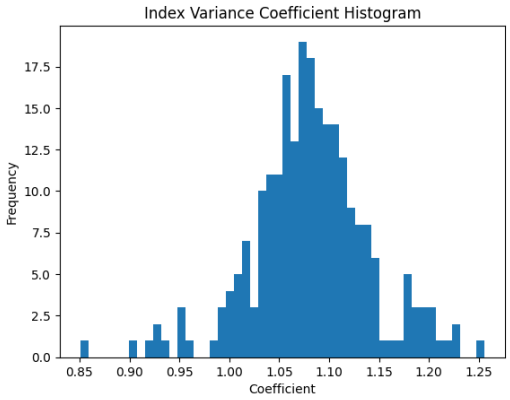


Figure 22: Index Variance Coefficient Histogram

Normalized Total Volume Coefficient

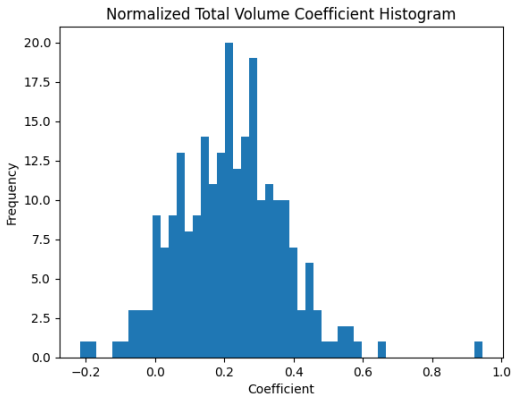


Figure 23: Normalized Total Volume Histogram

8. Discussion

8.1 Discussing Results

Our analysis highlights a statistically significant correlation between increased dark pool activity and heightened price variation. While the significance of this relationship is noteworthy, it is critical to approach these results with caution. The large dataset, comprising over 20,000 observations, naturally amplifies statistical significance, which may not always translate to economic or practical relevance. Furthermore, the direction of causality remains uncertain. It is equally plausible that increased uncertainty in the market drives both heightened price variation and increased dark pool activity, rather than dark pool activity being the root cause of volatility. Notably others have found the inverse correlation between dark pool activity and volatility (He & Lepone, 2014).

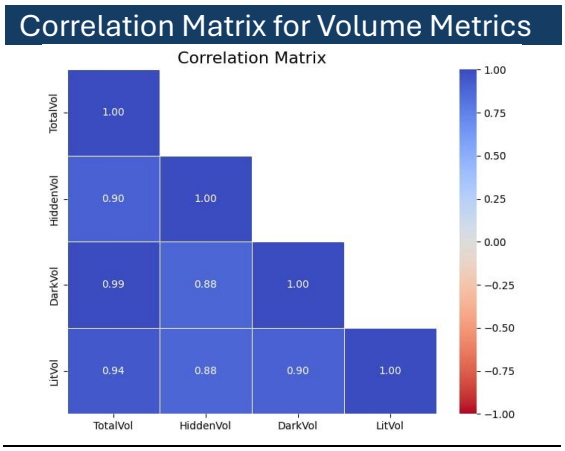


Figure 24: Correlation matrix for volmue metrics

One observation worth noting is the strong correlation between executed volume in dark pools and that in lit exchanges. This suggests that dark pool activity does not operate in isolation but rather reflects broader market dynamics. It might be the case that general market activity and heightened trading volumes in both lit and dark venues contribute to increased volatility, rather than dark pool activity being a direct driver of price variation (He & Lepone, 2014).

8.2 Data & Model Limitations

A key assumption underlying our analysis is that uncertainty in the fundamental value of assets is relatively stable across time. However, this assumption does not hold up to scrutiny. Macro-economic conditions, industry-specific developments, and unforeseen events, such as the imposition of tariffs or geopolitical disruptions, can significantly influence market uncertainty. These external factors likely interact with both dark pool activity and price variation, complicating the interpretation of our results. For instance, in industries where uncertainty spikes due to regulatory changes or earnings announcements, price variation may increase independently of dark pool activity.

Our model provides a framework for exploring the relationship between weekly price variation and dark pool market share, but it is inherently limited in scope. It does not account for daily variations or intraday dynamics, as dark pool volumetric data is aggregated on a weekly basis. Consequently, while we can identify broad trends, our model lacks the granularity needed to predict how specific daily market conditions or order flows influence price variation.

The availability of more frequently updated dark pool data is likely restricted to dark pool operators themselves. Currently, dark pool operators are required to report aggregated weekly data to FINRA, and

there is little incentive for them to increase the reporting frequency. Such high-frequency data is inherently valuable, as it provides insights into market dynamics that could be exploited by participants with access to this information. Furthermore, the nature of dark pool trading inherently prioritizes discretion, making it unlikely that operators would voluntarily disclose more granular data.

Our regression analysis is rooted in the theoretical frameworks proposed by Ye and Kyle. While these models provide valuable insights into market dynamics in a controlled theoretical setting, they oversimplify the complexities of real-world markets. When applied to real-world data, their explanatory power diminishes significantly, as they fail to capture the intricate and multifaceted factors that influence market behavior beyond the theoretical assumptions.

8.3 Market Participants

For large institutional traders, dark pools offer an important advantage by reducing information leakage, thereby mitigating transitory volatility that could arise if large trades were executed on lit exchanges. This discretion allows institutions to manage execution costs more effectively and supports broader market liquidity (Degryse, et al., 2015). However, as noted in our findings, the potential correlation between increased dark pool activity and heightened price variation in lit markets suggests a trade-off. By diverting order flow, dark pools might reduce price transparency and amplify volatility in visible markets, complicating residual trades executed outside of dark venues.

In theory, informed traders can exploit price variations by capitalizing on deviations from the fundamental value. However, in practice, the uncertainty surrounding the true fundamental value is often too significant to execute such strategies effectively.

9. Conclusion

In essence, our findings do not provide evidence of causality between dark pool activity and price variation. Instead, they highlight a correlation that warrants further investigation, ideally with more granular data and additional explanatory variables. The interaction between market uncertainty, trading behavior, and price variation remains complex, and while our analysis sheds light on some aspects of this relationship, it also raises important questions about the broader market dynamics at play.

10. Further Research

10.1 Further Model Development

Our analysis highlights significant limitations in applying linear regression to this dataset, as the models fail to fully meet the assumptions of OLS. Specifically, the presence of heteroskedasticity, autocorrelation, and residual patterns suggest potential non-linear relationships or omitted variable bias (Degiannakis & Xekalaki, 2004). These issues underscore the need for alternative approaches to better understand the dynamics between dark pool activity and price variance. Investigating whether a non-linear relationship exists or whether additional factors could address these issues is an essential next step.

Additionally, a fixed-effects model could provide a more nuanced approach to understanding asset-specific contributions to price variance. By assuming that each asset has its own individual effect, potentially tied to its fundamental value variance, this method could capture unobserved heterogeneity across stocks. Such a model might reveal additional explanatory power by accounting for asset-specific factors that contribute to price variation. On could also attempt to adapt a dynamic Kalman filter to try to capture time-varying parameters (Rockinger & Urga, 2012). However, this approach carries the risk of overfitting by assuming constant, individual effects for each asset, potentially masking the broader market-level dynamics we aim to uncover.

Combining these approaches, non-linear modeling and fixed-effects analysis, could provide a more robust framework for understanding the relationship between dark pool activity and price variance. This integrated perspective would help bridge the gap between theoretical models and observed market behaviors.

10.2 Live Trading Session

One avenue for further exploration could involve organizing a live trading experiment with real participants to study the behavior driving price variance. In such a controlled setting, participants could assume roles such as informed traders, noise traders, and market makers, interacting within a simulated market that includes both lit and dark pool trading venues. By replicating market dynamics in a structured environment, this experiment could provide unique insights into how human decision-making, strategic behavior, and the availability of opaque trading options influence price variance. Such a trading session could serve as an engaging and educational activity, perhaps as part of the FIN11 course. This approach would allow for the evaluation of real human behavior and the dynamics between dark pools and lit markets in a concrete and observable manner.

By comparing a trading session that includes a dark pool to one conducted without it, we could directly observe the impact of dark pools on the market mix. This comparison could provide valuable insights into whether the inclusion of a dark pool enhances or hinders price discovery.

Bibliografi

Brogaard, J., & Pan, J. (2021). *Dark Pool Trading and Information Acquisition*. The Review of Financial Studies, Volume 35, Issue 5, May 2022, Pages 2625–2666,: https://doi.org/10.1093/rfs/hhab098

Buti, s., Rindi, b., & Werner, I. (2022). *Diving into dark pools: The effects of fragmentation on market quality.* Fisher College of Business Working Paper No. 2022-03-01: http://dx.doi.org/10.2139/ssrn.1630499

Chen, J. (2024, December 7). *Investopedia*. Systematic Risk: Definition and Examples : https://www.investopedia.com/terms/s/systematicrisk.asp

Degiannakis, & Xekalaki. (2004). *Quality Technology and Quantitative Management, Vol. 1, No. 2, pp. 271-324, 2004.* Autoregressive Conditional Heteroscedasticity (Arch) Models: A Review: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=798432

Degryse, H., Jong, F. d., & Kervel, V. v. (2014). *The Impact of Dark Trading and Visible Fragmentation on Market Quality.* Review of Finance, Volume 19, Issue 4, July 2015, Pages 1587–1622,: https://doi.org/10.1093/rof/rfu027

Degryse, H., Karagiannis, N., Tombeur, G., & Wuyts, G. (2021). *Two shades of opacity: Hidden orders and dark trading*. Journal of Financial Intermediation, Forthcoming: http://dx.doi.org/10.2139/ssrn.2669447

Financial Industry Regulatory Authority. (2024, December 7). *FINRA*. OTC (ATS & Non-ATS) Transparency: https://www.finra.org/filing-reporting/otc-transparency

He, & Lepone. (2014). *Determinants of liquidity and execution probability in exchange operated dark pool: Evidence from the Australian Securities Exchange.* Pacific-Basin Finance Journal, Volume 30 page 1-16: https://doi.org/10.1016/j.pacfin.2014.07.004

Kyle, A. S. (1985). *Continuous Auctions and Insider Trading*. Econometrica Vol. 53, No. 6 (Nov., 1985), pp. 1315-1335 (21 pages): https://doi.org/10.2307/1913210

Næs, R., & Skjeltorp, J. (2004). *Order Book Characteristics and the Volume-Volatility Relation: Empirical Evidence from a Limit Order Market*. Norges Bank: https://dx.doi.org/10.2139/ssrn.565323

Ntourou, A., & Mallois, A. (2024). *A law and economic analysis of trading through dark pools*. Journal of Financial Regulation and Compliance: https://doi.org/10.1108/JFRC-03-2024-0036

Parkes, D., & Thorpe, C. (2015). *Achieving Trust without Disclosure: Dark Pools and a Role for Secrecy-Preserving Verification*. In Proceedings of the Third Conference on Auctions, Market Mechanisms and Their Applications (AMMA'15): https://econcs.seas.harvard.edu/files/econcs/files/parkes_amma15.pdf

RealVol. (2024, December 7). *Formula Special Considerations*. Adjustment for Dividends: https://www.realvol.com/VolatilityFormula3.html

Rockinger, M., & Urga, G. (2012). *A Time Varying Parameter Model to Test for Predictability and Integration in the Stock Markets of Transition Economies.* https://doi.org/10.1198/073500101524726 34

Rosenblatt Securities. (2024). *RBLT.com*. Let there be light: US edition: https://www.rblt.com/market-structure-reports/let-there-be-light-us-edition-54

US SEC. (2014, November 18). *SEC.gov*. Shedding light on dark pools: https://www.sec.gov/newsroom/speeches-statements/shedding-light-dark-pools#_edn4

Ye, M. (2011). *A Glimpse into the Dark: Price Formation, Transaction Cost and Market Share of the Crossing Network*. Cornell University; National Bureau of Economic Research (NBER); Cornell SC Johnson College of Business: https://dx.doi.org/10.2139/ssrn.1521494

Zhu, H. (2014). *Do Dark Pools Harm Price Discovery?* The Review of Financial Studies, Volume 27, Issue 3, March 2014, Pages 747–789,: https://doi.org/10.1093/rfs/hht078

Zhu, H. (2018). *Variance Risk Premia, Asset Predictability Puzzles, and Macroeconomic Uncertainty*. Annual Review of Financial Economics, Vol. 10:481-497 (Volume publication date November 2018): https://doi.org/10.1146/annurev-financial-110217-022737

Appendix 1

FIN11 Thesis

Data Gathering

In [1]:
```python
# Collect the data from SEC folder
import pandas as pd

# Data collected from Appendix 2

SEC_file = "./SEC_data/all_data.csv"
SEC = pd.read_csv(SEC_file)
```

In [2]:
```python
# Data collected from Appendix 3

FINRA_file = "./FINRA_data/finra_darkpool_volume_all_dropped.csv"
FINRA = pd.read_csv(FINRA_file)
```

In [3]:
```python
import yfinance as yf

def get_price_data(ticker):
    # Check if the data is already downloaded
    try:
        price_data = pd.read_csv(f'./price_data/{ticker}.csv')
        return price_data
    except FileNotFoundError:
        # If not, download the data
        # Get weekly data
        asset = yf.Ticker(ticker)
        price_data = asset.history(start='2019-11-25', end='2024-06-28', interval='1d')

        # Fixing stock split column
        price_data['Stock Splits'] = price_data['Stock Splits'].replace(0, 1)

        # Adjust for stock splits by calculating the cumulative product of 'Stock Splits'
        price_data['Split Factor'] = 1 # price_data['Stock Splits'].cumprod()
        # There are some wierd values in the stock split column, so we ignore it

        # Calculate the Adjusted close
        price_data['Adj Close'] = (price_data['Close'] + price_data['Dividends']) * price_data['Split Factor']

        price_data['Return'] = price_data['Adj Close'].pct_change()

        price_data.to_csv(f'./price_data/{ticker}.csv')

        price_data = pd.read_csv(f'./price_data/{ticker}.csv')
```

```
        return price_data

    except Exception as e:
        print(f'Error for ticker: {ticker}. Error: {e}')
        return None
```

In [4]:
```
tickers = [
    "AAPL", "NVDA", "MSFT", "AVGO", "META", "TSLA", "COST", "NFLX",
    "GOOG", "TMUS", "CSCO", "ADBE", "AMD", "PEP", "LIN", "ISRG", "TXN", "INTU",
    "QCOM", "BKNG", "CMCSA", "AMGN", "HON", "AMAT", "ADP", "PANW", "VRTX", "SBUX",
    "MU", "GILD", "ADI", "INTC", "MELI", "APP", "LRCX", "CTAS", "PYPL", "SNPS",
    "CRWD", "MDLZ", "KLAC", "CDNS", "MRVL", "MAR", "REGN", "CEG", "ORLY", "FTNT",
    "CSX", "ADSK", "DASH", "PDD", "ASML", "ABNB", "CPRT", "PCAR", "ROP", "TTD",
    "NXPI", "WDAY", "CHTR", "FANG", "MNST", "PAYX", "AEP", "ROST", "DDOG", "FAST",
    "ODFL", "KDP", "EA", "BKR", "TEAM", "VRSK", "XEL", "CTSH", "EXC", "KHC",
    "AZN", "GEHC", "LULU", "CCEP", "MCHP", "IDXX", "TTWO", "CSGP", "ZS", "ANSS",
    "ON", "DXCM", "WBD", "MDB", "CDW", "GFS", "BIIB", "ILMN", "SMCI", "ARM", "MRNA"
]
```

In [5]:
```
mondays = FINRA['Date'].unique()
mondays = pd.to_datetime(mondays)
mondays
```

Out[5]:
```
DatetimeIndex(['2019-11-25', '2019-12-02', '2019-12-09', '2019-12-16',
               '2019-12-23', '2019-12-30', '2020-01-06', '2020-01-13',
               '2020-01-20', '2020-01-27',
               ...
               '2024-09-02', '2024-09-09', '2024-09-16', '2024-09-23',
               '2024-09-30', '2024-10-07', '2024-10-14', '2024-10-21',
               '2024-10-28', '2024-11-04'],
              dtype='datetime64[ns]', length=259, freq=None)
```

In [6]:
```
YAHOO = pd.DataFrame()

for ticker in tickers:
    # Collecting price data
    price_data = get_price_data(ticker)

    # Adding ticker column
    price_data['Ticker'] = ticker

    # Keeping only the necessary columns
    price_data = price_data[['Date', 'Ticker', 'Adj Close']]

    # Appending to the main dataframe
    YAHOO = pd.concat([YAHOO, price_data], ignore_index=True)
```

Data Cleaning and Data Transformation

In [7]:
```
# Convert the date to datetime
YAHOO['Date'] = pd.to_datetime(YAHOO['Date'], utc=True)
```

```
# Remove time component
YAHOO['Date'] = YAHOO['Date'].dt.date
```

In [8]:
```
# SEC Date conversion
SEC['Date'] = pd.to_datetime(SEC['Date'], format='%Y%m%d')

# Selecting columns to transform
volume_columns = ["LitVol('000)", "HiddenVol('000)"]

# Dropping NA values
SEC = SEC.dropna(subset=volume_columns)

# Transforming values
SEC[volume_columns] = SEC[volume_columns].apply(lambda x: (x * 1000).astype(int))

# List of columns to keep
columns_to_keep = ["Date", "Security", "Ticker", "LitVol('000)", "LitTrades", "HiddenVol('000)", "TradesForHidden"]

# Select only the specified columns
SEC = SEC[columns_to_keep]
```

In [9]:
```
# Group by 'date' and 'security', summing the 'volume' column
FINRA = FINRA.groupby(['Date', 'Ticker'], as_index=False)['DarkVol'].sum()
```

In [10]:
```
YAHOO_weekly = pd.DataFrame()

YAHOO['Date'] = pd.to_datetime(YAHOO['Date'])

# Create an empty DataFrame for weekly data
YAHOO_weekly = pd.DataFrame()

for monday in mondays:
    monday_ts = pd.Timestamp(monday)

    # Selecting the data for the week
    data = YAHOO[(YAHOO['Date'] >= monday_ts) & (YAHOO['Date'] < monday_ts + pd.Timedelta(days=7))]

    # Group by 'Ticker' and calculate standard deviation and mean
    grouped = data.groupby('Ticker')['Adj Close']
    std_dev = grouped.std()
    mean_val = grouped.mean()

    # Calculate relative variation
    rel_variation = std_dev / mean_val  # Relative variation

    # Create a DataFrame for the weekly data
    weekly_data = pd.DataFrame({
        'Ticker': std_dev.index,
        'relPriceVar': rel_variation,  # New column for relative price variation
        'Adj Close': grouped.last(),   # Standard deviation for Adj Close
        'Date': monday_ts              # Add the week's starting date
    })
```

```
    # Append to the main DataFrame
    YAHOO_weekly = pd.concat([YAHOO_weekly, weekly_data], ignore_index=True)
```

In [11]:
```
YAHOO_weekly
```

Out[11]:

	Ticker	relPriceVar	Adj Close	Date
0	AAPL	0.005832	64.773552	2019-11-25
1	ADBE	0.006178	309.529999	2019-11-25
2	ADI	0.008105	103.033401	2019-11-25
3	ADP	0.003851	153.575455	2019-11-25
4	ADSK	0.035585	180.899994	2019-11-25
...	...	...	...	...
23006	VRTX	0.002119	472.510010	2024-06-24
23007	WBD	0.007119	7.350000	2024-06-24
23008	WDAY	0.013604	223.360001	2024-06-24
23009	XEL	0.001924	53.388115	2024-06-24
23010	ZS	0.024255	189.199997	2024-06-24

23011 rows × 4 columns

In [12]:
```
SEC_weekly = pd.DataFrame()

# Ensure SEC['Date'] is of type pd.Timestamp
SEC['Date'] = pd.to_datetime(SEC['Date'])

for monday in mondays:
    monday_ts = pd.Timestamp(monday)

    # Selecting the data for the week
    data = SEC[(SEC['Date'] >= monday_ts) & (SEC['Date'] < monday_ts + pd.Timedelta(days=7))]

    # Grouping by Ticker and summing numeric columns only
    weekly_data = data.groupby('Ticker', as_index=False).sum(numeric_only=True)

    # Add the starting Monday's date to indicate the week
    weekly_data['Date'] = monday_ts

    # Append the weekly data to the main DataFrame
    SEC_weekly = pd.concat([SEC_weekly, weekly_data], ignore_index=True)
```

In [13]:
```
# Ensure 'Date' column is in datetime format for all DataFrames
FINRA['Date'] = pd.to_datetime(FINRA['Date'])
YAHOO_weekly['Date'] = pd.to_datetime(YAHOO_weekly['Date'])

# Merge FINRA and YAHOO on 'Date' and 'Ticker'
```

```
DATA = pd.merge(YAHOO_weekly, SEC_weekly , on=['Date', 'Ticker'], how='inner')

# Merge the resulting DataFrame with SEC on 'Date' and 'Ticker'
DATA = pd.merge(DATA, FINRA, on=['Date', 'Ticker'], how='inner')
```

In [14]:
```
DATA
```

Out[14]:

	Ticker	relPriceVar	Adj Close	Date	LitVol('000)	LitTrades	HiddenVol('000)	TradesForHidden	DarkVol
0	AAPL	0.005832	64.773552	2019-11-25	22448390	251574.0	5035333	278950.0	29909206
1	ADBE	0.006178	309.529999	2019-11-25	2957048	59965.0	844244	74100.0	3420523
2	ADI	0.008105	103.033401	2019-11-25	5441187	81955.0	1401880	98722.0	10237731
3	ADP	0.003851	153.575455	2019-11-25	1590754	34667.0	289945	39658.0	3104617
4	ADSK	0.035585	180.899994	2019-11-25	3714576	61641.0	1479198	80374.0	3148558
...	...	...	...	...	...	...	...	...	...
20202	VRTX	0.002119	472.510010	2024-06-24	1225286	48703.0	935442	81975.0	6544350
20203	WBD	0.007119	7.350000	2024-06-24	28166643	77857.0	10776873	122025.0	87586114
20204	WDAY	0.013604	223.360001	2024-06-24	2361256	66728.0	1086355	94223.0	9377862
20205	XEL	0.001924	53.388115	2024-06-24	3566062	70795.0	798527	85061.0	10392894
20206	ZS	0.024255	189.199997	2024-06-24	3051604	83035.0	1531006	121387.0	9780462

20207 rows × 9 columns

In [15]:
```
DATA = DATA.rename(columns={
    'LitVol(\'000)': 'LitVol',
    'HiddenVol(\'000)': 'HiddenVol',
    'TradesForHidden': 'HiddenTrades'
    })

# Keep only the necessary columns
column_order = ['Date', 'Ticker', 'relPriceVar', 'LitVol', 'HiddenVol', 'DarkVol', 'Adj Close']

DATA = DATA[column_order]
```

In [16]:
```
DATA['TotalVol'] = DATA['LitVol'] + DATA['HiddenVol'] + DATA['DarkVol']
```

In [17]:
```
DATA['DarkVolShare'] = DATA['DarkVol'] / DATA['TotalVol']
```

In [18]:
```
DATA['k_param'] = DATA['DarkVol'] / (DATA['LitVol'] + DATA['HiddenVol'])
```

In [19]:
```
DATA
```

Out[19]:

	Date	Ticker	relPriceVar	LitVol	HiddenVol	DarkVol	Adj Close	TotalVol	DarkVolShare	k_param
0	2019-11-25	AAPL	0.005832	22448390	5035333	29909206	64.773552	57392929	0.521131	1.088252
1	2019-11-25	ADBE	0.006178	2957048	844244	3420523	309.529999	7221815	0.473638	0.899832
2	2019-11-25	ADI	0.008105	5441187	1401880	10237731	103.033401	17080798	0.599371	1.496073
3	2019-11-25	ADP	0.003851	1590754	289945	3104617	153.575455	4985316	0.622752	1.650778
4	2019-11-25	ADSK	0.035585	3714576	1479198	3148558	180.899994	8342332	0.377419	0.606218
...	...	...	...	...	...	...	...	...	...	...
20202	2024-06-24	VRTX	0.002119	1225286	935442	6544350	472.510010	8705078	0.751785	3.028771
20203	2024-06-24	WBD	0.007119	28166643	10776873	87586114	7.350000	126529630	0.692218	2.249055
20204	2024-06-24	WDAY	0.013604	2361256	1086355	9377862	223.360001	12825473	0.731190	2.720104
20205	2024-06-24	XEL	0.001924	3566062	798527	10392894	53.388115	14757483	0.704246	2.381185
20206	2024-06-24	ZS	0.024255	3051604	1531006	9780462	189.199997	14363072	0.680945	2.134256

20207 rows × 10 columns

In [20]:
```
# Calculate varriance in return for all stocks on each date
index_var = (
    DATA.groupby('Date', group_keys=False)
    .apply(lambda x: x['relPriceVar'].mean())
)

# Add index variance column
DATA['IndexVar'] = DATA['Date'].map(index_var)

# Drop rows with NaN values
DATA = DATA.dropna()
```

```
C:\Users\kimsp\AppData\Local\Temp\ipykernel_28600\495771617.py:4: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `include_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
  .apply(lambda x: x['relPriceVar'].mean())
```

In [21]:
```
norTotalVol = (
    DATA.groupby(['Ticker', 'Date'], group_keys=False)
    .apply(lambda x: ((x['TotalVol'] - x['TotalVol'].mean())/x['TotalVol']).std())
)

# Compute mean and std per Ticker
ticker_stats = DATA.groupby('Ticker')['TotalVol'].agg(['mean', 'std']).reset_index()
ticker_stats.rename(columns={'mean': 'TickerMean', 'std': 'TickerStd'}, inplace=True)

# Create a new DataFrame with normalized values
norTotalVol = DATA.merge(ticker_stats, on='Ticker', how='left')

# Add normalized column
```

```
norTotalVol['z_normalized_TotalVol'] = (
    (norTotalVol['TotalVol'] - norTotalVol['TickerMean']) / norTotalVol['TickerStd']
)

# Select only relevant columns for the new DataFrame
normalized_df = norTotalVol[['Ticker', 'Date', 'TotalVol', 'z_normalized_TotalVol']]

DATA['norTotalVol'] = normalized_df['z_normalized_TotalVol']

DATA['transotoryVar'] = DATA['relPriceVar'] - DATA['IndexVar']

# Drop rows with NaN values (if any due to missing stats or TotalVol)
DATA = DATA.dropna()

# Display the updated DataFrame
DATA
```

```
C:\Users\kimsp\AppData\Local\Temp\ipykernel_28600\713766642.py:3: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `include_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
  .apply(lambda x: ((x['TotalVol'] - x['TotalVol'].mean())/x['TotalVol']).std())
```

Out[21]:

	Date	Ticker	relPriceVar	LitVol	HiddenVol	DarkVol	Adj Close	TotalVol	DarkVolShare	k_param	IndexVar	norTotalVol	transotoryVar
0	2019-11-25	AAPL	0.005832	22448390	5035333	29909206	64.773552	57392929	0.521131	1.088252	0.008838	-1.905879	-0.003006
1	2019-11-25	ADBE	0.006178	2957048	844244	3420523	309.529999	7221815	0.473638	0.899832	0.008838	-1.214933	-0.002659
2	2019-11-25	ADI	0.008105	5441187	1401880	10237731	103.033401	17080798	0.599371	1.496073	0.008838	-0.052535	-0.000732
3	2019-11-25	ADP	0.003851	1590754	289945	3104617	153.575455	4985316	0.622752	1.650778	0.008838	-1.248559	-0.004986
4	2019-11-25	ADSK	0.035585	3714576	1479198	3148558	180.899994	8342332	0.377419	0.606218	0.008838	-0.035148	0.026747
...	...	...	...	...	...	...	...	...	...	...	...	...	...
20202	2024-06-24	VRTX	0.002119	1225286	935442	6544350	472.510010	8705078	0.751785	3.028771	0.012753	-0.020440	-0.010633
20203	2024-06-24	WBD	0.007119	28166643	10776873	87586114	7.350000	126529630	0.692218	2.249055	0.012753	-0.352989	-0.005634
20204	2024-06-24	WDAY	0.013604	2361256	1086355	9377862	223.360001	12825473	0.731190	2.720104	0.012753	0.106143	0.000851
20205	2024-06-24	XEL	0.001924	3566062	798527	10392894	53.388115	14757483	0.704246	2.381185	0.012753	-0.293925	-0.010829
20206	2024-06-24	ZS	0.024255	3051604	1531006	9780462	189.199997	14363072	0.680945	2.134256	0.012753	-0.025168	0.011502

20207 rows × 13 columns

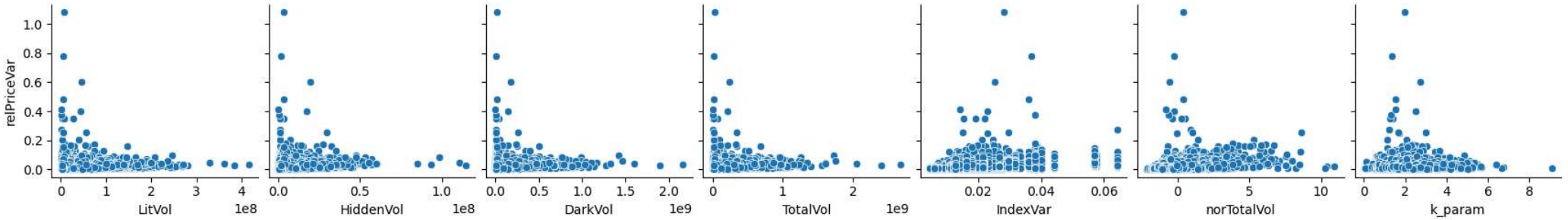
In [22]:
```
# Descriptive statistics of DATA
DATA.describe()
```

Out[22]:

	Date	relPriceVar	LitVol	HiddenVol	DarkVol	Adj Close	TotalVol	DarkVolShare	k_param	IndexVar	norTotalVol	transotoryVar
count	20207	20207.000000	2.020700e+04	2.020700e+04	2.020700e+04	20207.000000	2.020700e+04	20207.000000	20207.000000	20207.000000	2.020700e+04	2.020700e+04
mean	2022-04-03 13:34:27.501361152	0.019237	1.242026e+07	2.970585e+06	3.236691e+07	258.358433	4.775775e+07	0.608799	1.655476	0.019237	-1.687833e-17	-2.702484e-19
min	2019-11-25 00:00:00	0.000300	9.393300e+04	8.307900e+04	8.120500e+04	5.125693	5.585140e+05	0.023087	0.023633	0.004659	-2.123949e+00	-4.968624e-02
25%	2021-02-08 00:00:00	0.009088	1.883022e+06	7.933190e+05	4.221647e+06	66.730518	7.134772e+06	0.563562	1.291275	0.014633	-6.547665e-01	-7.970737e-03
50%	2022-04-18 00:00:00	0.014682	4.249169e+06	1.401179e+06	8.886430e+06	138.729996	1.478796e+07	0.608673	1.555406	0.017228	-2.148917e-01	-3.132601e-03
75%	2023-05-29 00:00:00	0.023581	1.121478e+07	2.775006e+06	2.101656e+07	283.294998	3.512926e+07	0.654791	1.896794	0.022412	4.047855e-01	3.980174e-03
max	2024-06-24 00:00:00	1.081302	4.156547e+08	1.145599e+08	2.163751e+09	5274.749146	2.672737e+09	0.900825	9.083202	0.064272	1.094555e+01	1.052976e+00
std	NaN	0.020037	2.389732e+07	5.023092e+06	8.784153e+07	416.910878	1.144109e+08	0.072706	0.572662	0.007767	9.977953e-01	1.847096e-02

In [23]:
```
# Checking for non linear realation ships between relPriceVar and exlanitory variables
import seaborn as sns
import matplotlib.pyplot as plt

# Plotting the relationships
sns.pairplot(DATA, x_vars=['LitVol', 'HiddenVol', 'DarkVol', 'TotalVol', 'IndexVar', 'norTotalVol', 'k_param'], y_vars='relPriceVar', kind='scatter')
plt.show()
```

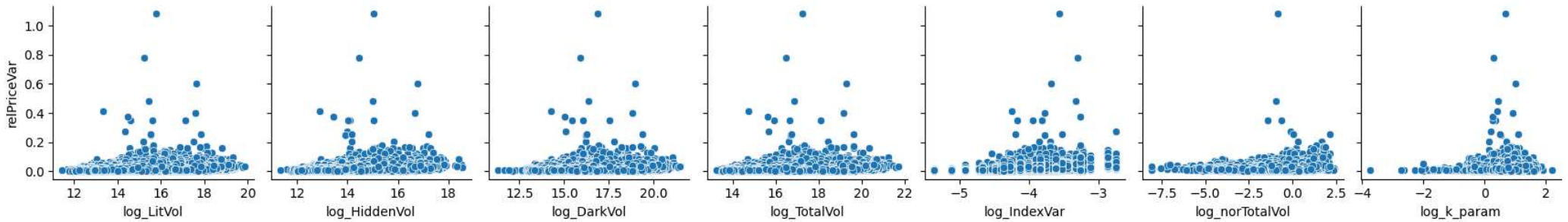


In [24]:
```
import numpy as np

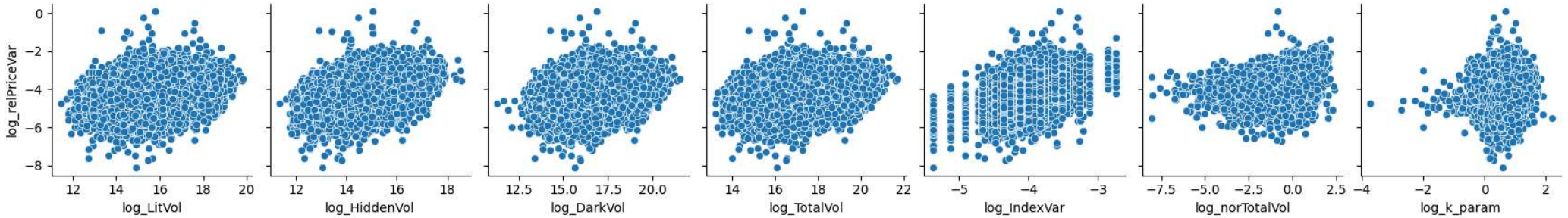
# Log transform and graph
DATA['log_LitVol'] = DATA['LitVol'].apply(lambda x: 1 if x == 0 else np.log(x))
DATA['log_HiddenVol'] = DATA['HiddenVol'].apply(lambda x: 1 if x == 0 else np.log(x))
DATA['log_DarkVol'] = DATA['DarkVol'].apply(lambda x: 1 if x == 0 else np.log(x))
DATA['log_TotalVol'] = DATA['TotalVol'].apply(lambda x: 1 if x == 0 else np.log(x))
DATA['log_IndexVar'] = DATA['IndexVar'].apply(lambda x: 1 if x == 0 else np.log(x))
DATA['log_norTotalVol'] = DATA['norTotalVol'].apply(lambda x: 1 if x == 0 else np.log(x))
DATA['log_k_param'] = DATA['k_param'].apply(lambda x: 1 if x == 0 else np.log(x))
DATA['log_relPriceVar'] = DATA['relPriceVar'].apply(lambda x: 1 if x == 0 else np.log(x))

# Plotting the relationships
sns.pairplot(DATA, x_vars=['log_LitVol', 'log_HiddenVol', 'log_DarkVol', 'log_TotalVol', 'log_IndexVar', 'log_norTotalVol', 'log_k_param'], y_vars='relPriceVar', kind='scatter')
plt.show()
```

```
C:\Users\kimsp\AppData\Local\Temp\ipykernel_28600\578315581.py:9: RuntimeWarning: invalid value encountered in log
  DATA['log_norTotalVol'] = DATA['norTotalVol'].apply(lambda x: 1 if x == 0 else np.log(x))
```



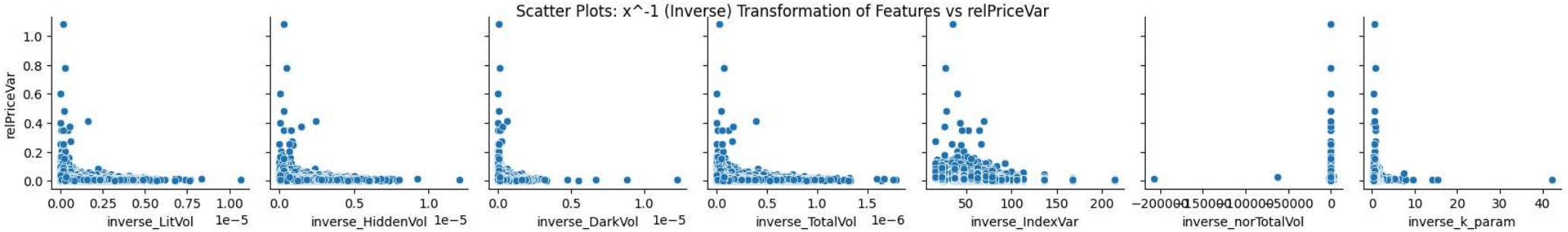
In [25]:
```
sns.pairplot(DATA, x_vars=['log_LitVol', 'log_HiddenVol', 'log_DarkVol', 'log_TotalVol', 'log_IndexVar', 'log_norTotalVol', 'log_k_param'], y_vars='log_relPriceVar', kind='scatter
plt.show()
```



In [26]:
```
# Apply inverse transformation (x^-1 = 1/x)
DATA['inverse_LitVol'] = DATA['LitVol'].apply(lambda x: 1 / (x + 1e-8))  # Add small constant to avoid division by zero
DATA['inverse_HiddenVol'] = DATA['HiddenVol'].apply(lambda x: 1 / (x + 1e-8))
DATA['inverse_DarkVol'] = DATA['DarkVol'].apply(lambda x: 1 / (x + 1e-8))
DATA['inverse_TotalVol'] = DATA['TotalVol'].apply(lambda x: 1 / (x + 1e-8))
DATA['inverse_IndexVar'] = DATA['IndexVar'].apply(lambda x: 1 / (x + 1e-8))
DATA['inverse_norTotalVol'] = DATA['norTotalVol'].apply(lambda x: 1 / (x + 1e-8))
DATA['inverse_k_param'] = DATA['k_param'].apply(lambda x: 1 / (x + 1e-8))

# Plotting the relationships using the inverse transformation
sns.pairplot(
    DATA,
    x_vars=[
        'inverse_LitVol',
        'inverse_HiddenVol',
        'inverse_DarkVol',
        'inverse_TotalVol',
        'inverse_IndexVar',
        'inverse_norTotalVol',
        'inverse_k_param'
    ],
    y_vars='relPriceVar',
    kind='scatter'
)
```

```
plt.suptitle("Scatter Plots: x^-1 (Inverse) Transformation of Features vs relPriceVar", y=1.02)
plt.show()
```



Correlation matrix

In [27]:
```
# Import necessary libraries
import seaborn as sns
import matplotlib.pyplot as plt

# Correlation matrix
corr = DATA[['relPriceVar', 'norTotalVol', 'log_k_param', 'log_IndexVar', 'log_relPriceVar']].corr()

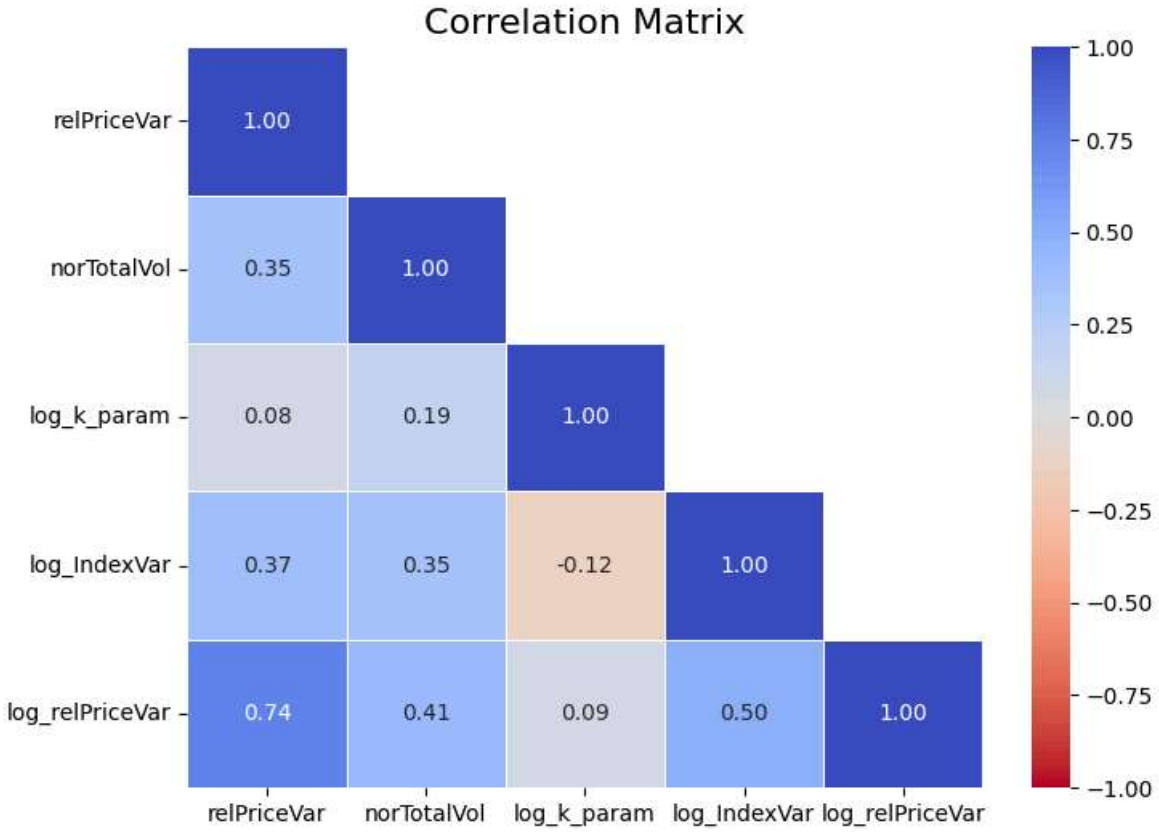
# Mask the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool), k=1)

# Set up the matplotlib figure
plt.figure(figsize=(8, 6))

# Draw the heatmap with the mask
sns.heatmap(corr, mask=mask, annot=True, fmt=".2f", cmap="coolwarm_r", vmin=-1, vmax=1, linewidths=0.5)

# Add title
plt.title('Correlation Matrix', fontsize=16)

# Show the plot
plt.show()
```



In [28]:

```
# Correlation matrix
corr = DATA[['relPriceVar', 'k_param', 'log_k_param', 'log_IndexVar', 'log_norTotalVol']].corr()

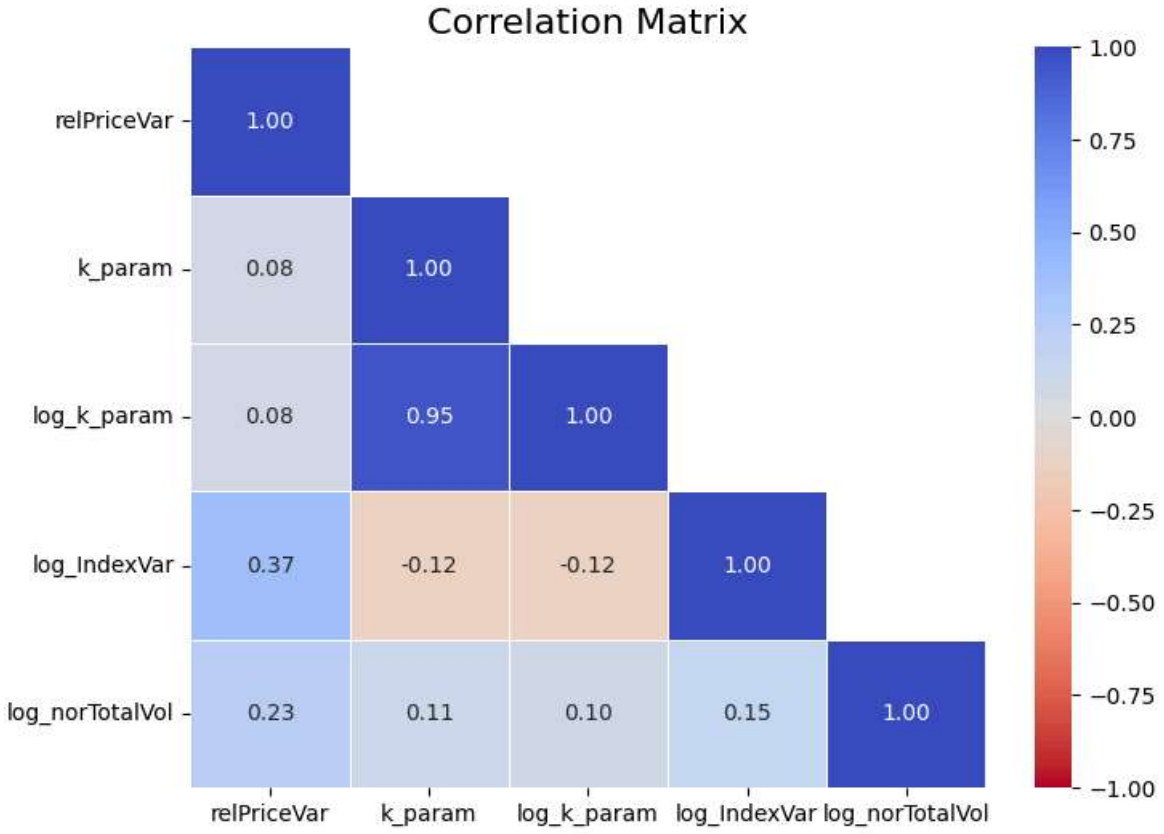
# Mask the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool), k=1)

# Set up the matplotlib figure
plt.figure(figsize=(8, 6))

# Draw the heatmap with the mask
sns.heatmap(corr, mask=mask, annot=True, fmt=".2f", cmap="coolwarm_r", vmin=-1, vmax=1, linewidths=0.5)

# Add title
plt.title('Correlation Matrix', fontsize=16)

# Show the plot
plt.show()
```



In [77]:

```
# Correlation matrix
corr = DATA[['TotalVol', 'HiddenVol', 'DarkVol', 'LitVol']].corr()

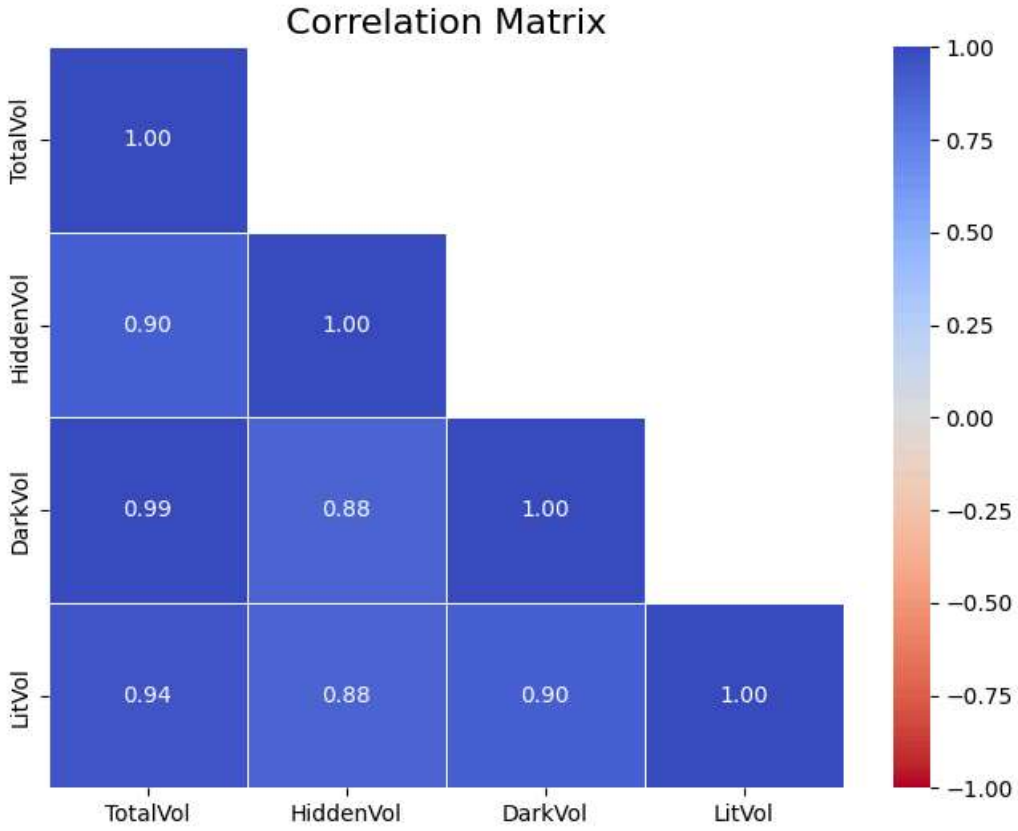
# Mask the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool), k=1)

# Set up the matplotlib figure
plt.figure(figsize=(8, 6))

# Draw the heatmap with the mask
sns.heatmap(corr, mask=mask, annot=True, fmt=".2f", cmap="coolwarm_r", vmin=-1, vmax=1, linewidths=0.5)

# Add title
plt.title('Correlation Matrix', fontsize=16)

# Show the plot
plt.show()
```



Cross-Section regression

In [30]:

```
import statsmodels.api as sm

cross_data = DATA

tickers = cross_data['Ticker'].unique()
dates = cross_data['Date'].unique()

results_list = []  # Use a list to store DataFrames

for date in dates:
    # Filter data for the current date
    date_data = cross_data[cross_data['Date'] == date]

    # Initialize the regression model
    model = sm.OLS(date_data['log_relPriceVar'], sm.add_constant(date_data[['log_k_param', 'log_IndexVar', 'norTotalVol']]))
    results = model.fit()

    # Create a DataFrame from the results
    params_df = pd.DataFrame([results.params]).assign(Date=date)
```

```
    results_list.append(params_df)  # Append the DataFrame to the list

# Concatenate all results into a single DataFrame
results_df = pd.concat(results_list, ignore_index=True)

results_df
```

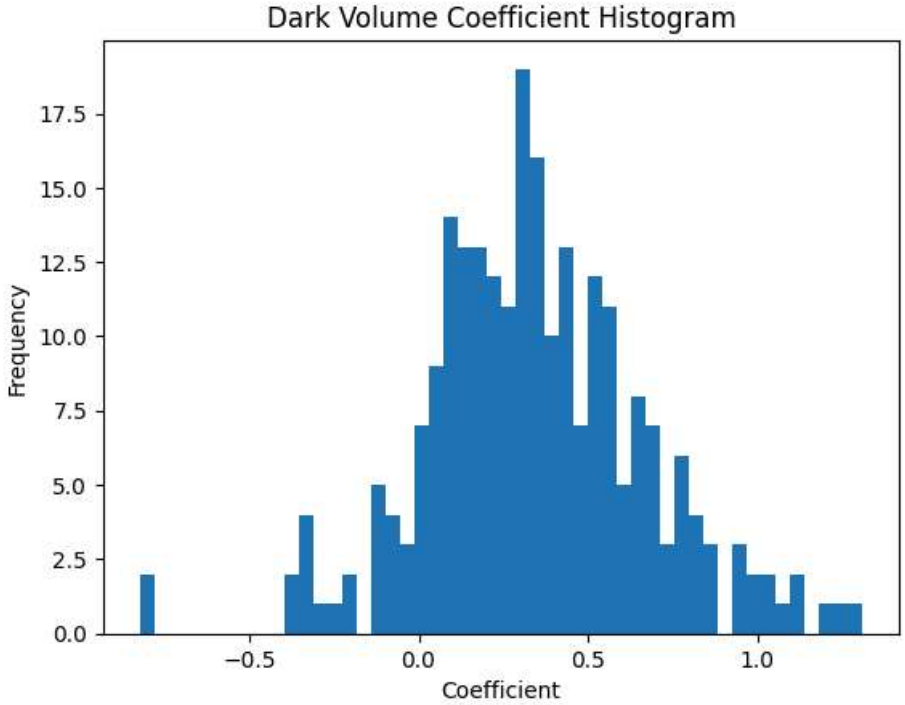
Out[30]:

	log_k_param	log_IndexVar	norTotalVol	Date
0	-0.311903	0.926072	0.544302	2019-11-25
1	0.022507	1.005430	0.168107	2019-12-02
2	0.693409	1.106210	0.140905	2019-12-09
3	-0.307698	1.001791	0.074770	2019-12-16
4	0.394539	0.850563	0.946851	2019-12-23
...	...	...	...	...
235	-0.385156	0.992291	0.187804	2024-05-27
236	0.341289	1.104361	0.039804	2024-06-03
237	0.190123	1.114536	0.213413	2024-06-10
238	-0.795752	0.903412	0.161721	2024-06-17
239	-0.823942	0.917009	0.223130	2024-06-24

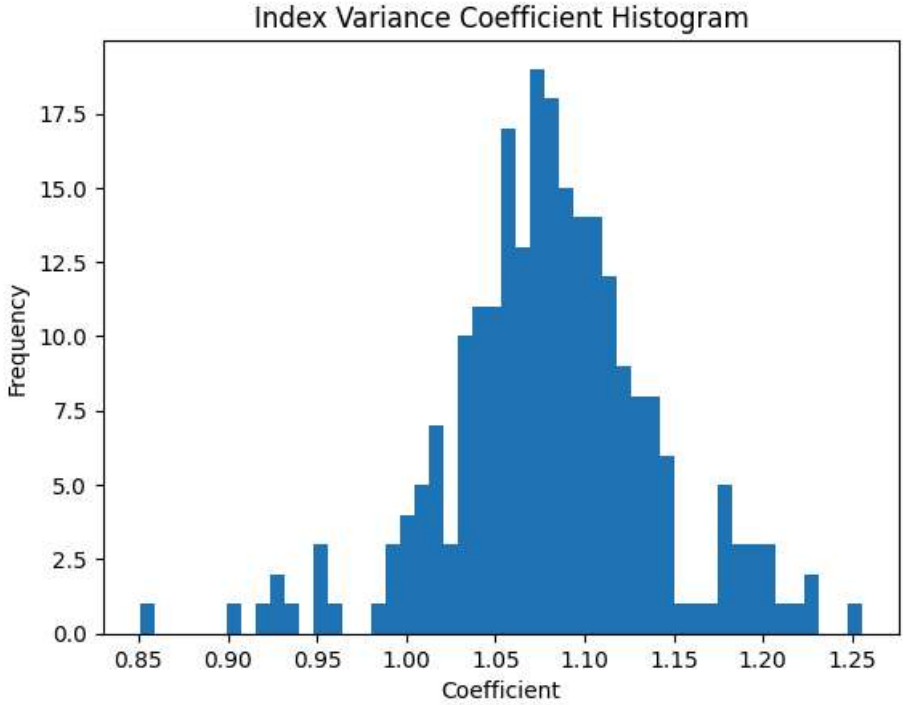
240 rows × 4 columns

In [31]:

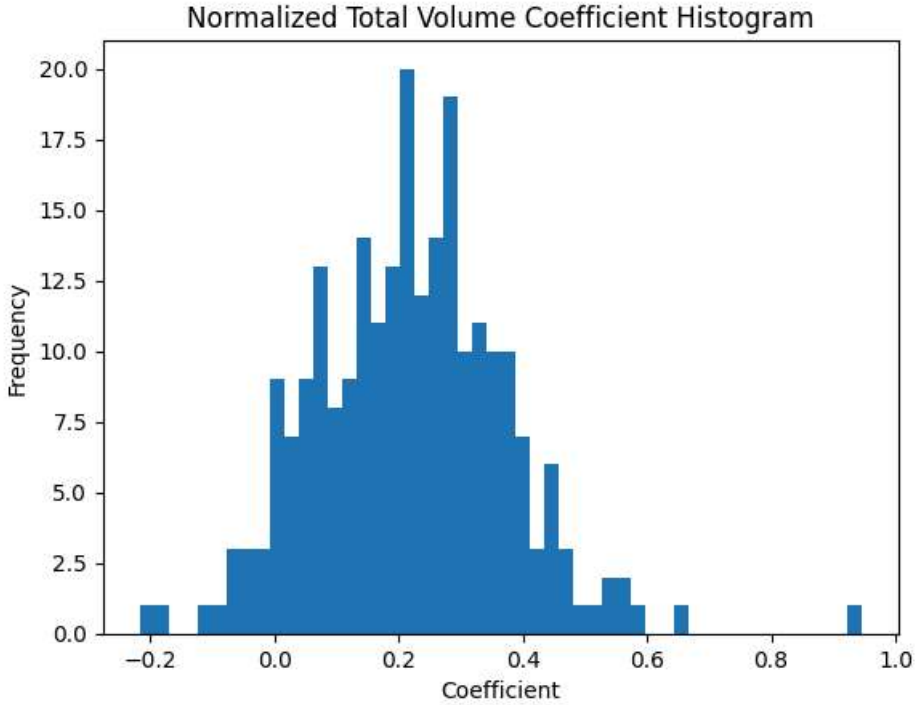
```
# Histogram of the coefficients
plt.hist(results_df['log_k_param'], bins=50)
plt.xlabel('Coefficient')
plt.ylabel('Frequency')
plt.title('Dark Volume Coefficient Histogram')
plt.show()
```



In [32]:
```
# Histogram of the coefficients
plt.hist(results_df['log_IndexVar'], bins=50)
plt.xlabel('Coefficient')
plt.ylabel('Frequency')
plt.title('Index Variance Coefficient Histogram')
plt.show()
```



In [33]:
```
# Histogram of the coefficients
plt.hist(results_df['norTotalVol'], bins=50)
plt.xlabel('Coefficient')
plt.ylabel('Frequency')
plt.title('Normalized Total Volume Coefficient Histogram')
plt.show()
```



Linear regression

Model 1

In [34]:
```
# Linreg of all data
X = DATA[['k_param']]
y = DATA['relPriceVar']

model1 = sm.OLS(y, sm.add_constant(X))
model1_results = model1.fit()

print(model1_results.summary().as_text())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            relPriceVar   R-squared:                       0.007
Model:                            OLS   Adj. R-squared:                  0.007
Method:                 Least Squares   F-statistic:                     147.0
Date:                Thu, 05 Dec 2024   Prob (F-statistic):           1.03e-33
Time:                        19:44:18   Log-Likelihood:                 50414.
No. Observations:               20207   AIC:                        -1.008e+05
Df Residuals:                   20205   BIC:                        -1.008e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0143      0.000     33.316      0.000       0.013       0.015
k_param        0.0030      0.000     12.124      0.000       0.002       0.003
==============================================================================
Omnibus:                    39730.004   Durbin-Watson:                   1.688
Prob(Omnibus):                  0.000   Jarque-Bera (JB):        291510387.357
Skew:                          15.285   Prob(JB):                         0.00
Kurtosis:                     590.618   Cond. No.                         6.96
==============================================================================

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

In [35]:

```python
from statsmodels.iolib.summary2 import summary_col

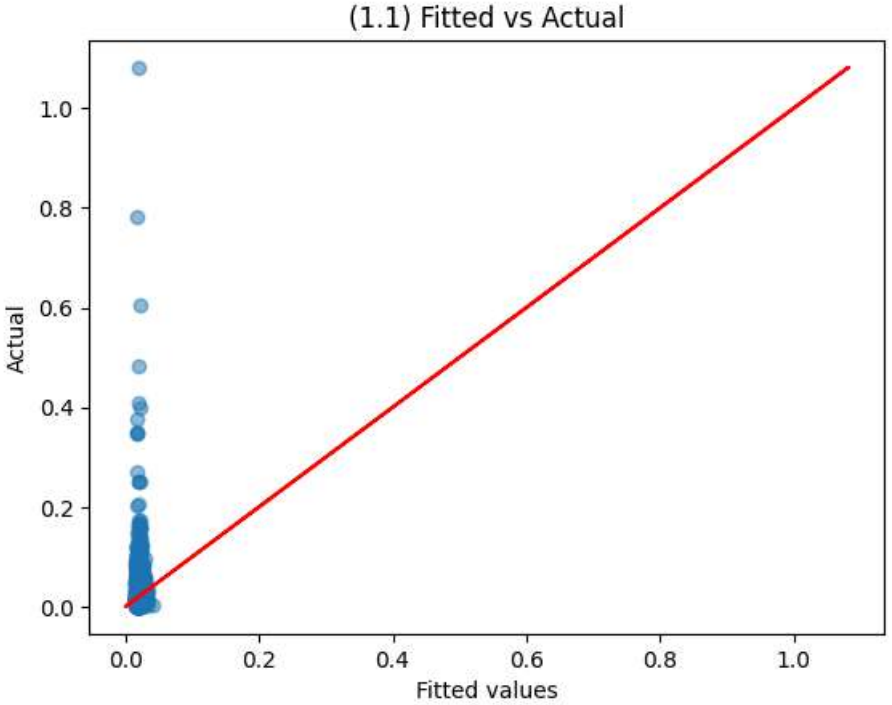
results_table = summary_col([model1_results],
                            stars=True,
                            float_format='%0.3f',
                            model_names=['(1.1)'],
                            info_dict={'N': lambda x: f"{int(x.nobs)}",
                                       'Log Likelihood': lambda x: f"{x.llf:.3f}",
                                       'AIC': lambda x: f"{x.aic:.3f}"})

# Add significance levels and notes
results_table.add_title('Model 1: Price Variation vs k parameter')

# Print or save the results
print(results_table.as_text()) # Can do as latex
```

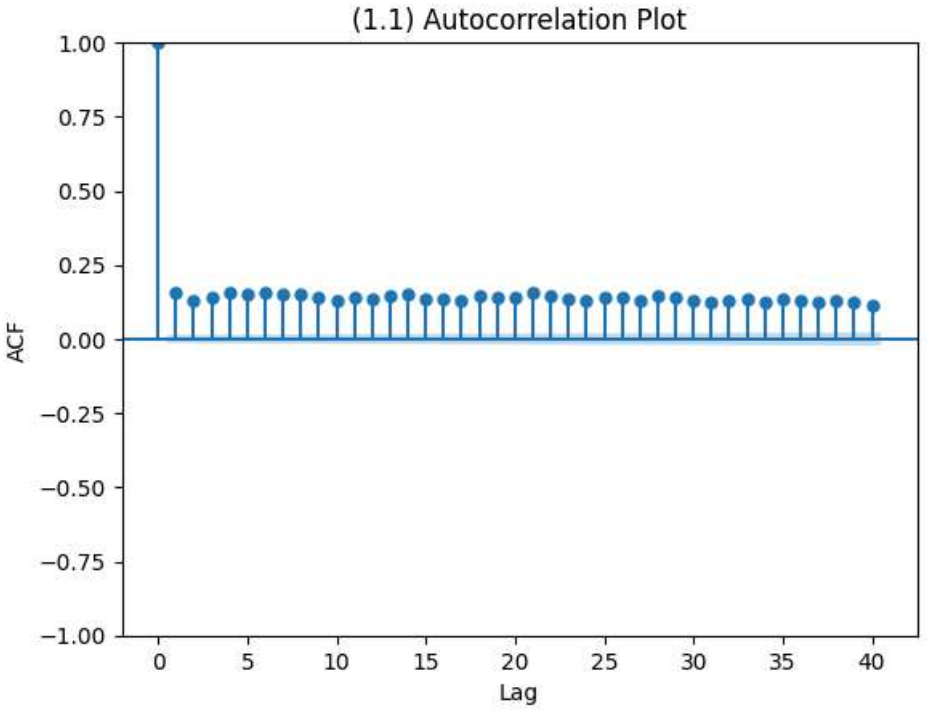
```
Model 1: Price Variation vs k parameter
==========================
                  (1.1)
--------------------------
const          0.014***
               (0.000)
k_param        0.003***
               (0.000)
R-squared      0.007
R-squared Adj. 0.007
N              20207
Log Likelihood 50413.676
AIC            -100823.352
==========================
Standard errors in
parentheses.
* p<.1, ** p<.05, ***p<.01
```

In [36]:
```
# Plotting the predicted vs actual values
plt.scatter(model1_results.fittedvalues, DATA['relPriceVar'], alpha=0.5)
plt.plot(DATA['relPriceVar'], DATA['relPriceVar'], color='red')
plt.xlabel('Fitted values')
plt.ylabel('Actual')
plt.title('(1.1) Fitted vs Actual')
plt.show()
```

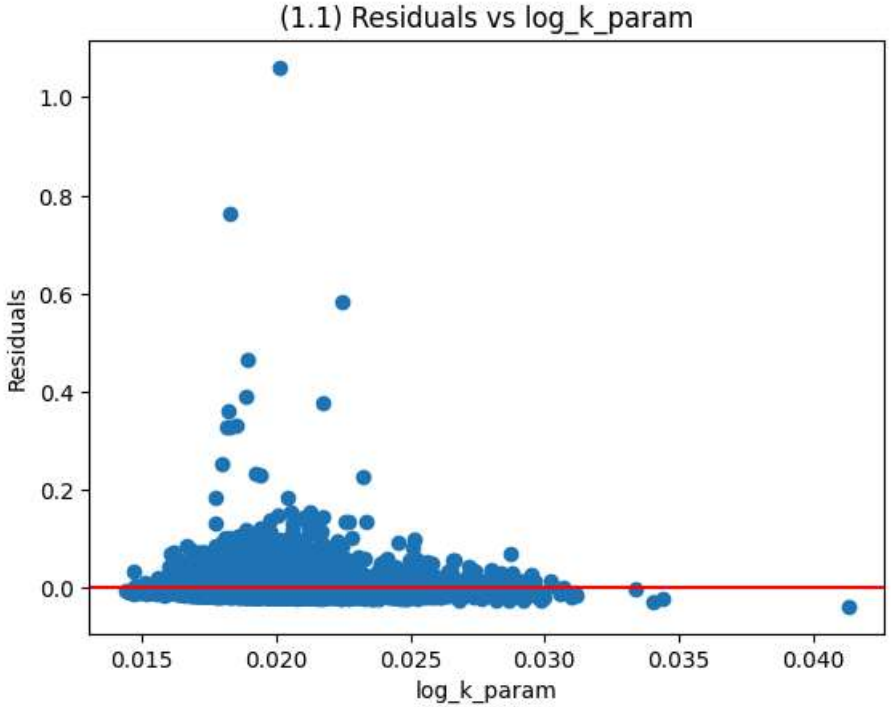


In [37]:
```
import statsmodels.tsa.api as smt
```

```
acf = smt.graphics.plot_acf(model1_results.resid, lags=40, alpha=0.05)
plt.xlabel('Lag')
plt.ylabel('ACF')
plt.title('(1.1) Autocorrelation Plot')
plt.show()
```

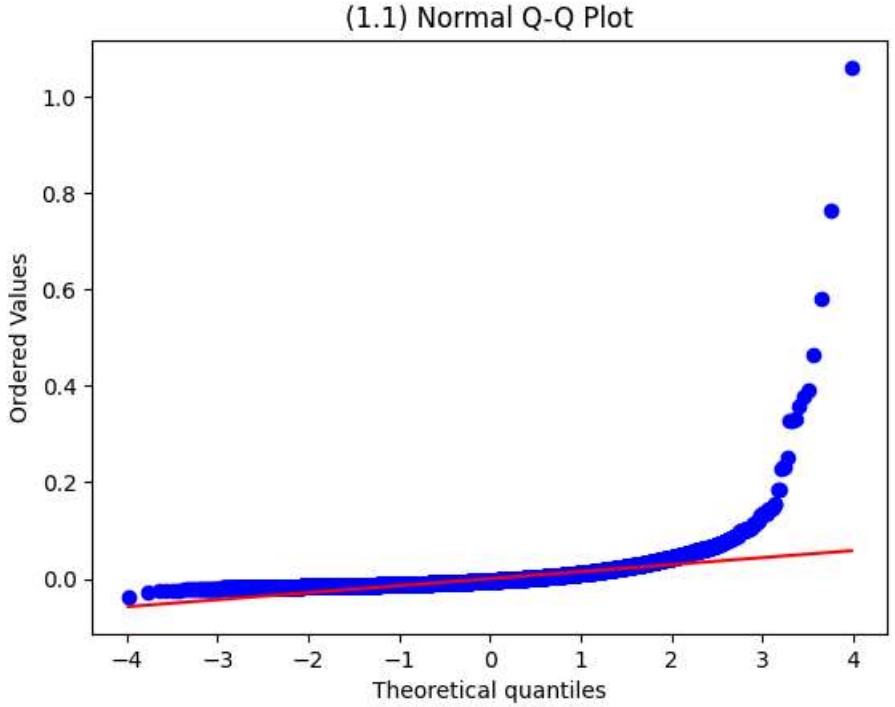


In [38]:
```
plt.scatter(model1_results.fittedvalues, model1_results.resid)
plt.axhline(y=0, color='r', linestyle='-')
plt.xlabel('log_k_param')
plt.ylabel('Residuals')
plt.title('(1.1) Residuals vs log_k_param')
plt.show()
```

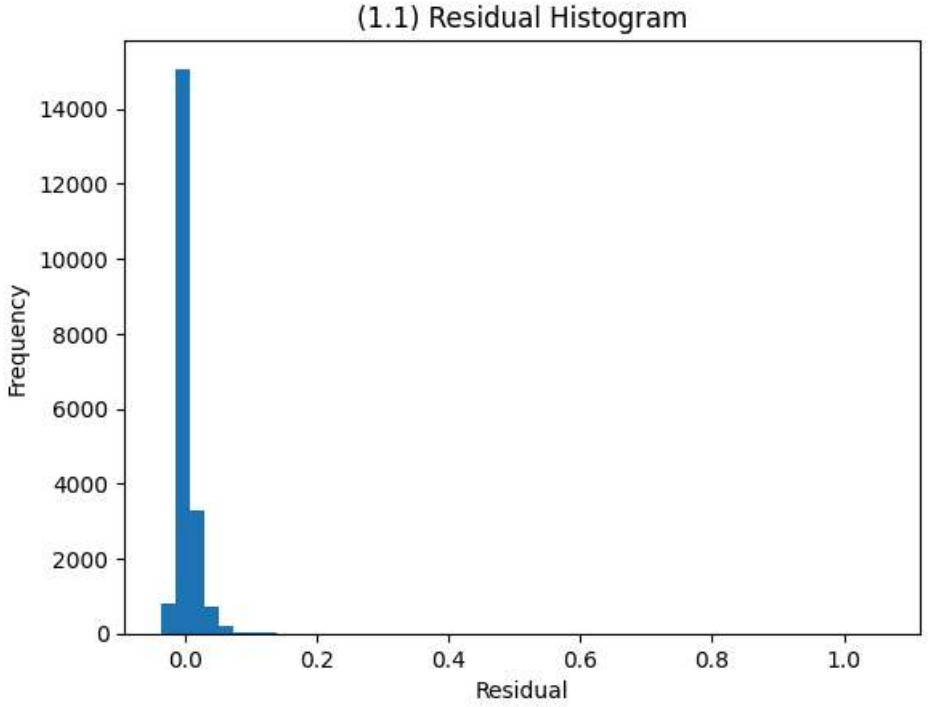


In [39]:
```
import scipy.stats as stats

stats.probplot(model1_results.resid, dist='norm', plot=plt)
plt.title('(1.1) Normal Q-Q Plot')
plt.show()
```



In [40]:
```
# Histogram residuals
plt.hist(model1_results.resid, bins=50)
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.title('(1.1) Residual Histogram')
plt.show()
```



Model 1

log transformed

In [41]:
```
# Linreg of all data
X = DATA[['log_k_param']]
y = DATA['log_relPriceVar']

model1_log = sm.OLS(y, sm.add_constant(X))
model1_log_results = model1_log.fit()

print(model1_log_results.summary().as_text())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        log_relPriceVar   R-squared:                       0.007
Model:                            OLS   Adj. R-squared:                  0.007
Method:                 Least Squares   F-statistic:                     147.7
Date:                Thu, 05 Dec 2024   Prob (F-statistic):           7.19e-34
Time:                        19:44:22   Log-Likelihood:                -22326.
No. Observations:               20207   AIC:                          4.466e+04
Df Residuals:                   20205   BIC:                          4.467e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          -4.3138      0.009   -484.261      0.000      -4.331      -4.296
log_k_param     0.1954      0.016     12.154      0.000       0.164       0.227
==============================================================================
Omnibus:                      143.575   Durbin-Watson:                   1.489
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              215.583
Skew:                          -0.047   Prob(JB):                     1.54e-47
Kurtosis:                       3.497   Cond. No.                         3.83
==============================================================================

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

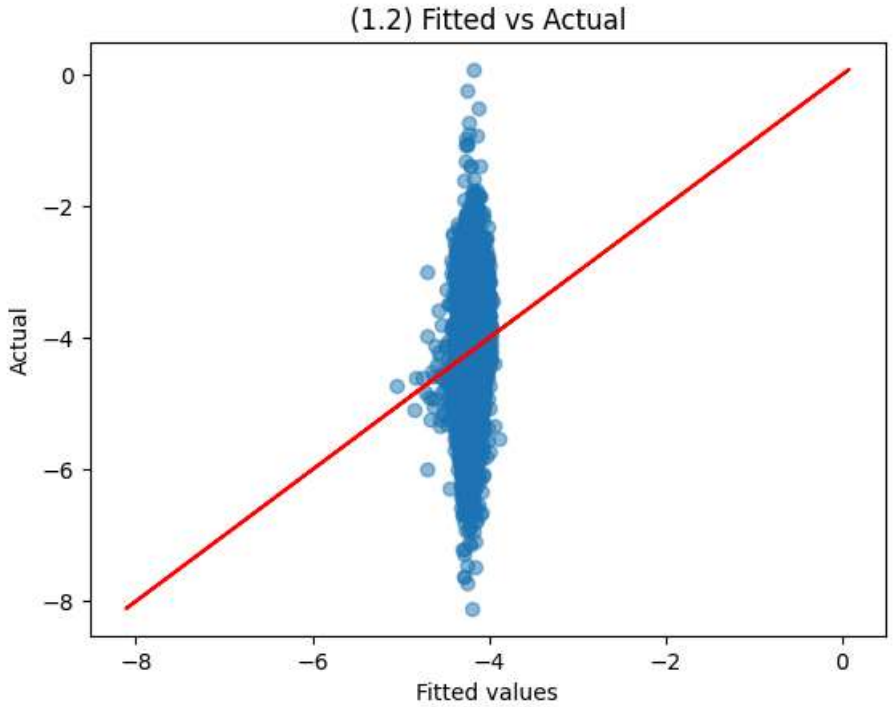
In [42]:
```python
results_table = summary_col([model1_log_results],
                            stars=True,
                            float_format='%0.3f',
                            model_names=['(1.2)'],
                            info_dict={'N': lambda x: f"{int(x.nobs)}",
                                       'Log Likelihood': lambda x: f"{x.llf:.3f}",
                                       'AIC': lambda x: f"{x.aic:.3f}"})

# Add significance levels and notes
results_table.add_title('Model 1.2: Price Variation vs k parameter')

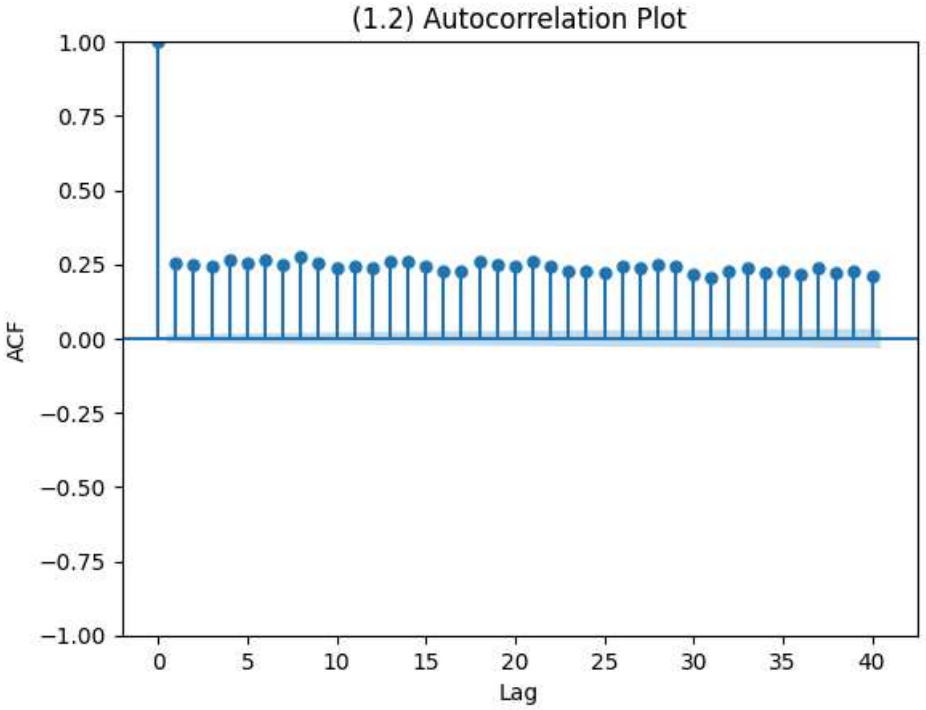
# Print or save the results
print(results_table.as_text()) # Can do as latex
```

```
Model 1.2: Price Variation vs k parameter
==========================
                  (1.2)
--------------------------
const           -4.314***
                (0.009)
log_k_param     0.195***
                (0.016)
R-squared       0.007
R-squared Adj.  0.007
N               20207
Log Likelihood -22326.188
AIC             44656.376
==========================
Standard errors in
parentheses.
* p<.1, ** p<.05,
***p<.01
```

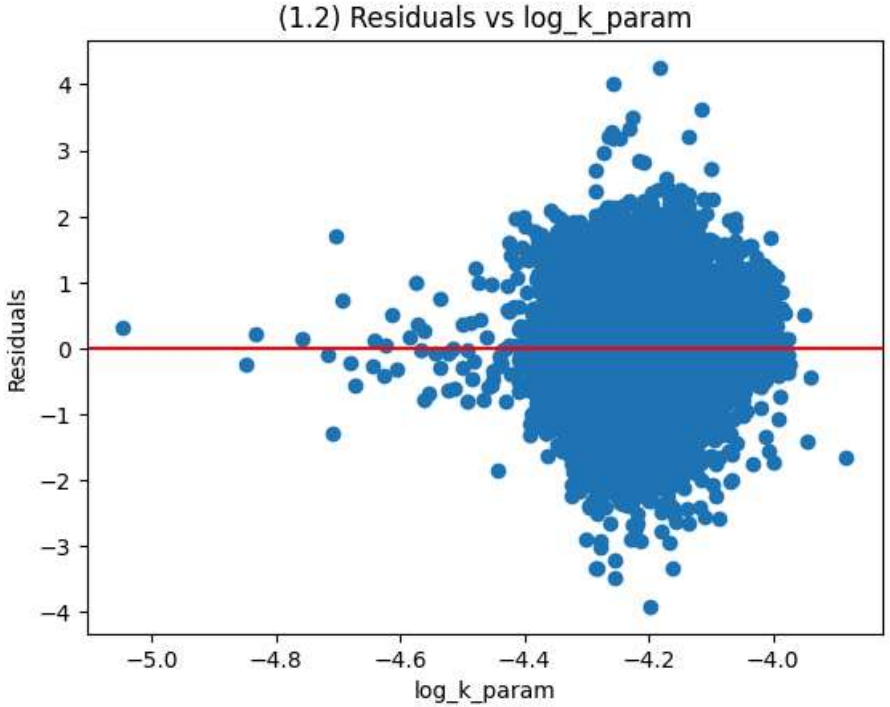
In [43]:
```
# Plotting the predicted vs actual values
plt.scatter(model1_log_results.fittedvalues, DATA['log_relPriceVar'], alpha=0.5)
plt.plot(DATA['log_relPriceVar'], DATA['log_relPriceVar'], color='red')
plt.xlabel('Fitted values')
plt.ylabel('Actual')
plt.title('(1.2) Fitted vs Actual')
plt.show()
```



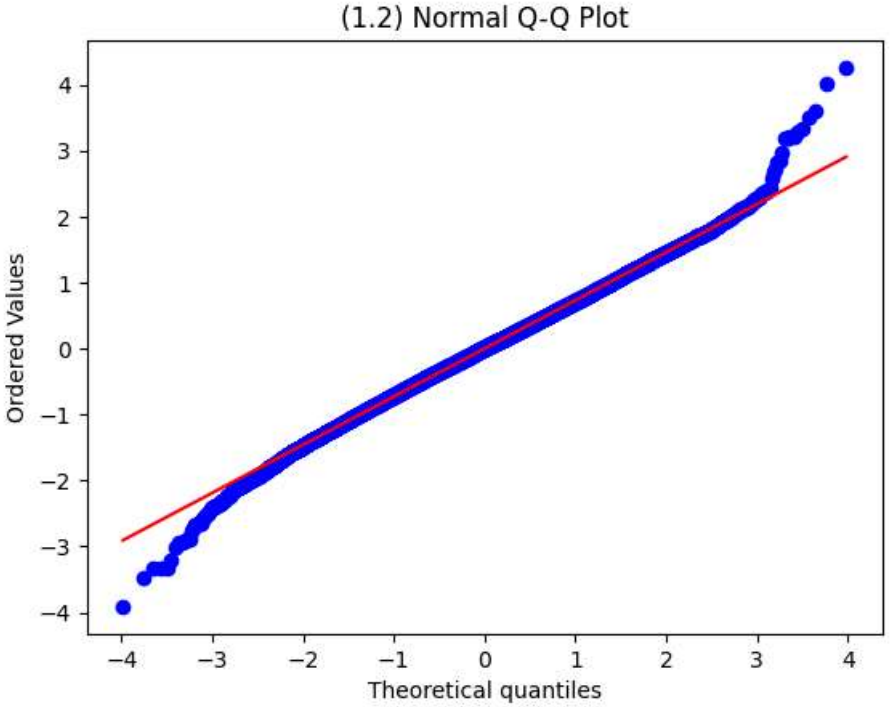
In [44]:
```
acf = smt.graphics.plot_acf(model1_log_results.resid, lags=40, alpha=0.05)
plt.xlabel('Lag')
plt.ylabel('ACF')
plt.title('(1.2) Autocorrelation Plot')
plt.show()
```



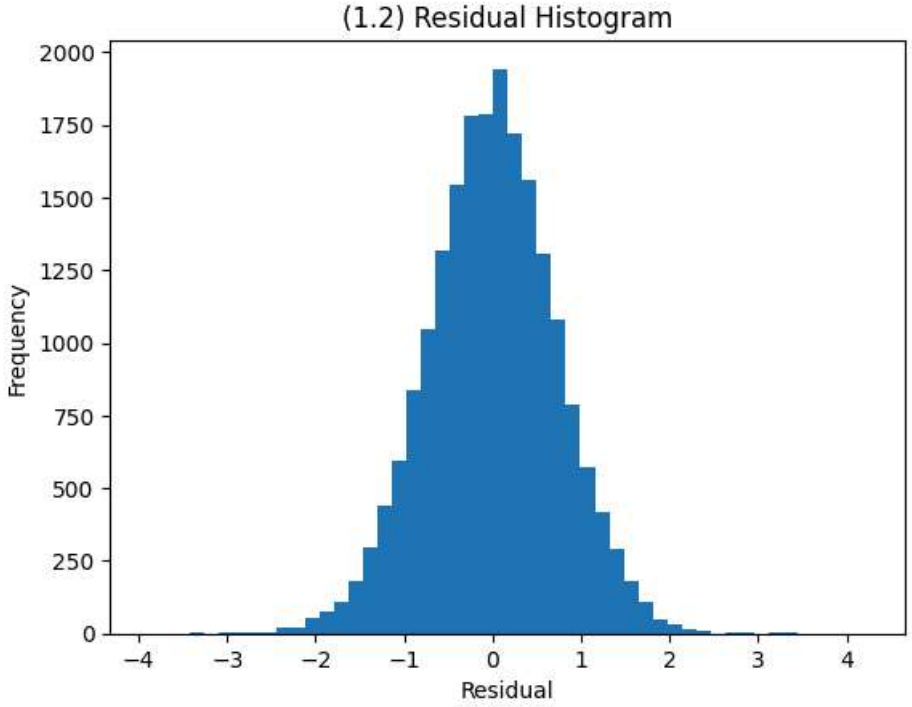
In [45]:
```
plt.scatter(model1_log_results.fittedvalues, model1_log_results.resid)
plt.axhline(y=0, color='r', linestyle='-')
plt.xlabel('log_k_param')
plt.ylabel('Residuals')
plt.title('(1.2) Residuals vs log_k_param')
plt.show()
```



In [46]:
```
stats.probplot(model1_log_results.resid, dist='norm', plot=plt)
plt.title('(1.2) Normal Q-Q Plot')
plt.show()
```



In [47]:
```
# Histogram residuals
plt.hist(model1_log_results.resid, bins=50)
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.title('(1.2) Residual Histogram')
plt.show()
```



In [48]:
```
results_table = summary_col([model1_results, model1_log_results],
                            stars=True,
                            float_format='%0.3f',
                            model_names=['(1.1)', '(1.2)'],
                            info_dict={'N': lambda x: f"{int(x.nobs)}",
                                       'Log Likelihood': lambda x: f"{x.llf:.3f}",
                                       'AIC': lambda x: f"{x.aic:.3f}"})

# Add significance levels and notes
results_table.add_title('Model 1: Price Variation vs k parameter')

# Print or save the results
print(results_table.as_text()) # Can do as latex
```

```
Model 1: Price Variation vs k parameter
====================================
                  (1.1)      (1.2)
------------------------------------
const          0.014***    -4.314***
               (0.000)     (0.009)
k_param        0.003***
               (0.000)
log_k_param                0.195***
                           (0.016)
R-squared      0.007       0.007
R-squared Adj. 0.007       0.007
AIC            -100823.352 44656.376
Log Likelihood 50413.676   -22326.188
N              20207       20207
====================================
Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
```

Not included: Model log index variation against log relative price variation

In [49]:
```python
# Remove inf and nan values
lin_reg_data = DATA.replace([np.inf, -np.inf], np.nan).dropna()

# Linreg of all data
X = lin_reg_data[['log_IndexVar']]
y = lin_reg_data['log_relPriceVar']

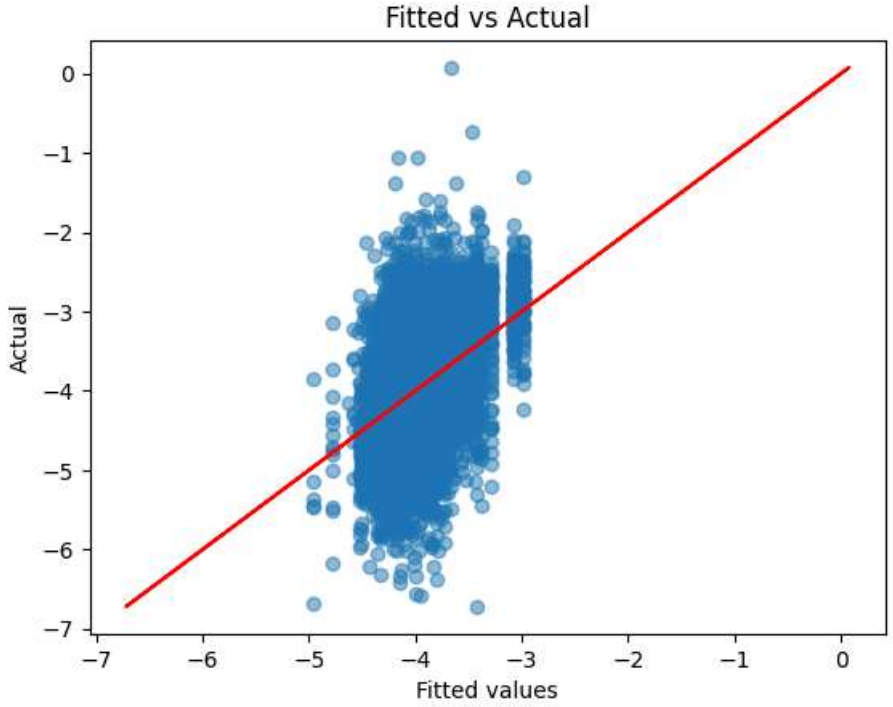
model = sm.OLS(y, sm.add_constant(X))
results = model.fit()

print(results.summary().as_text())
```

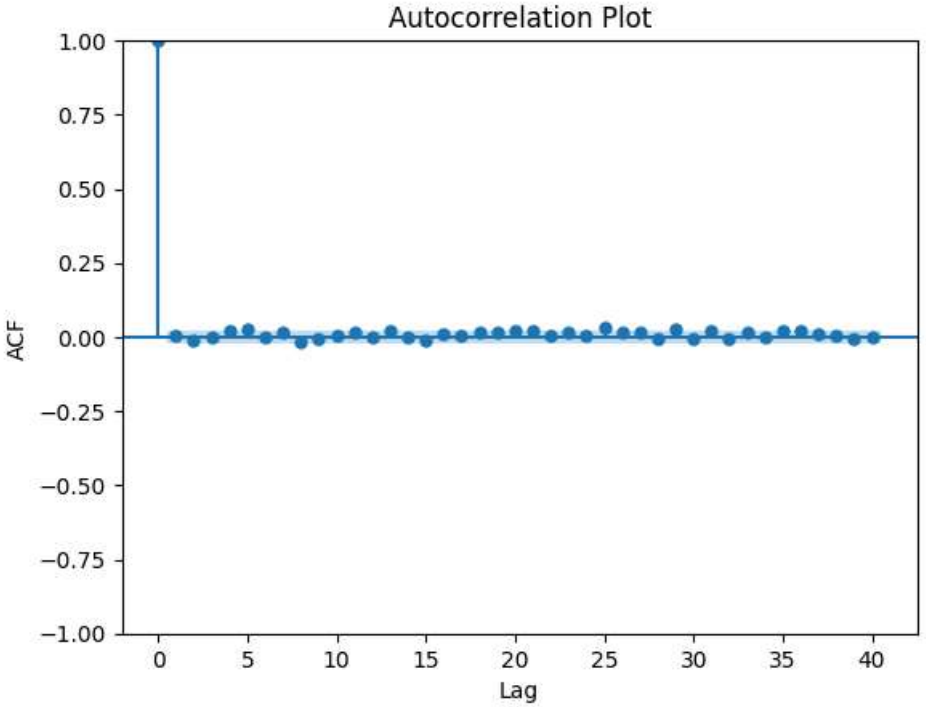
```
                            OLS Regression Results
==============================================================================
Dep. Variable:        log_relPriceVar   R-squared:                       0.169
Model:                            OLS   Adj. R-squared:                  0.169
Method:                 Least Squares   F-statistic:                     1635.
Date:                Thu, 05 Dec 2024   Prob (F-statistic):               0.00
Time:                        19:44:26   Log-Likelihood:                -7945.7
No. Observations:                8056   AIC:                         1.590e+04
Df Residuals:                    8054   BIC:                         1.591e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          -0.6985      0.080     -8.692      0.000      -0.856      -0.541
log_IndexVar    0.8319      0.021     40.432      0.000       0.792       0.872
==============================================================================
Omnibus:                       82.807   Durbin-Watson:                   1.995
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              138.597
Skew:                           0.030   Prob(JB):                     8.02e-31
Kurtosis:                       3.640   Cond. No.                         46.2
==============================================================================

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

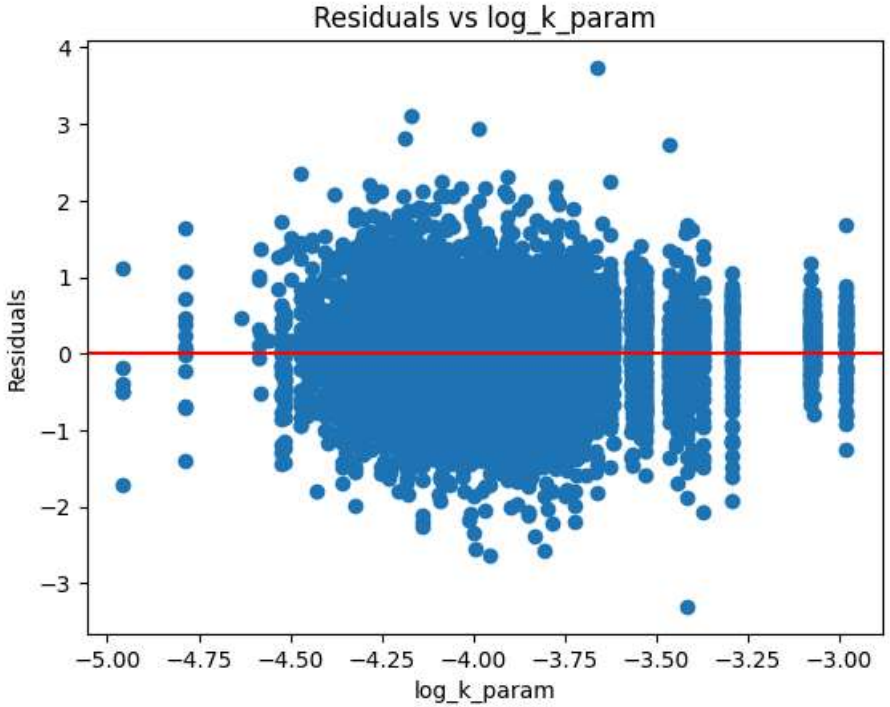
In [50]:
```
# Plotting the predicted vs actual values
plt.scatter(results.fittedvalues, lin_reg_data['log_relPriceVar'], alpha=0.5)
plt.plot(lin_reg_data['log_relPriceVar'], lin_reg_data['log_relPriceVar'], color='red')
plt.xlabel('Fitted values')
plt.ylabel('Actual')
plt.title('Fitted vs Actual')
plt.show()
```



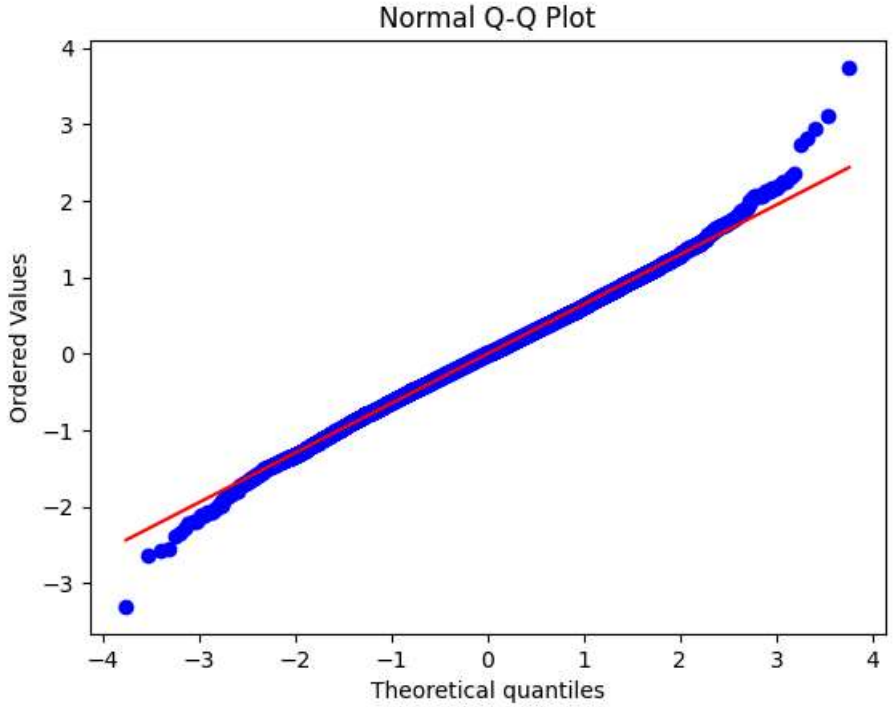
In [51]:
```
acf = smt.graphics.plot_acf(results.resid, lags=40, alpha=0.05)
plt.xlabel('Lag')
plt.ylabel('ACF')
plt.title('Autocorrelation Plot')
plt.show()
```



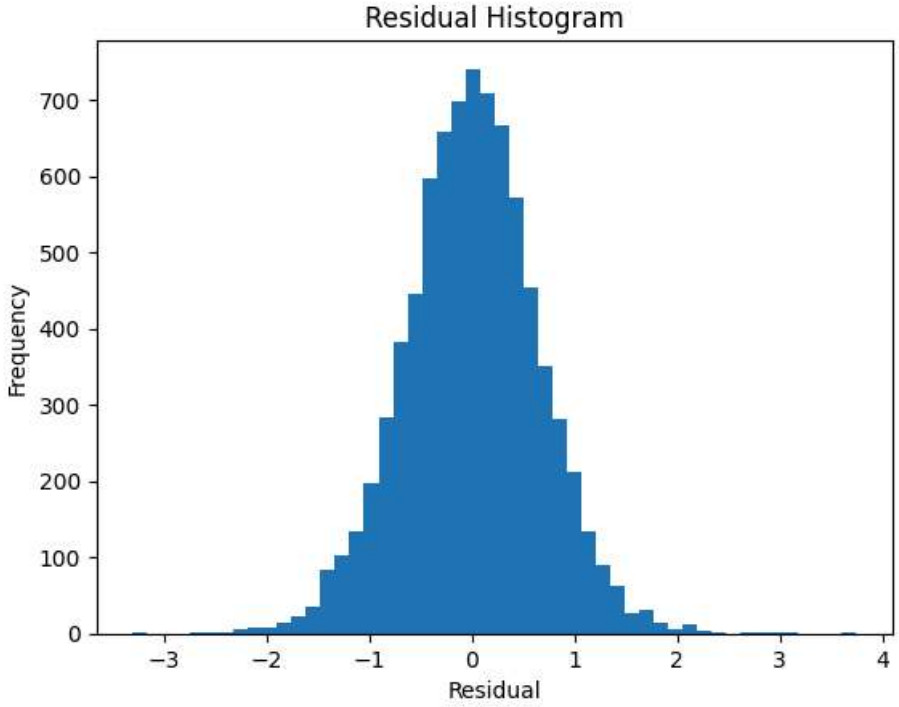
In [52]:
```
plt.scatter(results.fittedvalues, results.resid)
plt.axhline(y=0, color='r', linestyle='-')
plt.xlabel('log_k_param')
plt.ylabel('Residuals')
plt.title('Residuals vs log_k_param')
plt.show()
```



In [53]:
```
stats.probplot(results.resid, dist='norm', plot=plt)
plt.title('Normal Q-Q Plot')
plt.show()
```



In [54]:
```
# Histogram residuals
plt.hist(results.resid, bins=50)
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.title('Residual Histogram')
plt.show()
```



Model 2 log transformed

In [55]:
```
# Remove inf and nan values
lin_reg_data = DATA.replace([np.inf, -np.inf], np.nan).dropna()

# Linreg of all data
X = lin_reg_data[['log_k_param', 'log_IndexVar']]
y = lin_reg_data['log_relPriceVar']

model2_log = sm.OLS(y, sm.add_constant(X))
model2_log_results = model2_log.fit()

print(model2_log_results.summary().as_text())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        log_relPriceVar   R-squared:                       0.193
Model:                            OLS   Adj. R-squared:                  0.193
Method:                 Least Squares   F-statistic:                     966.1
Date:                Thu, 05 Dec 2024   Prob (F-statistic):               0.00
Time:                        19:44:28   Log-Likelihood:                -7823.8
No. Observations:                8056   AIC:                         1.565e+04
Df Residuals:                    8053   BIC:                         1.567e+04
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          -0.4671      0.081     -5.801      0.000      -0.625      -0.309
log_k_param     0.3572      0.023     15.728      0.000       0.313       0.402
log_IndexVar    0.9385      0.021     43.915      0.000       0.897       0.980
==============================================================================
Omnibus:                       85.963   Durbin-Watson:                   1.972
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              146.144
Skew:                          -0.023   Prob(JB):                     1.84e-32
Kurtosis:                       3.658   Cond. No.                         47.6
==============================================================================

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

In [56]:

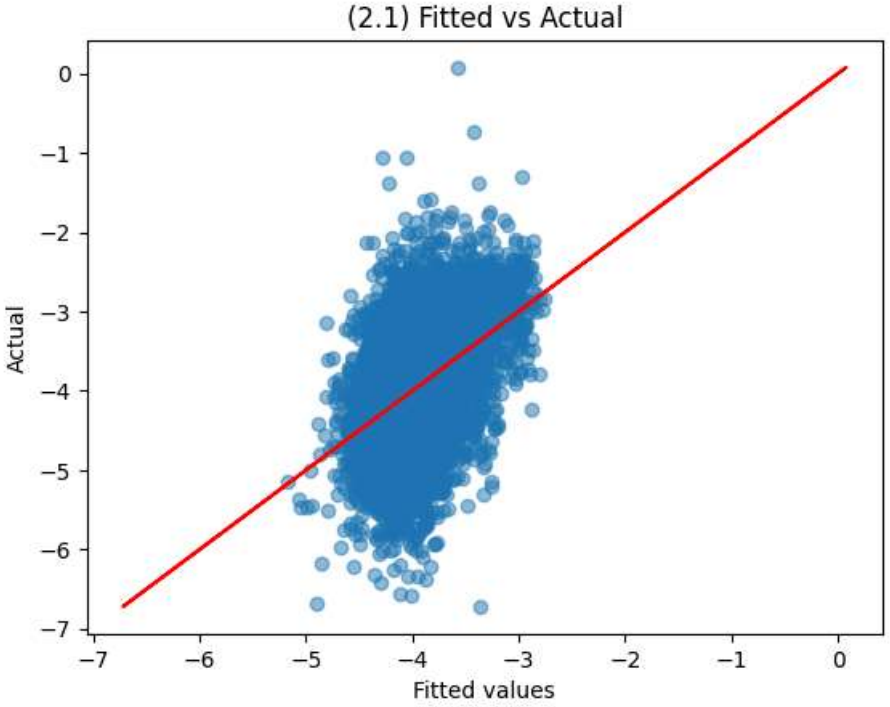
```python
results_table = summary_col([model2_log_results],
                            stars=True,
                            float_format='%0.3f',
                            model_names=['(2.1)'],
                            info_dict={'N': lambda x: f"{int(x.nobs)}",
                                       'Log Likelihood': lambda x: f"{x.llf:.3f}",
                                       'AIC': lambda x: f"{x.aic:.3f}"})

# Add significance levels and notes
results_table.add_title('Model 2: Logged Price Variation vs k parameter and Index Variance')

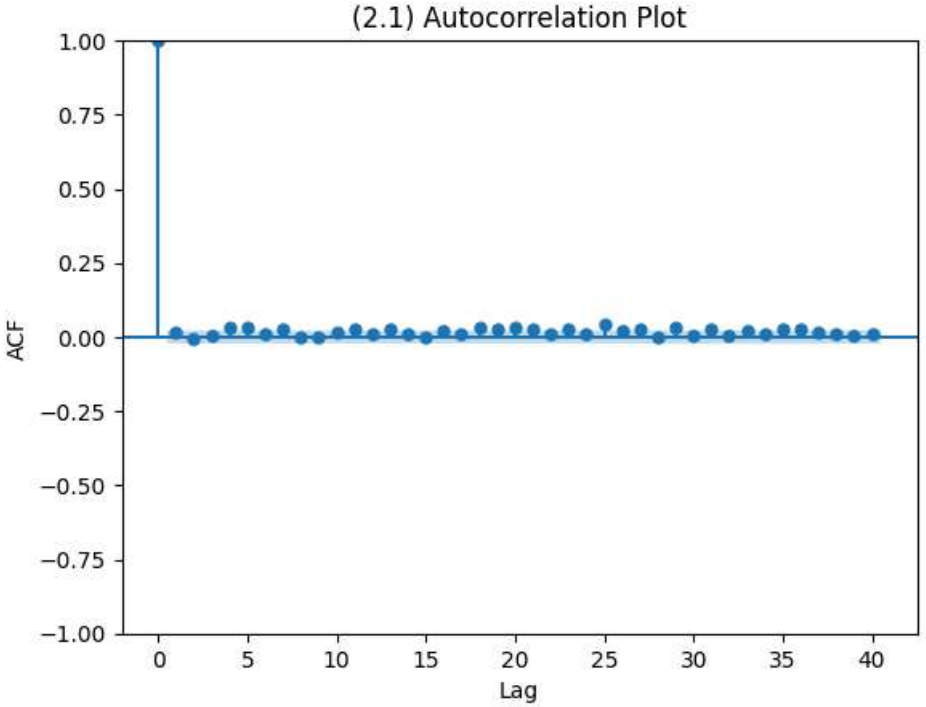
# Print or save the results
print(results_table.as_text()) # Can do as latex
```

```
Model 2: Logged Price Variation vs k parameter and Index Variance
========================
                 (2.1)
------------------------
const          -0.467***
               (0.081)
log_k_param    0.357***
               (0.023)
log_IndexVar   0.939***
               (0.021)
R-squared      0.193
R-squared Adj. 0.193
N              8056
Log Likelihood -7823.836
AIC            15653.672
========================
Standard errors in
parentheses.
* p<.1, ** p<.05,
***p<.01
```

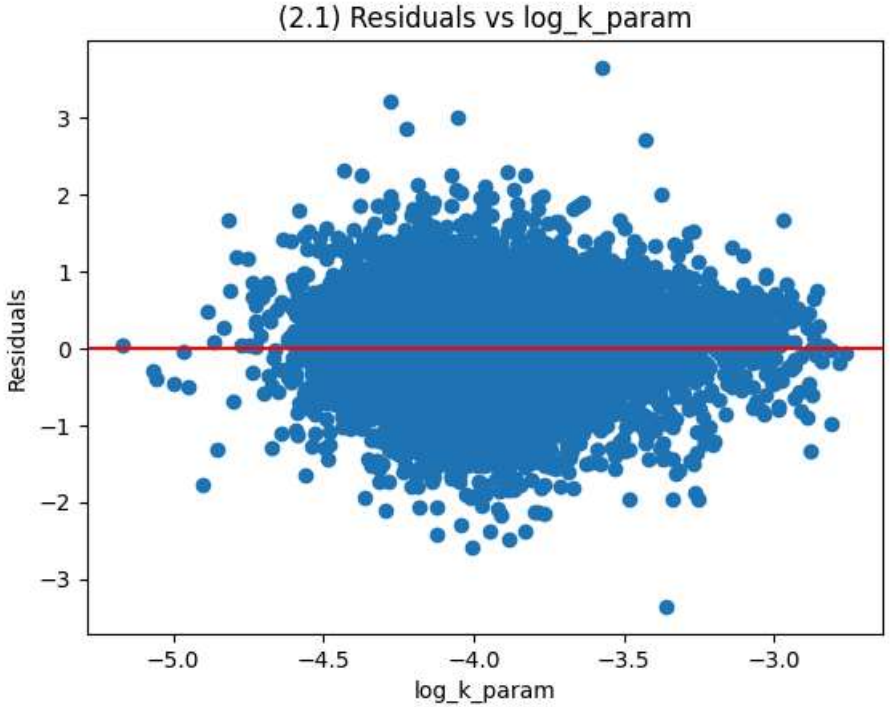
In [57]:
```python
# Plotting the predicted vs actual values
plt.scatter(model2_log_results.fittedvalues, lin_reg_data['log_relPriceVar'], alpha=0.5)
plt.plot(lin_reg_data['log_relPriceVar'], lin_reg_data['log_relPriceVar'], color='red')
plt.xlabel('Fitted values')
plt.ylabel('Actual')
plt.title('(2.1) Fitted vs Actual')
plt.show()
```



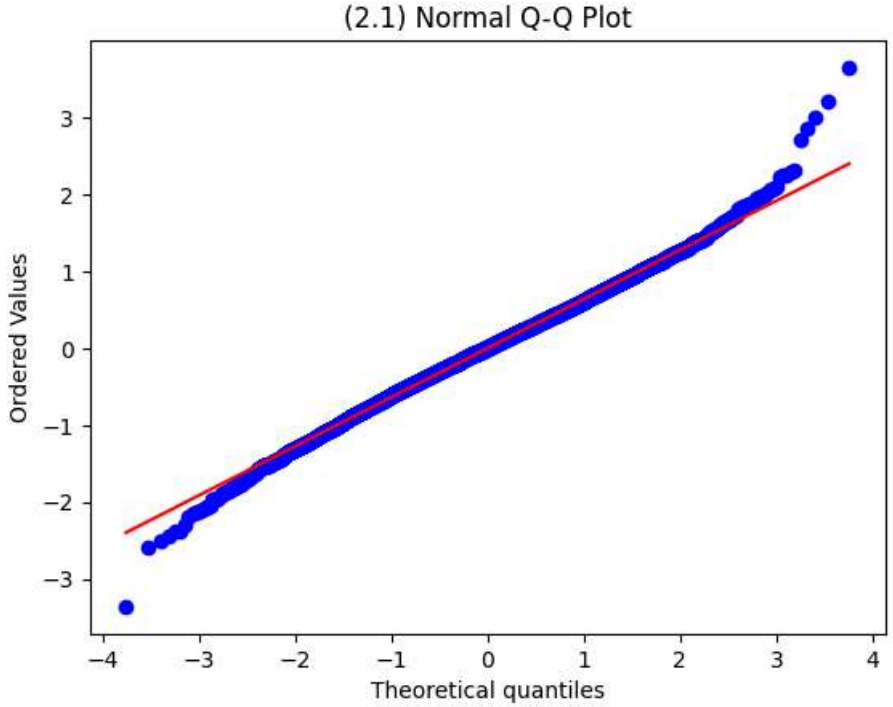
In [58]:
```
acf = smt.graphics.plot_acf(model2_log_results.resid, lags=40, alpha=0.05)
plt.xlabel('Lag')
plt.ylabel('ACF')
plt.title('(2.1) Autocorrelation Plot')
plt.show()
```



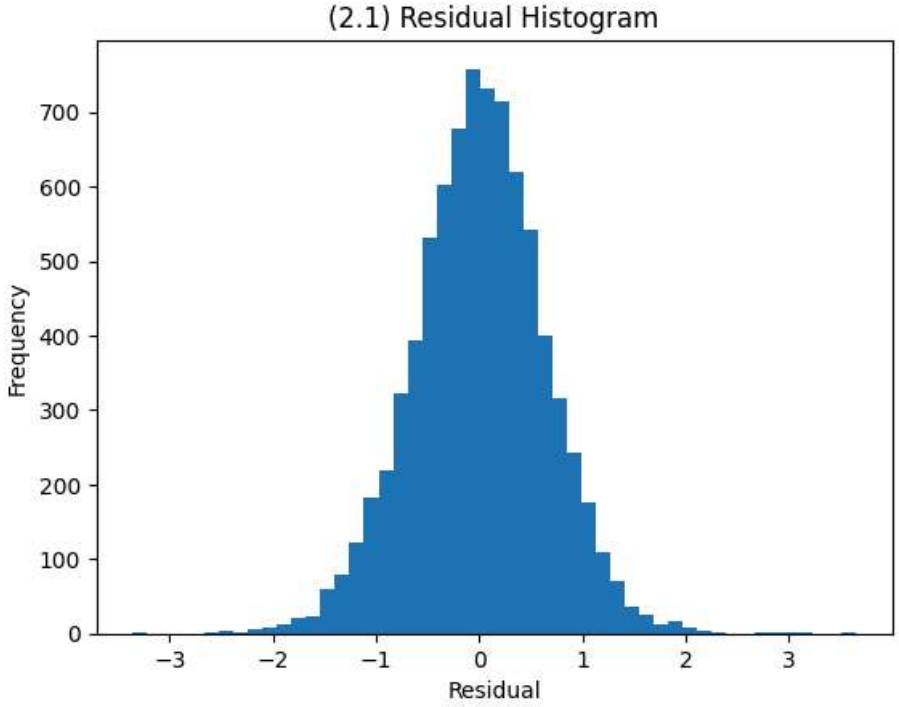
In [59]:
```
plt.scatter(model2_log_results.fittedvalues, model2_log_results.resid)
plt.axhline(y=0, color='r', linestyle='-')
plt.xlabel('log_k_param')
plt.ylabel('Residuals')
plt.title('(2.1) Residuals vs log_k_param')
plt.show()
```



In [60]:
```
stats.probplot(model2_log_results.resid, dist='norm', plot=plt)
plt.title('(2.1) Normal Q-Q Plot')
plt.show()
```



In [61]:
```
# Histogram residuals
plt.hist(model2_log_results.resid, bins=50)
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.title('(2.1) Residual Histogram')
plt.show()
```



Model 3 log transformed

In [62]:
```
# Remove inf and nan values
lin_reg_data = DATA.replace([np.inf, -np.inf], np.nan).dropna()

# Linreg of all data
X = lin_reg_data[['log_k_param', 'log_IndexVar', 'norTotalVol']]
y = lin_reg_data['log_relPriceVar']

model3_log = sm.OLS(y, sm.add_constant(X))
model3_log_results = model3_log.fit()

print(model3_log_results.summary().as_text())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        log_relPriceVar   R-squared:                       0.241
Model:                            OLS   Adj. R-squared:                  0.241
Method:                 Least Squares   F-statistic:                     854.0
Date:                Thu, 05 Dec 2024   Prob (F-statistic):               0.00
Time:                        19:44:30   Log-Likelihood:                -7577.3
No. Observations:                8056   AIC:                         1.516e+04
Df Residuals:                    8052   BIC:                         1.519e+04
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          -0.9940      0.082    -12.194      0.000      -1.154      -0.834
log_k_param     0.2707      0.022     12.109      0.000       0.227       0.315
log_IndexVar    0.8290      0.021     38.941      0.000       0.787       0.871
norTotalVol     0.1628      0.007     22.545      0.000       0.149       0.177
==============================================================================
Omnibus:                      116.344   Durbin-Watson:                   1.971
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              184.050
Skew:                          -0.138   Prob(JB):                     1.08e-40
Kurtosis:                       3.687   Cond. No.                         50.9
==============================================================================

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

In [63]:
```python
results_table = summary_col([model3_log_results],
                            stars=True,
                            float_format='%0.3f',
                            model_names=['(3.1)'],
                            info_dict={'N': lambda x: f"{int(x.nobs)}",
                                       'Log Likelihood': lambda x: f"{x.llf:.3f}",
                                       'AIC': lambda x: f"{x.aic:.3f}"})

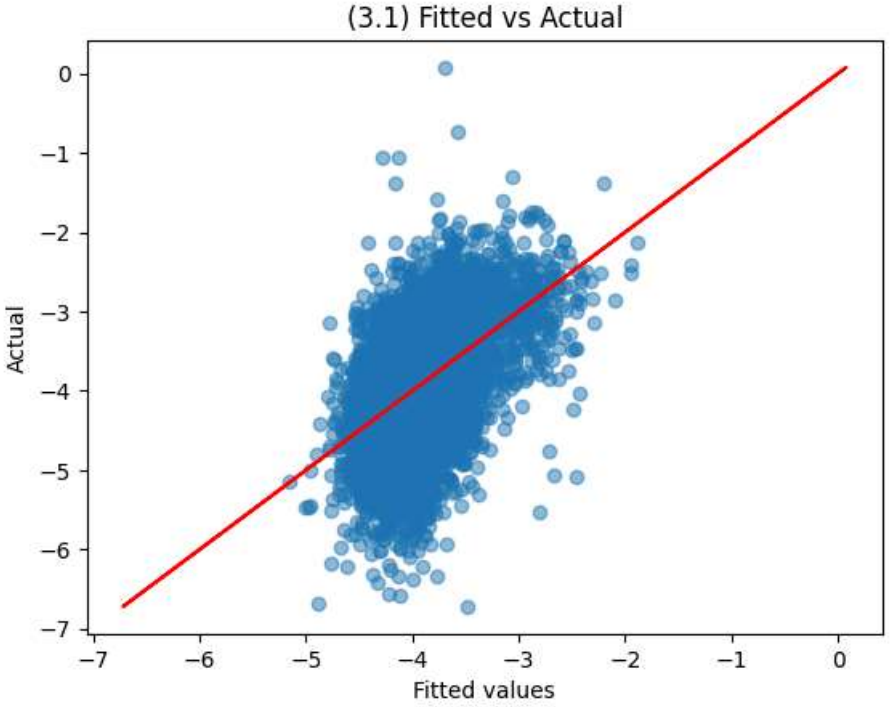
# Add significance levels and notes
results_table.add_title('Model 3: Logged Price Variation')

# Print or save the results
print(results_table.as_text()) # Can do as latex
```

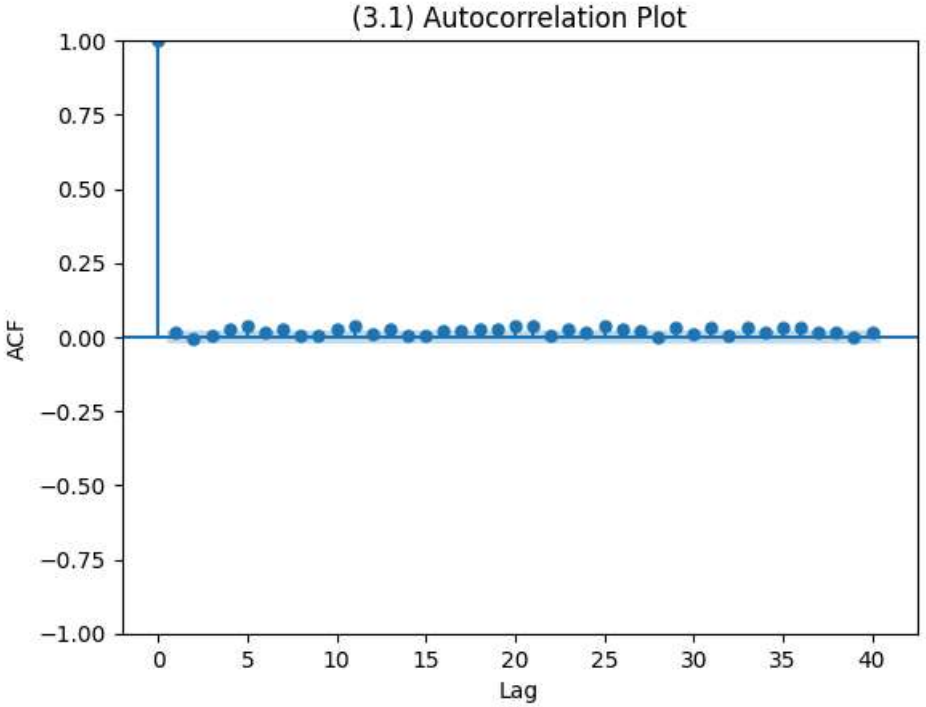
```
Model 3: Logged Price Variation
========================
                 (3.1)
------------------------
const          -0.994***
               (0.082)
log_k_param    0.271***
               (0.022)
log_IndexVar   0.829***
               (0.021)
norTotalVol    0.163***
               (0.007)
R-squared      0.241
R-squared Adj. 0.241
N              8056
Log Likelihood -7577.269
AIC            15162.538
========================
Standard errors in
parentheses.
* p<.1, ** p<.05,
***p<.01
```

In [64]:

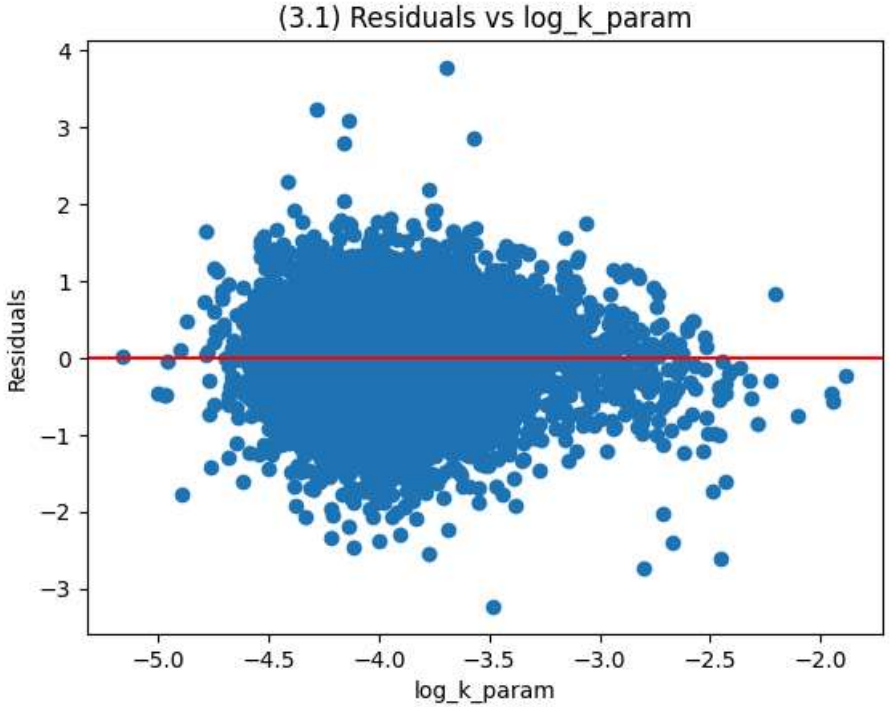
```python
# Plotting the predicted vs actual values
plt.scatter(model3_log_results.fittedvalues, lin_reg_data['log_relPriceVar'], alpha=0.5)
plt.plot(lin_reg_data['log_relPriceVar'], lin_reg_data['log_relPriceVar'], color='red')
plt.xlabel('Fitted values')
plt.ylabel('Actual')
plt.title('(3.1) Fitted vs Actual')
plt.show()
```



In [65]:
```
acf = smt.graphics.plot_acf(model3_log_results.resid, lags=40, alpha=0.05)
plt.xlabel('Lag')
plt.ylabel('ACF')
plt.title('(3.1) Autocorrelation Plot')
plt.show()
```

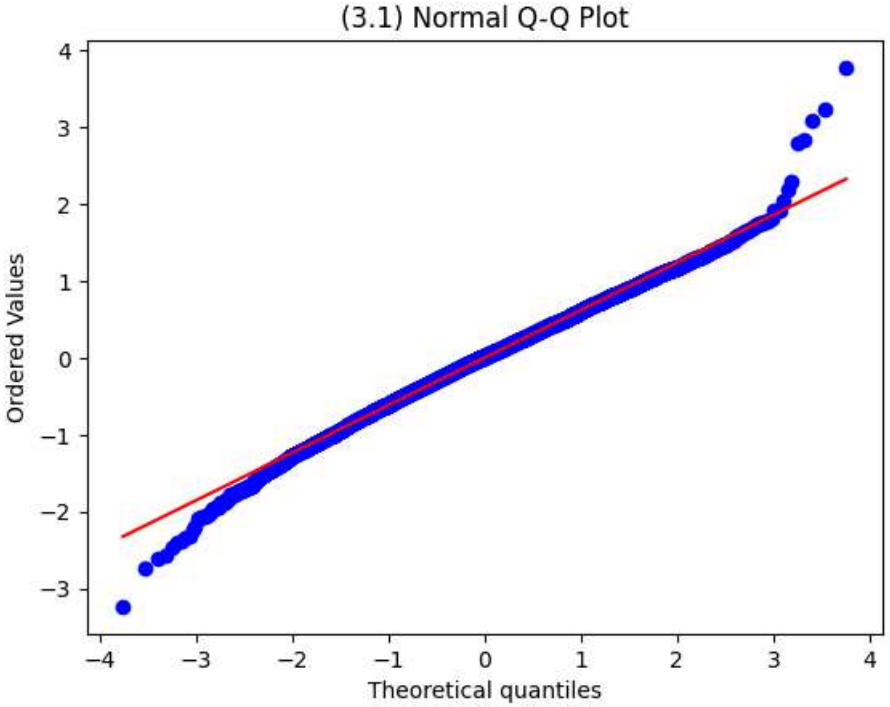


In [66]:
```
plt.scatter(model3_log_results.fittedvalues, model3_log_results.resid)
plt.axhline(y=0, color='r', linestyle='-')
plt.xlabel('log_k_param')
plt.ylabel('Residuals')
plt.title('(3.1) Residuals vs log_k_param')
plt.show()
```

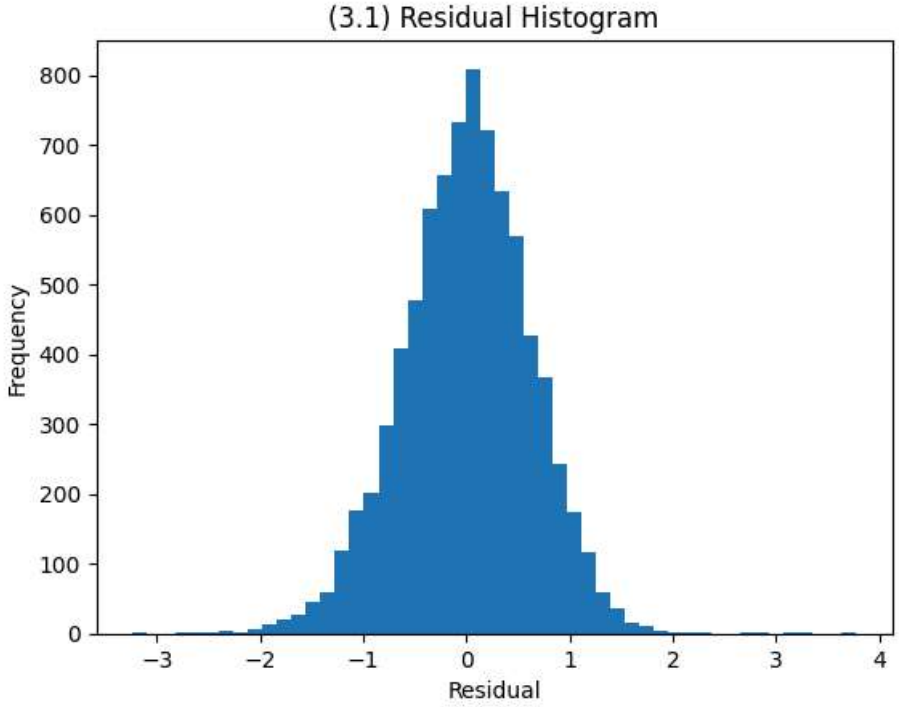


In [67]:
```
import scipy.stats as stats

stats.probplot(model3_log_results.resid, dist='norm', plot=plt)
plt.title('(3.1) Normal Q-Q Plot')
plt.show()
```



In [68]:
```
# Histogram residuals
plt.hist(model3_log_results.resid, bins=50)
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.title('(3.1) Residual Histogram')
plt.show()
```



Not included: Model relative price variation against log explanitory variables

In [69]:
```
# Remove inf and nan values
lin_reg_data = DATA.replace([np.inf, -np.inf], np.nan).dropna()

# Linreg of all data
X = lin_reg_data[['log_k_param', 'log_IndexVar', 'norTotalVol']]
y = lin_reg_data['relPriceVar']

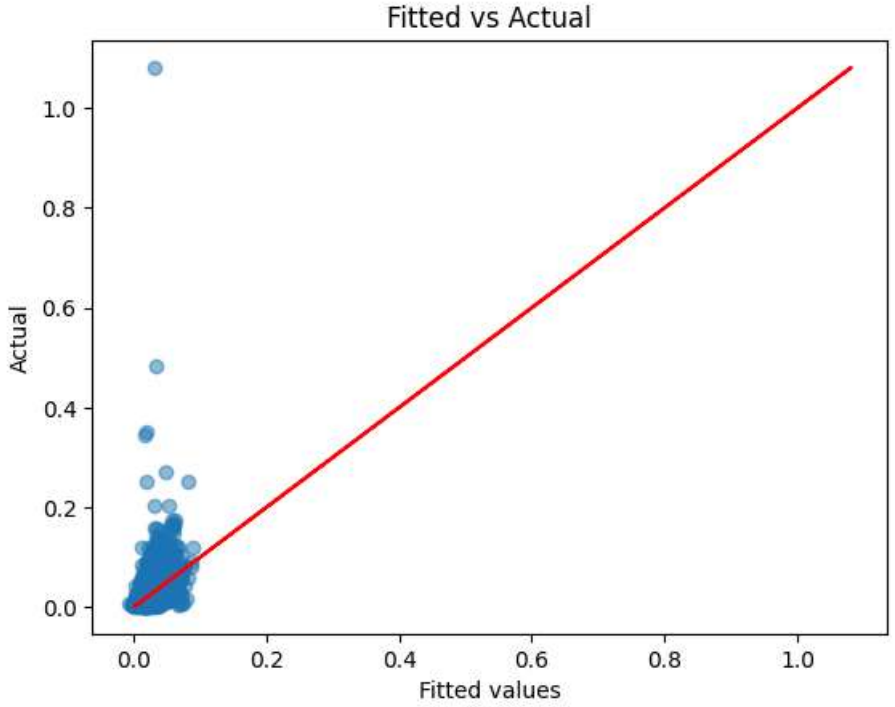
model = sm.OLS(y, sm.add_constant(X))
results = model.fit()

print(results.summary().as_text())
```

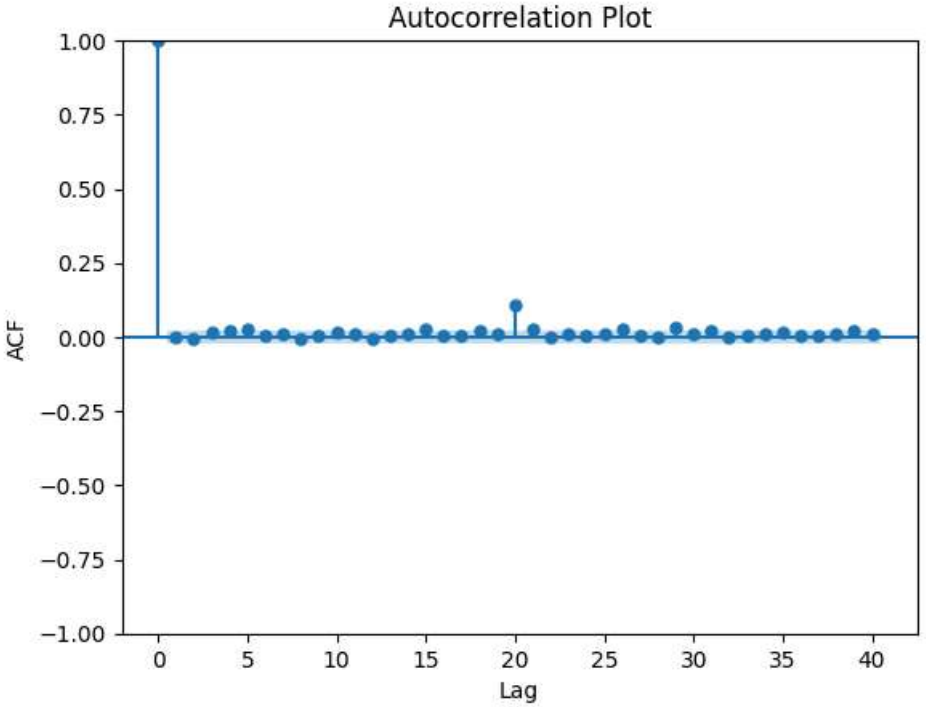
```
                            OLS Regression Results
==============================================================================
Dep. Variable:            relPriceVar   R-squared:                       0.175
Model:                            OLS   Adj. R-squared:                  0.175
Method:                 Least Squares   F-statistic:                     569.1
Date:                Thu, 05 Dec 2024   Prob (F-statistic):               0.00
Time:                        19:44:32   Log-Likelihood:                 19346.
No. Observations:                8056   AIC:                        -3.868e+04
Df Residuals:                    8052   BIC:                        -3.866e+04
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0992      0.003     34.393      0.000       0.093       0.105
log_k_param    0.0079      0.001     10.044      0.000       0.006       0.009
log_IndexVar   0.0214      0.001     28.374      0.000       0.020       0.023
norTotalVol    0.0058      0.000     22.590      0.000       0.005       0.006
==============================================================================
Omnibus:                    16564.856   Durbin-Watson:                   2.003
Prob(Omnibus):                  0.000   Jarque-Bera (JB):        162064596.142
Skew:                          16.761   Prob(JB):                         0.00
Kurtosis:                     697.039   Cond. No.                         50.9
==============================================================================

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

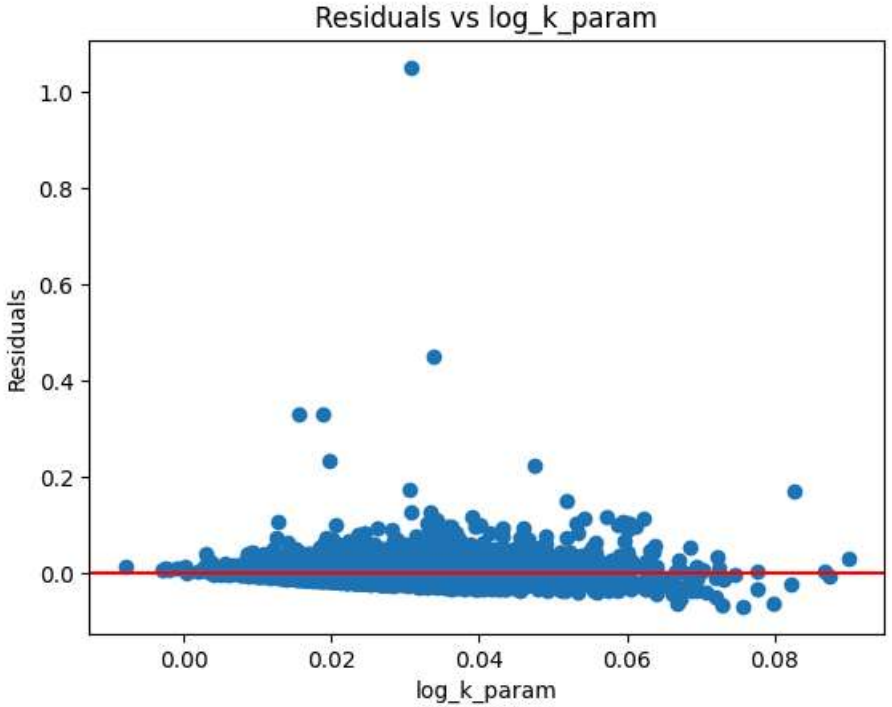
In [70]:
```python
# Plotting the predicted vs actual values
plt.scatter(results.fittedvalues, lin_reg_data['relPriceVar'], alpha=0.5)
plt.plot(lin_reg_data['relPriceVar'], lin_reg_data['relPriceVar'], color='red')
plt.xlabel('Fitted values')
plt.ylabel('Actual')
plt.title('Fitted vs Actual')
plt.show()
```



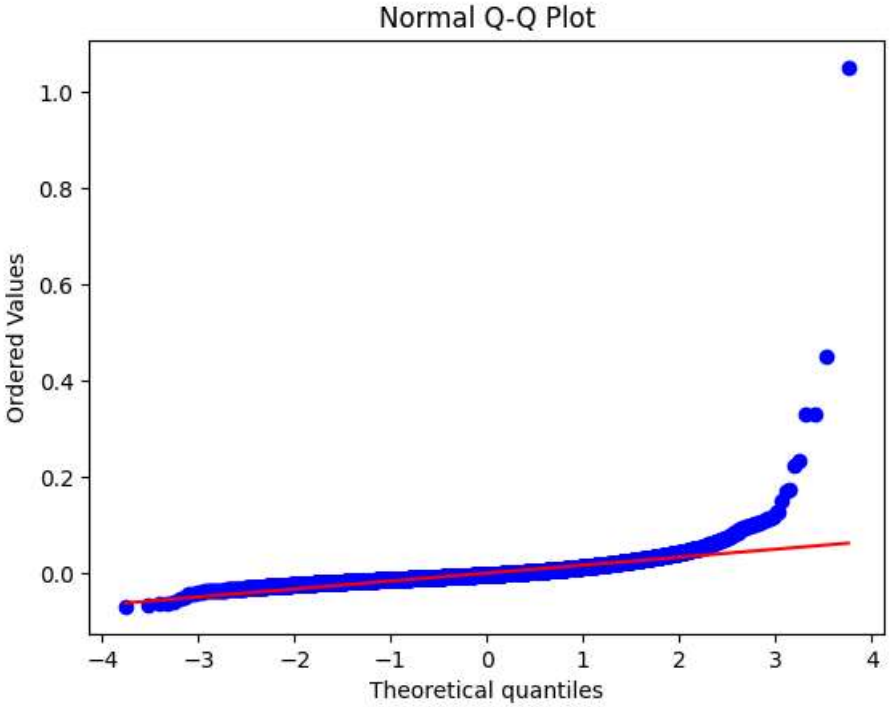
In [71]:
```
acf = smt.graphics.plot_acf(results.resid, lags=40, alpha=0.05)
plt.xlabel('Lag')
plt.ylabel('ACF')
plt.title('Autocorrelation Plot')
plt.show()
```



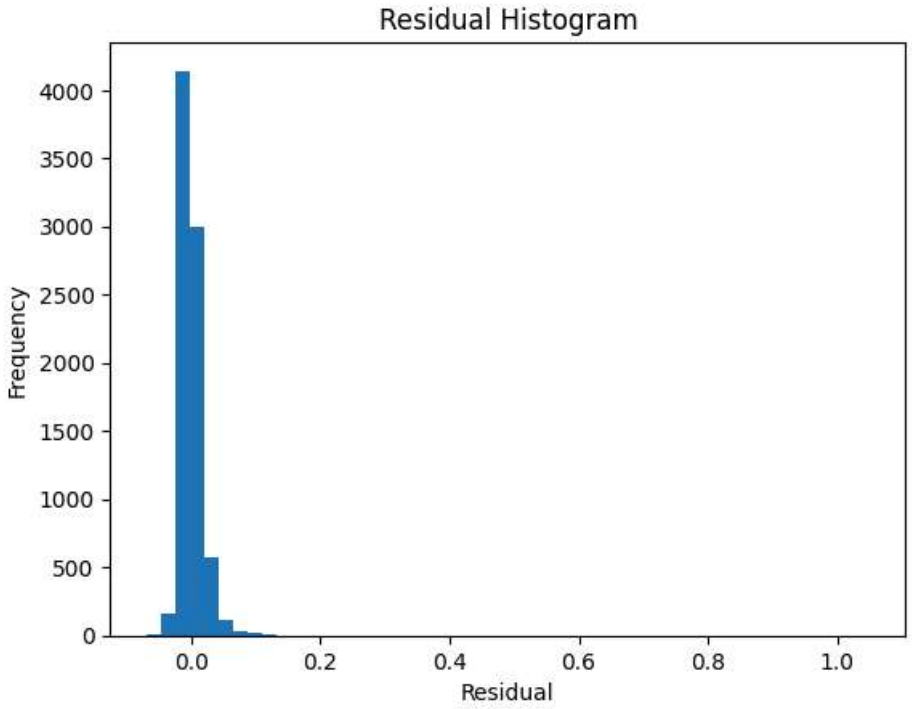
In [72]:
```
plt.scatter(results.fittedvalues, results.resid)
plt.axhline(y=0, color='r', linestyle='-')
plt.xlabel('log_k_param')
plt.ylabel('Residuals')
plt.title('Residuals vs log_k_param')
plt.show()
```



In [73]:
```
stats.probplot(results.resid, dist='norm', plot=plt)
plt.title('Normal Q-Q Plot')
plt.show()
```



In [74]:
```
# Histogram residuals
plt.hist(results.resid, bins=50)
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.title('Residual Histogram')
plt.show()
```



Not included: Random forest regression

In [75]:

```
# Random forest regression
from sklearn.ensemble import RandomForestRegressor

# Remove inf and nan values
lin_reg_data = DATA.replace([np.inf, -np.inf], np.nan).dropna()

# Random Forest Regression
X = lin_reg_data[['log_k_param', 'log_IndexVar', 'norTotalVol']]
y = lin_reg_data['log_relPriceVar']

# Initialize the model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# Split data set into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

# Fit the model
rf_model.fit(X_train, y_train)

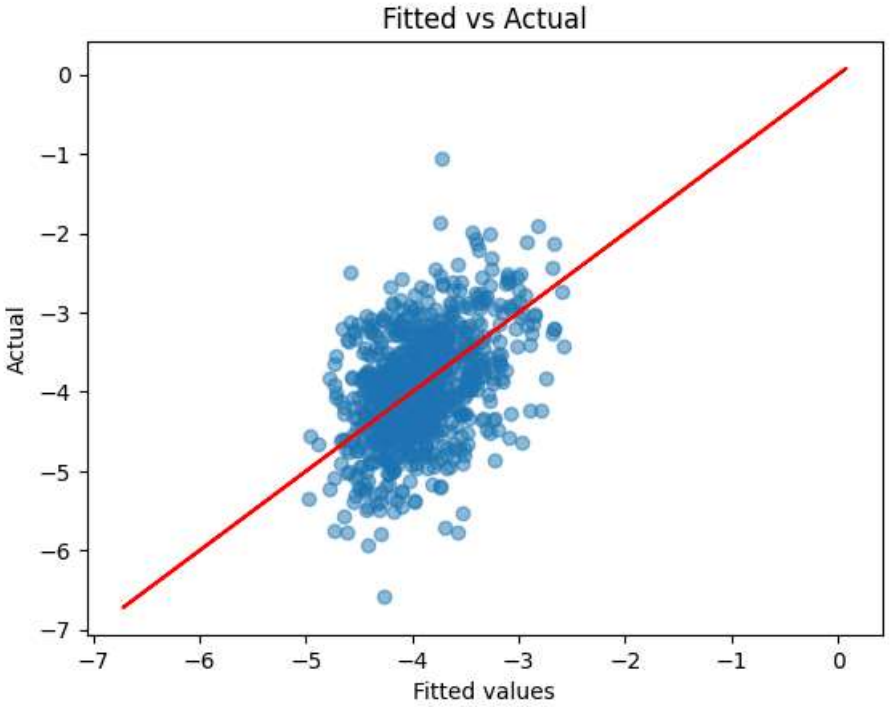
# Predict
y_pred = rf_model.predict(X_test)
```

```
# Calculate R-squared
r_squared = rf_model.score(X_test, y_test)

# Print the R-squared
print(f"R-squared: {r_squared}")

# Plotting the predicted vs actual values
plt.scatter(y_pred, y_test, alpha=0.5)
plt.plot(y, y, color='red')
plt.xlabel('Fitted values')
plt.ylabel('Actual')
plt.title('Fitted vs Actual')
plt.show()
```

```
R-squared: 0.15553498948965572
```



In [ ]:

Appendix 2

Data is maually downloaded from SEC: https://www.sec.gov/about/marketstructuredata-security

In [7]:
```
import pandas as pd


years = [2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023 2024]

df_data = pd.DataFrame()

for year in years:
    for quarter in range(1, 5):
        # Open the file with pandas
        if year == 2024 and quarter == 3:
            break

        file = f"q{quarter}_{year}_all.csv"
        df_quarter = pd.read_csv(file)
        df_data = pd.concat([df_data, df_quarter])
```

In [8]:
```
df_data.head()
```

Out[8]:

	Date	Security	Ticker	McapRank	TurnRank	VolatilityRank	PriceRank	LitVol('000)	OrderVol('000)	Hidden	TradesForHidd
0	20120103.0	Stock	A	10.0	7.0	4.0	9.0	2034.226	138332.959	960.0	1611
1	20120103.0	Stock	AA	10.0	10.0	3.0	5.0	23439.037	369286.544	5985.0	4774
2	20120103.0	Stock	AACC	4.0	1.0	10.0	3.0	4.080	1693.565	5.0	2
3	20120103.0	Stock	AAME	2.0	1.0	7.0	2.0	0.580	178.206	0.0	
4	20120103.0	Stock	AAN	8.0	4.0	2.0	8.0	180.055	13920.751	172.0	157

In [9]:
```
df_data.tail()
```

Out[9]:

	Date	Security	Ticker	McapRank	TurnRank	VolatilityRank	PriceRank	LitVol('000)	OrderVol('000)	Hidden	TradesFc
464159	20240628.0	ETF	ZTAX	1.0	4.0	4.0	2.0	3.452	775.338	28.0	
464160	20240628.0	ETF	ZTEN	1.0	3.0	1.0	3.0	0.100	3113.512	0.0	
464161	20240628.0	ETF	ZTRE	1.0	1.0	1.0	4.0	0.100	991.340	0.0	
464162	20240628.0	ETF	ZTWO	1.0	4.0	1.0	4.0	2.410	530.581	0.0	
464163	20240628.0	ETF	ZZZ	1.0	4.0	4.0	1.0	1.040	13083.370	0.0	

In [11]:
```
df_data.to_csv("all_data.csv", index=False)
```

Appendix 3

In [ ]:
```
import requests
from requests.auth import HTTPBasicAuth

# Client ID and Client Secret
client_id = 'ccb99f25c6274399882b'
client_secret = '******************'

# FINRA Identity Platform (FIP) endpoint
token_url = 'https://ews.fip.finra.org/fip/rest/ews/oauth2/access_token?grant_type=client_credentials'

# Obtain the access token
response = requests.post(token_url, auth=HTTPBasicAuth(client_id, client_secret))
response_data = response.json()
access_token = response_data['access_token']
```

In [ ]:
```
response_data
```

Out[ ]:

```
{'access_token': '*AAJTSQACMDIABHR5cGUAA0pXVAACUzEAAjAx*eyJ0eXAiOiJKV1QiLCJjdHkiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ZXlKMGVYQWl
PaUpLVjFRaUxDSmxibU1pT2lKQk1USTRRMEpETFVoVE1qVTJJaXdpWVd4bklqb2laR2x5SW4wLi51SC1LZlNEVEVmVkhzUTk2QmJ3dl9BLnE3LTNrbU5jTnJ
hblZTNEZId0tJMzUycHVXS1VVOXhob0RFOU9HaDJkclU4QWcyV0o3d19WMERuOHlmVjRqU3djem14dXNxRkhZU2JCZHlhRkV1YXBxNks0RWVRSUM4SXF1SEJ
xaUprTEhlOXo5ZXhrMkxqdjJiczRZMlg3QzRnSk1CX3plUVhCM0x3anpvZ2ZUbUNOeUhkSUUxeUw3cFNQdGRsNS1ILTE4aHgyWWZCcy1RaUVlWnRwdS1YT1d
paDhibllJVjdVTDBhUGd0RnlvMGtOUWtZdHNicVM2cHIta1Z2NUxHcm9ndVgxWHdmVUhuMkpZeTNsMnZ2b1Bmd3pURGV5S0x2TGgyRlEtdWxMb0g4NmtfTEl
hQl90V1FYVEgwMmNXT0M1QVFnb2dUMEppY2tTYXM5UGFLNVdUX0NQY05EaHFwcXdpcm5MdTVwZW1RRy1hRG04bVlPZmJwZWFJSk9JM2pMdldtMk96VWlXR2R
qMGtLdXNNOGo3aWt0dVE5UGtiSFQtXzFkU3V6Q0FNUmhhbEhCRTdlYVVyQlF5WXNIdGFqX1VBejVxTTdxUmtodzdLR1VySFZBdVRHOVYtOF9wVWg4YlAyenB
SLVlaZTIxMnBySjNrRkFndDRXSk5PR2dtdFBKZDlkT0VFMk1MX2NuWkNSbFlwMUk5UkVnY1EyWUxZcnpsZnEyS2xqdWFWZzRGQXdBb2lOVUpkaEZUS1hXRVN
fT19UU2U4ZDdsYWw5WjNsdGd4LUdjbTVmc1BBMzNHTlYwUzY4VHUzSjdnVmNveExpVkQ4SUUycENkdEFMamcxZGtkalQtZGV6NmVmeG5tSF9NV0VwRDRtNFB
JcURUWVBJamd0Vk9KN0w0S1hkTGp5MHgzbDFKbTk1ZkVueWhJQzhnREhZaGM5SjNpRXR2R3hrTVVwMUNkU1c3eDA0ZDMzYlBVaDg3MlJhWThEaFFMV0N2aUt
3QUMtUmZSLTMtbUNPU2M5WTBHNWpNQm56dWowU3FKZlU0ZlJDcjRqMlBrUGp3TUdhYTVaSFI0Ul9tS0ViZktBbWx6Wi1iTlF5YzdKNEYtdnNPVEFrZE9PYnN
ZSkczcTdyMWpIRGxzdGxYZ0prTGExLWhTUHQyOWFob21oR3dhdUdTOU10a1J5dTBRYlNmRXpHU1Z5NVBrNW1xaWxjOG16Y1J6OU1hSURxVWEtUG9KZEpQVDR
4MkRKSjRNM1kyeW92VEdkN056ZXd3Y04xcXVDNTF4VzRUTmctSTFxeldnczRkUG1iZGVLeS03cHVFRy1heWVpTjRDYjRFT2pweGF2Y0hHRlpJemU1ZWNjTXd
yOUtUZkJGNThndEFtMTluRTNsZ3llR3BsVURsdHJURnBQY3hkeW1NRDI1d1J4TlZjTE83UjQycW04NGZmTFczRko3aHZpdjViSEVOZ0pVRkpPWFVPRUlhZXA
xcGNDT2hqXy15MVg5R09EZXdRLVVrNHExVXI2Z1UzTDVrY2FUMzA0MTYwOTFvWkV4dWFCUm5MRnAycDI4ZldWM0hSREllTk4yU0JZU01ZOEhTcTg5LXhQaFF
DbWtseTlHUmpaVTVpZ0dSQjRGaHEteUhxMlc3QVNXNE11bm0wVjh5U3NGV0h1ckRuMUJabTkxN1FTdnlLN3l0YTVnQ2VEVWpSOUJ3Z1lHOC1KODNmTm9PSTN
CajZnWjNYaENxYVRTX2RubjZSNW43RzRJUFRIYjlETVJBMXhjVkt1ZHZKeEFJRlhCRkVMcy1fSTdsaHJCSFUya0dSTk11MUJyd2dKb2IyZklDc1FWQ3dJVjd
MdHotQ3J6YUk4VDV4cmZQcXg2Tmpfa2ZfVFBYU0FPMXFkVVBiLVh1bkpkWm1sY3Ztd3BkcUtGZ3U2aTNwanRrX2FmaFpZMmhGY3J2RUFROE1zNVVzelU2V1h
wWkFzNHVWMko1WU9kUDRiYk9mRjJaUF9fUGJMb2l2OUZLZmpDZ2g0QmVZN213ZUNJSzJydWhmUWw1Wl9tRExGRnpXY2h0TldxdmFUSkljN1dVQmR5Vi5lUER
sZ1pnNXFqUzZabVpJMXFTRUNn.et5AlhOA94yxEyI9cTxUKG01UoFKatKZnG0d0QyaQ1Y',
 'scope': 'any',
 'token_type': 'Bearer',
 'expires_in': '43169'}
```


Get data function

In the url you sepcify the dataset.
Booth:
https://api.finra.org/data/group/otcMarket/name/weeklySummary\ and
https://api.finra.org/data/group/otcMarket/name/weeklySummaryHistoric\ have been used to gather data. FINRA moves data to the historic dataset week for week.

In [331…
```
import requests

# Define the API function

def get_hist_data(offset, record_limit, week_start_date):
    # API endpoint
    url = "https://api.finra.org/data/group/otcMarket/name/weeklySummary" # Historic

    # Headers for authentication and content type
    headers = {
        "Authorization": f"Bearer {access_token}",
        "Content-Type": "application/json",
        "Accept": "application/json"
    }

    # Request payload with filtering
    payload = {
        "limit": record_limit,  # Limits the number of records returned
        'offset': offset,
        "fields": [
            "issueSymbolIdentifier",
```

```
            "totalWeeklyShareQuantity",
            "weekStartDate",
            "marketParticipantName"
        ],
        "compareFilters": [
            { "compareType": "equal", "fieldName": "tierIdentifier", "fieldValue": "T1" },
            { "compareType": "equal", "fieldName": "weekStartDate", "fieldValue": f"{week_start_date}" },
        ]
    }

    # Send the POST request
    response = requests.post(url, headers=headers, json=payload)

    # Check the response
    if response.status_code == 200:
        data = response.json()
        print("Filtered data retrieved successfully:")
        return data, response.headers
    else:
        print(f"Error {response.status_code}: {response.text}")
        print(response.headers)
```

Time definition

Here we create a list of all mondays in the time horizion we want. Note data availability from FINRA.

In [ ]:
```
# Define the time horizon

from datetime import date, timedelta

# Define start and end dates
# This needs two rund
start_date = date(2023, 11, 13)  # Start, ish Q3 2019 # start date is 2019-11-25
end_date = date(2024, 11, 18)   # End of Q3 2024 # Probably max date: 2023-11-13 # 2024, 9, 30 # 2020, 3, 16

# List to store all Mondays
mondays = []

# Adjust start_date to the first Monday
current_date = start_date + timedelta(days=(7 - start_date.weekday())) if start_date.weekday() != 0 else start_date

# Iterate through dates, adding Mondays to the list
while current_date <= end_date:
    mondays.append(current_date.strftime('%Y-%m-%d'))
    current_date += timedelta(days=7)
```

In [333…
```
# For info only
mondays
```

Out[333…

```
['2023-11-13',
 '2023-11-20',
 '2023-11-27',
 '2023-12-04',
 '2023-12-11',
 '2023-12-18',
 '2023-12-25',
 '2024-01-01',
 '2024-01-08',
 '2024-01-15',
 '2024-01-22',
 '2024-01-29',
 '2024-02-05',
 '2024-02-12',
 '2024-02-19',
 '2024-02-26',
 '2024-03-04',
 '2024-03-11',
 '2024-03-18',
 '2024-03-25',
 '2024-04-01',
 '2024-04-08',
 '2024-04-15',
 '2024-04-22',
 '2024-04-29',
 '2024-05-06',
 '2024-05-13',
 '2024-05-20',
 '2024-05-27',
 '2024-06-03',
 '2024-06-10',
 '2024-06-17',
 '2024-06-24',
 '2024-07-01',
 '2024-07-08',
 '2024-07-15',
 '2024-07-22',
 '2024-07-29',
 '2024-08-05',
 '2024-08-12',
 '2024-08-19',
 '2024-08-26',
 '2024-09-02',
 '2024-09-09',
 '2024-09-16',
 '2024-09-23',
 '2024-09-30',
 '2024-10-07',
 '2024-10-14',
 '2024-10-21',
 '2024-10-28',
 '2024-11-04',
 '2024-11-11',
 '2024-11-18']
```

In [ ]:

```python
import pandas as pd

# Define the dataframe
data_df = pd.DataFrame()

trips = 15
weeks = len(mondays)
u = 0
record = 5000

# Loop through weeks
for week in mondays[:weeks]:
    print(f"Fetching data for week starting {week}")
    print(f"Total progress: {mondays.index(week)}/{len(mondays)}")

    # Loop trhough offsets
    for offset in range(0, (record + 1)*trips, (record + 1)):
        u += 1

        # Get data
        try:
            data, response_headers = get_hist_data(offset, 5000, week)
            #print(response_headers)
        except:
```

```
            remaining = int(response_headers['record-total']) % 5001
            try:
                data, response_headers = get_hist_data(offset - remaining, remaining, week)
                # Use a list to store data first
                rows = []

                # Loop through the data and append to the list
                for value in data:
                    rows.append({
                        "Date": value["weekStartDate"],
                        "Ticker": value["issueSymbolIdentifier"],
                        "DarkVol": value["totalWeeklyShareQuantity"],
                        "Market" : value["marketParticipantName"]
                    })

                # Create the DataFrame from the list of dictionaries
                new_data = pd.DataFrame(rows)
                data_df = pd.concat([data_df, new_data], ignore_index=True)
                break
            except:
                break

        # Use a list to store data first
        rows = []

        # Loop through the data and append to the list
        for value in data:
            rows.append({
                "Date": value["weekStartDate"],
                "Ticker": value["issueSymbolIdentifier"],
                "DarkVol": value["totalWeeklyShareQuantity"],
                "Market" : value["marketParticipantName"]
            })

        # Create the DataFrame from the list of dictionaries
        new_data = pd.DataFrame(rows)
        data_df = pd.concat([data_df, new_data], ignore_index=True)

        print(f"Progress: {round((u/(weeks*trips))*100, 2)}%")
```

Code cell output removed for space reasons

Now the data should be processed

In [335…
```
# String the data in case wee need to revert back.
data = data_df
```

In [336…
```
# Saving data to .csv file
data.to_csv('finra_darkpool_volume2.csv', index=False)
```

In [337…
```
# Combine two .csv files
data1 = pd.read_csv('finra_darkpool_volume.csv')
data2 = pd.read_csv('finra_darkpool_volume2.csv')

# Comnine two dataframes
data = pd.concat([data1, data2], ignore_index=True)

# Saving data to .csv file
data.to_csv('finra_darkpool_volume_all.csv', index=False)
```

In [350…
```
# Taking backup of data
data_refined = pd.read_csv('finra_darkpool_volume_all.csv')
len(data_refined)
```

Out[350… 12567767

In [356…
```
data_refined = data_refined.drop_duplicates()

data_refined.reset_index(drop=True, inplace=True)

data_refined.to_csv('finra_darkpool_volume_all_dropped.csv', index=False)
```