

Lee_Recchia_Uqaily_Yan

January 28, 2025

1 Gamma Scalping Trading Strategy

1.1 FINM 33150 - Quant Trading Strategies (Winter 2024)

1.1.1 Team: Nicholas Lee (ID: 12408152), Joey Recchia (ID: 12151786), Raafay Uqaily (ID: 12225382), Jesse Yan (ID: 12406982)

1.2 1. Introduction

The Black-Scholes (B-S) implied volatility is a forecast of a likely movement in a security's price which indicates expected volatility in the future. The model-free implied volatility from [Jiang et al.] computes the implied volatility directly from the prices of options across different strikes and maturities, without depending on assumptions of any specific options pricing model. The model-free implied volatility subsumes all information contained in the B-S implied volatility and the past realized volatility, and is a more efficient forecast of future realized volatility. We aim to develop a gamma scalping strategy based on model-free implied volatility versus B-S implied volatility.

A straddle combines a call and a put of the same strike and expiration. A long position would mean buying both, and a short position would mean shorting both. Delta measures the rate of change in an option's price relative to changes in the price of the underlying asset. The delta of an at-the-money straddle would come out to near zero, since the negative delta of the put would cancel out the positive delta of the call, as they have the similar magnitude from being at-the-money. Gamma measures the rate of change in an option's delta for a one-point change in the price of the underlying asset. Gamma is highest at-the-money and close to expiry. Gamma scalping involves profiting from short-term price movements in an underlying asset by adjusting positions based on changes in delta and gamma, mainly through delta-hedging. Advantages include: flexibility, limited risk exposure, and high potential for returns. See the below excerpt from *Trading Volatility* by Colin Bennet:

Gamma scalping (delta re-hedging) locks in profit as underlying moves

We shall assume an investor has purchased a zero delta straddle (or strangle), but the argument will hold for long call or put positions as well. If equity markets fall (from position 1 to position 2 in the chart) the position will become profitable and the delta will decrease from zero to a negative value. In order to lock in the profit, the investor must buy stock (or futures) for the portfolio to return to zero delta. Now that the portfolio is equity market neutral, it will profit from a movement up or down in the equity market. If equity markets then rise, the initial profit will be kept and a further profit earned (movement from position 2 to position 3). At position 3 the stock (or futures) position is sold and a short position initiated to return the position to zero delta.

Figure 27. Locking in Gains through Delta Hedging



The SPDR S&P 500 ETF Trust (NYSEARCA: SPY) was selected due to high liquidity in both the underlying and options markets. Options data was extracted from Jan 2018 to Feb 2024 (five-year period) to balance effectively capturing the most recent trading environment and using sufficient data. **Jiang, George J., and Yisong S. Tian.** “The model-free implied volatility and its information content.” *Review of Financial Studies*, vol. 18, no. 4, 2005, pp. 1305–1342, <https://doi.org/10.1093/rfs/hhi027>. The paper by Jiang and Tian discusses the concept of a model-free implied volatility (MFIV), a measure of the market’s expectation of future volatility that does not depend on any specific option pricing model, such as the Black-Scholes model. This approach to calculating implied volatility is significant because it potentially offers a more accurate and robust indicator of future market volatility, free from the biases and limitations inherent in model-based approaches. The equation from the paper can be seen below:

$$\sum_{i=1}^m [g(T, K_i) + g(T, K_{i-1})] \Delta K$$

Where m is the number of tradable strike prices, $\Delta K = \frac{K_{\max} - K_{\min}}{m}$, $K_i = K_{\min} + i\Delta K$, and $g(T, K_i) = \frac{C^F(T, K_i) - \max(0, F_0 - K_i)}{K_i^2}$.

The research paper delves into the methodology for calculating MFIV, which involves aggregating the implied volatilities derived from a wide range of options across different strike prices and maturities. By doing so, the MFIV captures a more comprehensive picture of the market's expectations for volatility, encompassing information from both deep out-of-the-money and in-the-money options. The paper also explores the informational content of MFIV, examining how it reflects market participants' collective views and expectations about future volatility. This includes an analysis of how MFIV predicts future market movements, its response to macroeconomic announcements or significant market events, and its relationship with historical volatility and other financial indicators. The paper also uses market data to compare the predictive power and informational efficiency of MFIV against other volatility measures. Finally, Jiang and Tian discuss the implications of their findings for both practitioners and theorists. For practitioners, the use of MFIV could lead to better hedging strategies, more accurate pricing of derivatives, and improved portfolio management. For theorists, the findings could challenge existing models of market behavior and contribute to the development of new theories that more accurately reflect the complexities of financial markets.

Oliveira, Rogerio, and Gustavo Wasserstein. “The Quest for Alpha in Equity Gamma.” https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4474815.

Hirsa, Ali, and Dilip B. Madan. “Pricing American Options Under Variance Gamma.” <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=16061c8cc6ee13947162a35ceb>

Bennett, C. (2014). Trading Volatility: Trading Volatility, Correlation, Term Structure and Skew. CreateSpace Independent Publishing Platform. ISBN: 978-146110875. <https://www.trading-volatility.com/Trading-Volatility.pdf>.

Konikov, Mikhail, and Dilip B. Madan. “Option Pricing Using Variance Gamma Markov Chains.” <https://link.springer.com/article/10.1023/A:1013816400834>.

Sepp, Artur. “When You Hedge Discretely: Optimization of Sharpe Ratio for Delta-Hedging Strategy under Discrete Hedging and Transaction Costs.https://papers.ssrn.com/sol3/papers.cfm?abstract_id=186599815.

1.3 2. Methodology Overview

We enter a long or short gamma position based on implied volatility [Oliveira et al]. The B-S implied volatility is compared with the model-free implied volatility from [Jiang et al] to provide a decision boundary on which to enter the position. If the model-free implied volatility value is higher than the Black-Scholes implied volatility by a certain threshold, we go long gamma, and if the model-free implied volatility value is lower than the Black-Scholes implied volatility value by the same threshold, we will go short gamma. We analyze daily whether to enter a long or short position on a delta-hedged straddle based on this decision boundary. A straddle reduces the number of shares necessary to initially delta hedge, as the delta of the put and the delta of the call will cancel each other out. By entering daily, it is possible to net the number of shares required for rebalancing to a near zero-delta state, thus saving on transaction costs. Each straddle position that we hold is evaluated independent of other positions for rebalancing each day. If its delta value is above a certain threshold, we rebalance by buying or selling delta shares of the underlying with the aim of getting to net zero delta. We enter each straddle with an expiry of approximately 1 month and exit each straddle after an approximately 3-week holding period (1 week before expiration), as most gamma should have been realized by this point. This minimizes the blow-up risk associated with being short gamma, since we no longer need to delta-hedge potentially volatile moves in

the week leading up to expiration (gamma is highest close to expiration). Also, by exiting our options positions with 1 week until expiration, we reduce our exposure to market microstructure idiosyncrasies. We will simulate our strategy using a number of different thresholds for when to enter positions and when to rebalance positions. We also experiment with the effects of assuming zero trading costs versus fixed trading costs at 0.01% of the notional value traded on each trade we make. Additionally, we plan on examining the impact of assuming fractional positions can be traded versus if they cannot be traded. We assume the midpoint between best offer and best ask as the price we will get in our simulation. To evaluate the performance of our strategy, we will look at metrics such as sharpe ratio, market beta, and downside beta. The only equity that this strategy's PnL is dependent on is the SPY ETF so we anticipate the strategy may have somewhat strong correlations with the market.

Oliveira, Rogerio, and Gustavo Wasserstein. “The Quest for Alpha in Equity Gamma.”
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4474815.

1.4 Code init

Imports Import all dependencies and functions via source file.

```
[1]: from source import *
```

Quandl/WRDS Credentials Importing quandl API key. Individual keys are protected in a .json file.

```
[2]: with open('quandl_config.json', 'r') as f:
    config = json.load(f)
    quandl_api_key = config['api_key']

#quandl.ApiConfig.api_key = quandl_api_key
```

```
[3]: with open('wrds_config.json', 'r') as config_file:
    wrds_credentials = json.load(config_file)
```

Local use

The API keys and WRDS login are loaded from *quandl_config.json* in the same directory as this notebook locally, which has the following content,

```
{
    "api_key": ""}
```

and *wrds_config.json*,

```
{
    "wrds": {
        "username": "",
        "password": ""
    }
}
```

To skip WRDS input on pulls, we have .pgpass with,

```
hostname:port:database:username:password
wrds-pgdata.wharton.upenn.edu:9737:wrds:(user):(pass)
```

1.5 3. Data Retrieval

For our data load process, we worked through github to upload functions and called them locally to each grab data, so we have several checkpoints where we save data to .csv format in the local directory so they can be reloaded directly when continuing work.

1.5.1 SPY price data via Quandl

```
[4]: datapull = fetch_quandl_table('QUOTEMEDIA/PRICES', quandl_api_key,
    ↪ticker='SPY', avoid_download=True)
```

Skipping any possible download of QUOTEMEDIA/PRICES

Optional saving checkpoint:

```
[5]: datapull.to_csv('spy_tickerdata.csv', index=False)
```

```
[6]: display(datapull)
```

	ticker	date	open	high	low	close	\
0	SPY	1993-08-13	45.093700	45.156200	45.093700	45.125000	
1	SPY	1993-08-16	45.156200	45.500000	45.156200	45.375000	
2	SPY	1993-08-17	45.343700	45.531200	45.343700	45.531200	
3	SPY	1993-08-18	45.687500	45.875000	45.656200	45.781200	
4	SPY	1993-08-19	45.812500	45.812500	45.718700	45.781200	
...	\
7827	SPY	1993-06-22	44.656200	44.656200	44.562500	44.625000	
7828	SPY	1994-11-16	46.765598	46.843700	46.609299	46.843700	
7829	SPY	2010-03-10	114.510002	115.279999	114.410004	114.970001	
7830	SPY	2022-02-15	443.730000	446.280000	443.180000	446.100000	
7831	SPY	1993-02-09	44.812500	44.812500	44.562500	44.656200	
	volume	dividend	split	adj_open	adj_high	adj_low	\
0	103500.0	0.0	1.0	25.778428	25.814157	25.778428	
1	241800.0	0.0	1.0	25.814157	26.010695	25.814157	
2	369300.0	0.0	1.0	25.921344	26.028531	25.921344	
3	414300.0	0.0	1.0	26.117882	26.225069	26.099989	
4	28500.0	0.0	1.0	26.189340	26.189340	26.135718	
...	\
7827	137500.0	0.0	1.0	25.528325	25.528325	25.474761	
7828	106900.0	0.0	1.0	27.589587	27.635663	27.497377	
7829	186088800.0	0.0	1.0	87.964535	88.556033	87.887719	
7830	88435136.0	0.0	1.0	430.043935	432.515285	429.510899	
7831	122100.0	0.0	1.0	25.316163	25.316163	25.174929	

```

        adj_close    adj_volume
0      25.796321   103500.0
1      25.939237   241800.0
2      26.028531   369300.0
3      26.171447   414300.0
4      26.171447   28500.0
...
7827   25.510490   137500.0
7828   27.635663   106900.0
7829   88.317898  186088800.0
7830   432.340837  88435136.0
7831   25.227864   122100.0

```

[7832 rows x 14 columns]

1.5.2 WRDS Options Data for SPY

Connection Toggle Not meant to run in sequence; rather, toggles an open connection to WRDS to pull data.

```
[7]: db = wrds.Connection(
    wrds_username=wrds_credentials['wrds']['username'],
)
```

Loading library list...

Done

```
[11]: db.close()
```

Data Pull Pulls the data and stores in .csv file in the same directory, to allow reloading from local. This could be done more elegantly like the provided Quandl function, but since the data doesn't change day-by-day, it only needs to be pulled once.

```
[8]: start_year = 2018
end_year = 2023
```

```
[9]: optiondata = pd.DataFrame()

for year in range(start_year, end_year + 1):
    sql_query = f"""
        SELECT date, exdate, last_date, cp_flag, strike_price, best_bid, best_offer,
               volume, open_interest, impl_volatility, delta, gamma, vega, theta,
               expiry_indicator
        FROM optionm_all.opprcd{year}
        WHERE secid = '109820'
        AND date BETWEEN '{year}-01-01' AND '2023-02-28'
        AND exdate - date < 800
    """
```

```

year_data = db.raw_sql(sql_query)
optiondata = pd.concat([optiondata, year_data], ignore_index=True)

```

We could add “AND volume >= 100” to filter volume, if we choose. We ended up keeping all volumes, otherwise there were not enough strikes with proper bid/ask and Greeks data.

```
[10]: csv_file_path = 'option_data.csv'
optiondata.to_csv(csv_file_path, index=False)
print(f"All data saved to {csv_file_path}")
```

All data saved to option_data.csv

1.6 4. Initial Data Restructuring

1.6.1 Checkpoint 1: Retrieve as Variable (csv save 1)

Retrieving the first checkpoint:

```
[2]: csv_file_path = 'option_data.csv'
spydata = pd.read_csv(csv_file_path)

[3]: display(spydata)
```

	date	exdate	last_date	cp_flag	strike_price	best_bid	\
0	2018-01-02	2018-01-03	2017-12-28	C	235000.0	33.59	
1	2018-01-02	2018-01-03	2018-01-02	C	240000.0	28.59	
2	2018-01-02	2018-01-03	2017-12-27	C	242500.0	26.09	
3	2018-01-02	2018-01-03	2018-01-02	C	245000.0	23.59	
4	2018-01-02	2018-01-03		C	247500.0	21.08	
...	
10448189	2023-02-28	2025-03-21	2022-12-28	P	600000.0	200.50	
10448190	2023-02-28	2025-03-21		P	605000.0	205.50	
10448191	2023-02-28	2025-03-21		P	610000.0	210.00	
10448192	2023-02-28	2025-03-21		P	615000.0	215.00	
10448193	2023-02-28	2025-03-21		P	620000.0	220.00	
	best_offer	volume	open_interest	impl_volatility	delta	gamma	\
0	33.81	0.0	187.0		NaN	NaN	NaN
1	28.76	1.0	88.0		NaN	NaN	NaN
2	26.32	0.0	2.0		NaN	NaN	NaN
3	23.81	12.0	58.0		NaN	NaN	NaN
4	21.32	0.0	0.0		NaN	NaN	NaN
...	
10448189	205.50	0.0	0.0		NaN	NaN	NaN
10448190	210.50	0.0	0.0		NaN	NaN	NaN
10448191	215.00	0.0	0.0		NaN	NaN	NaN
10448192	220.00	0.0	0.0		NaN	NaN	NaN
10448193	225.00	0.0	0.0		NaN	NaN	NaN
	vega	theta	expiry_indicator				

```

0      NaN    NaN      w
1      NaN    NaN      w
2      NaN    NaN      w
3      NaN    NaN      w
4      NaN    NaN      w
...
10448189   NaN    NaN      NaN
10448190   NaN    NaN      NaN
10448191   NaN    NaN      NaN
10448192   NaN    NaN      NaN
10448193   NaN    NaN      NaN

```

[10448194 rows x 15 columns]

Order by the appropriate date, then expiry date, then strike price within the date spread.

```
[4]: spydata_sorted = spydata.sort_values(by=['date', 'exdate', 'cp_flag', 'strike_price'])
spydata_final = spydata_sorted.reset_index(drop=True)
```

```
[5]: display(spydata_final)
```

	date	exdate	last_date	cp_flag	strike_price	best_bid	\
0	2018-01-02	2018-01-03	2017-12-28	C	235000.0	33.59	
1	2018-01-02	2018-01-03	2018-01-02	C	240000.0	28.59	
2	2018-01-02	2018-01-03	2017-12-27	C	242500.0	26.09	
3	2018-01-02	2018-01-03	2018-01-02	C	245000.0	23.59	
4	2018-01-02	2018-01-03		C	247500.0	21.08	
...	
10448189	2023-02-28	2025-03-21	2022-12-28	P	600000.0	200.50	
10448190	2023-02-28	2025-03-21		P	605000.0	205.50	
10448191	2023-02-28	2025-03-21		P	610000.0	210.00	
10448192	2023-02-28	2025-03-21		P	615000.0	215.00	
10448193	2023-02-28	2025-03-21		P	620000.0	220.00	
	best_offer	volume	open_interest	impl_volatility	delta	gamma	\
0	33.81	0.0	187.0		NaN	NaN	NaN
1	28.76	1.0	88.0		NaN	NaN	NaN
2	26.32	0.0	2.0		NaN	NaN	NaN
3	23.81	12.0	58.0		NaN	NaN	NaN
4	21.32	0.0	0.0		NaN	NaN	NaN
...	
10448189	205.50	0.0	0.0		NaN	NaN	NaN
10448190	210.50	0.0	0.0		NaN	NaN	NaN
10448191	215.00	0.0	0.0		NaN	NaN	NaN
10448192	220.00	0.0	0.0		NaN	NaN	NaN
10448193	225.00	0.0	0.0		NaN	NaN	NaN
	vega	theta	expiry_indicator				

```

0      NaN    NaN      w
1      NaN    NaN      w
2      NaN    NaN      w
3      NaN    NaN      w
4      NaN    NaN      w
...
10448189  NaN    NaN      NaN
10448190  NaN    NaN      NaN
10448191  NaN    NaN      NaN
10448192  NaN    NaN      NaN
10448193  NaN    NaN      NaN

```

[10448194 rows x 15 columns]

```
[6]: spydata_final.to_csv('sorteddata.csv', index=False)
```

1.6.2 Checkpoint 2: Retrieve sorted data; make adjustments (csv save 2)

```
[2]: spy_optiondata = pd.read_csv('sorteddata.csv')
```

Load the ticker price data, directly from local.

```
[4]: spy_tickerdata = pd.read_csv('spy_tickerdata.csv')
```

Make some adjustments and attach to our data table for options. * Set a start and end date. We chose this initially to be 2018 to the end of the data; in particular, this is what is considered to be the modern trading regime for SPY and its options. * Regularize the price

```
[6]: start_date = '2018-01-01'
end_date = '2023-02-28'
```

```
[7]: spy_tickerdata_reduced = prepare_ticker_data(spy_tickerdata, '2018-01-01', ↴
                                                 '2023-02-28')
spy_optiondata_prepared = prepare_option_data(spy_optiondata)
spy_optiondata_enriched = enrich_option_data(spy_optiondata_prepared, ↴
                                              spy_tickerdata_reduced)
display(spy_optiondata_enriched)
```

	date	exdate	last_date	cp_flag	strike_price	best_bid	\
0	2018-01-02	2018-01-03	2017-12-28	C	235.0	33.59	
1	2018-01-02	2018-01-03	2018-01-02	C	240.0	28.59	
2	2018-01-02	2018-01-03	2017-12-27	C	242.5	26.09	
3	2018-01-02	2018-01-03	2018-01-02	C	245.0	23.59	
4	2018-01-02	2018-01-03		C	247.5	21.08	
...	
10448189	2023-02-28	2025-03-21	2022-12-28	P	600.0	200.50	
10448190	2023-02-28	2025-03-21		NaN	605.0	205.50	
10448191	2023-02-28	2025-03-21		NaN	610.0	210.00	
10448192	2023-02-28	2025-03-21		NaN	615.0	215.00	

10448193	2023-02-28	2025-03-21	NaN	P	620.0	220.00	
0	best_offer	volume	open_interest	impl_volatility	delta	gamma	\
1	33.81	0.0	187.0	NaN	NaN	NaN	
2	28.76	1.0	88.0	NaN	NaN	NaN	
3	26.32	0.0	2.0	NaN	NaN	NaN	
4	23.81	12.0	58.0	NaN	NaN	NaN	
...	
10448189	205.50	0.0	0.0	NaN	NaN	NaN	
10448190	210.50	0.0	0.0	NaN	NaN	NaN	
10448191	215.00	0.0	0.0	NaN	NaN	NaN	
10448192	220.00	0.0	0.0	NaN	NaN	NaN	
10448193	225.00	0.0	0.0	NaN	NaN	NaN	
0	vega	theta	expiry_indicator	close	adj_open	adj_close	\
1	NaN	NaN	w	268.77	242.053393	242.893856	
2	NaN	NaN	w	268.77	242.053393	242.893856	
3	NaN	NaN	w	268.77	242.053393	242.893856	
4	NaN	NaN	w	268.77	242.053393	242.893856	
...	
10448189	NaN	NaN	NaN	396.26	391.240559	390.285185	
10448190	NaN	NaN	NaN	396.26	391.240559	390.285185	
10448191	NaN	NaN	NaN	396.26	391.240559	390.285185	
10448192	NaN	NaN	NaN	396.26	391.240559	390.285185	
10448193	NaN	NaN	NaN	396.26	391.240559	390.285185	
0	adj_volume						
1	86655749.0						
2	86655749.0						
3	86655749.0						
4	86655749.0						
...	...						
10448189	96141367.0						
10448190	96141367.0						
10448191	96141367.0						
10448192	96141367.0						
10448193	96141367.0						

[10448194 rows x 19 columns]

Two more saving checkpoints - this saves the data, one with all datapoints, and the other removing those where Greeks and IV are missing from the data. For this, we will use the further cleaned file to see where exactly the missing data is.

[8]: spy_optiondata_enriched.to_csv('combinedata.csv', index=False)

```
[9]: spydata_cleaned = spy_optiondata_enriched.dropna(subset=['impl_volatility',  
    'delta', 'gamma', 'vega', 'theta'])  
spydata_dropna = spydata_cleaned.reset_index(drop=True)  
spydata_dropna.to_csv('combinedata_dropna.csv', index=False)
```

1.6.3 Checkpoint 3: Retrieve combined data; determining missing data, combined strikes

This is another load checkpoint.

```
[2]: filepath = 'combinedata.csv'  
filepath1 = 'combinedata_dropna.csv'  
options_df = pd.read_csv(filepath)  
options_df1 = pd.read_csv(filepath1)
```

Now we'll take a look to see if for our strategy, which we will initially set as approximately month-long options to be closed 1 week to expiry, have data in the dataset for the entire time that we hold them.

Looking at the combined dataset, at a glance:

```
[3]: exdates = set(options_df['exdate'])  
quotedates = set(options_df['date'])
```

```
[4]: exdates - quotedates
```

```
[4]: {'2018-12-05',  
      '2023-03-01',  
      '2023-03-02',  
      '2023-03-03',  
      '2023-03-06',  
      '2023-03-07',  
      '2023-03-08',  
      '2023-03-09',  
      '2023-03-10',  
      '2023-03-13',  
      '2023-03-14',  
      '2023-03-17',  
      '2023-03-24',  
      '2023-03-31',  
      '2023-04-06',  
      '2023-04-21',  
      '2023-05-19',  
      '2023-06-16',  
      '2023-06-30',  
      '2023-07-21',  
      '2023-08-18',  
      '2023-09-15',  
      '2023-09-29',
```

```
'2023-12-15',
'2023-12-29',
'2024-01-19',
'2024-03-15',
'2024-06-21',
'2024-12-20',
'2025-01-17',
'2025-03-21'}
```

```
[5]: quotedates - exdates
```

```
[5]: {'2018-01-02',
'2018-01-04',
'2018-01-08',
'2018-01-09',
'2018-01-11',
'2018-01-16',
'2018-01-18',
'2018-01-22',
'2018-01-23',
'2018-01-25',
'2018-01-29',
'2018-01-30',
'2018-02-01',
'2018-02-05',
'2018-02-06',
'2018-02-08',
'2018-02-12',
'2018-02-13',
'2018-02-15',
'2018-02-20',
'2018-02-22',
'2018-02-27',
'2018-03-01',
'2018-03-06',
'2018-03-08',
'2018-03-13',
'2018-03-15',
'2018-03-20',
'2018-03-22',
'2018-03-27',
'2018-04-03',
'2018-04-05',
'2018-04-10',
'2018-04-12',
'2018-04-17',
'2018-04-19',
```

'2018-04-24',
'2018-04-26',
'2018-05-01',
'2018-05-03',
'2018-05-08',
'2018-05-10',
'2018-05-15',
'2018-05-17',
'2018-05-22',
'2018-05-24',
'2018-05-31',
'2018-06-05',
'2018-06-07',
'2018-06-12',
'2018-06-14',
'2018-06-19',
'2018-06-21',
'2018-06-26',
'2018-06-28',
'2018-07-05',
'2018-07-10',
'2018-07-12',
'2018-07-17',
'2018-07-19',
'2018-07-24',
'2018-07-26',
'2018-07-31',
'2018-08-02',
'2018-08-07',
'2018-08-09',
'2018-08-14',
'2018-08-16',
'2018-08-21',
'2018-08-23',
'2018-08-28',
'2018-08-30',
'2018-09-06',
'2018-09-11',
'2018-09-13',
'2018-09-18',
'2018-09-20',
'2018-09-25',
'2018-09-27',
'2018-10-02',
'2018-10-04',
'2018-10-09',
'2018-10-11',

'2018-10-16',
'2018-10-18',
'2018-10-23',
'2018-10-25',
'2018-10-30',
'2018-11-01',
'2018-11-06',
'2018-11-08',
'2018-11-13',
'2018-11-15',
'2018-11-20',
'2018-11-27',
'2018-11-29',
'2018-12-04',
'2018-12-06',
'2018-12-11',
'2018-12-13',
'2018-12-18',
'2018-12-20',
'2018-12-27',
'2019-01-03',
'2019-01-08',
'2019-01-10',
'2019-01-15',
'2019-01-17',
'2019-01-24',
'2019-01-29',
'2019-01-31',
'2019-02-05',
'2019-02-07',
'2019-02-12',
'2019-02-14',
'2019-02-21',
'2019-02-26',
'2019-02-28',
'2019-03-05',
'2019-03-07',
'2019-03-12',
'2019-03-14',
'2019-03-19',
'2019-03-21',
'2019-03-26',
'2019-03-28',
'2019-04-02',
'2019-04-04',
'2019-04-09',
'2019-04-11',

'2019-04-16',
'2019-04-23',
'2019-04-25',
'2019-04-30',
'2019-05-02',
'2019-05-07',
'2019-05-09',
'2019-05-14',
'2019-05-16',
'2019-05-21',
'2019-05-23',
'2019-05-30',
'2019-06-04',
'2019-06-06',
'2019-06-11',
'2019-06-13',
'2019-06-18',
'2019-06-20',
'2019-06-25',
'2019-06-27',
'2019-07-02',
'2019-07-09',
'2019-07-11',
'2019-07-16',
'2019-07-18',
'2019-07-23',
'2019-07-25',
'2019-07-30',
'2019-08-01',
'2019-08-06',
'2019-08-08',
'2019-08-13',
'2019-08-15',
'2019-08-20',
'2019-08-22',
'2019-08-27',
'2019-08-29',
'2019-09-05',
'2019-09-10',
'2019-09-12',
'2019-09-17',
'2019-09-19',
'2019-09-24',
'2019-09-26',
'2019-10-01',
'2019-10-03',
'2019-10-08',

'2019-10-10',
'2019-10-15',
'2019-10-17',
'2019-10-22',
'2019-10-24',
'2019-10-29',
'2019-10-31',
'2019-11-05',
'2019-11-07',
'2019-11-12',
'2019-11-14',
'2019-11-19',
'2019-11-21',
'2019-11-26',
'2019-12-03',
'2019-12-05',
'2019-12-10',
'2019-12-12',
'2019-12-17',
'2019-12-19',
'2019-12-26',
'2020-01-02',
'2020-01-07',
'2020-01-09',
'2020-01-14',
'2020-01-16',
'2020-01-23',
'2020-01-28',
'2020-01-30',
'2020-02-04',
'2020-02-06',
'2020-02-11',
'2020-02-13',
'2020-02-20',
'2020-02-25',
'2020-02-27',
'2020-03-03',
'2020-03-05',
'2020-03-10',
'2020-03-12',
'2020-03-17',
'2020-03-19',
'2020-03-24',
'2020-03-26',
'2020-04-02',
'2020-04-07',
'2020-04-14',

'2020-04-16',
'2020-04-21',
'2020-04-23',
'2020-04-28',
'2020-04-30',
'2020-05-05',
'2020-05-07',
'2020-05-12',
'2020-05-14',
'2020-05-19',
'2020-05-21',
'2020-05-28',
'2020-06-02',
'2020-06-04',
'2020-06-09',
'2020-06-11',
'2020-06-16',
'2020-06-18',
'2020-06-23',
'2020-06-25',
'2020-07-07',
'2020-07-09',
'2020-07-14',
'2020-07-16',
'2020-07-21',
'2020-07-23',
'2020-07-28',
'2020-07-30',
'2020-08-04',
'2020-08-06',
'2020-08-11',
'2020-08-13',
'2020-08-18',
'2020-08-20',
'2020-08-25',
'2020-08-27',
'2020-09-01',
'2020-09-03',
'2020-09-10',
'2020-09-15',
'2020-09-17',
'2020-09-22',
'2020-09-24',
'2020-09-29',
'2020-10-01',
'2020-10-06',
'2020-10-08',

'2020-10-13',
'2020-10-15',
'2020-10-20',
'2020-10-22',
'2020-10-27',
'2020-10-29',
'2020-11-03',
'2020-11-05',
'2020-11-10',
'2020-11-12',
'2020-11-17',
'2020-11-19',
'2020-11-24',
'2020-12-01',
'2020-12-03',
'2020-12-08',
'2020-12-10',
'2020-12-15',
'2020-12-17',
'2020-12-22',
'2020-12-29',
'2021-01-05',
'2021-01-07',
'2021-01-12',
'2021-01-14',
'2021-01-21',
'2021-01-26',
'2021-01-28',
'2021-02-02',
'2021-02-04',
'2021-02-09',
'2021-02-11',
'2021-02-18',
'2021-02-23',
'2021-02-25',
'2021-03-02',
'2021-03-04',
'2021-03-09',
'2021-03-11',
'2021-03-16',
'2021-03-18',
'2021-03-23',
'2021-03-25',
'2021-03-30',
'2021-04-06',
'2021-04-08',
'2021-04-13',

'2021-04-15',
'2021-04-20',
'2021-04-22',
'2021-04-27',
'2021-04-29',
'2021-05-04',
'2021-05-06',
'2021-05-11',
'2021-05-13',
'2021-05-18',
'2021-05-20',
'2021-05-25',
'2021-05-27',
'2021-06-03',
'2021-06-08',
'2021-06-10',
'2021-06-15',
'2021-06-17',
'2021-06-22',
'2021-06-24',
'2021-06-29',
'2021-07-01',
'2021-07-08',
'2021-07-13',
'2021-07-15',
'2021-07-20',
'2021-07-22',
'2021-07-27',
'2021-07-29',
'2021-08-03',
'2021-08-05',
'2021-08-10',
'2021-08-12',
'2021-08-17',
'2021-08-19',
'2021-08-24',
'2021-08-26',
'2021-08-31',
'2021-09-02',
'2021-09-09',
'2021-09-14',
'2021-09-16',
'2021-09-21',
'2021-09-23',
'2021-09-28',
'2021-10-05',
'2021-10-07',

'2021-10-12',
'2021-10-14',
'2021-10-19',
'2021-10-21',
'2021-10-26',
'2021-10-28',
'2021-11-02',
'2021-11-04',
'2021-11-09',
'2021-11-11',
'2021-11-16',
'2021-11-18',
'2021-11-23',
'2021-11-30',
'2021-12-02',
'2021-12-07',
'2021-12-09',
'2021-12-14',
'2021-12-16',
'2021-12-21',
'2021-12-28',
'2021-12-30',
'2022-01-04',
'2022-01-06',
'2022-01-11',
'2022-01-13',
'2022-01-20',
'2022-01-25',
'2022-01-27',
'2022-02-01',
'2022-02-03',
'2022-02-08',
'2022-02-10',
'2022-02-15',
'2022-02-17',
'2022-02-24',
'2022-03-01',
'2022-03-03',
'2022-03-08',
'2022-03-10',
'2022-03-15',
'2022-03-17',
'2022-03-22',
'2022-03-24',
'2022-03-29',
'2022-04-05',
'2022-04-07',

'2022-04-12',
'2022-04-19',
'2022-04-21',
'2022-04-26',
'2022-04-28',
'2022-05-03',
'2022-05-05',
'2022-05-10',
'2022-05-12',
'2022-05-17',
'2022-05-19',
'2022-05-24',
'2022-05-26',
'2022-06-02',
'2022-06-07',
'2022-06-09',
'2022-06-14',
'2022-06-16',
'2022-06-23',
'2022-06-28',
'2022-07-07',
'2022-07-12',
'2022-07-14',
'2022-07-19',
'2022-07-21',
'2022-07-26',
'2022-07-28',
'2022-08-02',
'2022-08-04',
'2022-08-09',
'2022-08-11',
'2022-08-16',
'2022-08-18',
'2022-08-23',
'2022-08-25',
'2022-08-30',
'2022-09-01',
'2022-09-08',
'2022-09-13',
'2022-09-15',
'2022-09-20',
'2022-09-22',
'2022-09-27',
'2022-09-29',
'2022-10-04',
'2022-10-06',
'2022-10-11',

```
'2022-10-13',
'2022-10-18',
'2022-10-20',
'2022-10-25',
'2022-10-27',
'2022-11-01',
'2022-11-03',
'2022-11-08',
'2022-11-10']
```

[6] : exdates & quotedates

```
[6] : {'2018-01-03',
'2018-01-05',
'2018-01-10',
'2018-01-12',
'2018-01-17',
'2018-01-19',
'2018-01-24',
'2018-01-26',
'2018-01-31',
'2018-02-02',
'2018-02-07',
'2018-02-09',
'2018-02-14',
'2018-02-16',
'2018-02-21',
'2018-02-23',
'2018-02-26',
'2018-02-28',
'2018-03-02',
'2018-03-05',
'2018-03-07',
'2018-03-09',
'2018-03-12',
'2018-03-14',
'2018-03-16',
'2018-03-19',
'2018-03-21',
'2018-03-23',
'2018-03-26',
'2018-03-28',
'2018-03-29',
'2018-04-02',
'2018-04-04',
'2018-04-06',
'2018-04-09',
```

'2018-04-11',
'2018-04-13',
'2018-04-16',
'2018-04-18',
'2018-04-20',
'2018-04-23',
'2018-04-25',
'2018-04-27',
'2018-04-30',
'2018-05-02',
'2018-05-04',
'2018-05-07',
'2018-05-09',
'2018-05-11',
'2018-05-14',
'2018-05-16',
'2018-05-18',
'2018-05-21',
'2018-05-23',
'2018-05-25',
'2018-05-29',
'2018-05-30',
'2018-06-01',
'2018-06-04',
'2018-06-06',
'2018-06-08',
'2018-06-11',
'2018-06-13',
'2018-06-15',
'2018-06-18',
'2018-06-20',
'2018-06-22',
'2018-06-25',
'2018-06-27',
'2018-06-29',
'2018-07-02',
'2018-07-03',
'2018-07-06',
'2018-07-09',
'2018-07-11',
'2018-07-13',
'2018-07-16',
'2018-07-18',
'2018-07-20',
'2018-07-23',
'2018-07-25',
'2018-07-27',

'2018-07-30',
'2018-08-01',
'2018-08-03',
'2018-08-06',
'2018-08-08',
'2018-08-10',
'2018-08-13',
'2018-08-15',
'2018-08-17',
'2018-08-20',
'2018-08-22',
'2018-08-24',
'2018-08-27',
'2018-08-29',
'2018-08-31',
'2018-09-04',
'2018-09-05',
'2018-09-07',
'2018-09-10',
'2018-09-12',
'2018-09-14',
'2018-09-17',
'2018-09-19',
'2018-09-21',
'2018-09-24',
'2018-09-26',
'2018-09-28',
'2018-10-01',
'2018-10-03',
'2018-10-05',
'2018-10-08',
'2018-10-10',
'2018-10-12',
'2018-10-15',
'2018-10-17',
'2018-10-19',
'2018-10-22',
'2018-10-24',
'2018-10-26',
'2018-10-29',
'2018-10-31',
'2018-11-02',
'2018-11-05',
'2018-11-07',
'2018-11-09',
'2018-11-12',
'2018-11-14',

'2018-11-16',
'2018-11-19',
'2018-11-21',
'2018-11-23',
'2018-11-26',
'2018-11-28',
'2018-11-30',
'2018-12-03',
'2018-12-07',
'2018-12-10',
'2018-12-12',
'2018-12-14',
'2018-12-17',
'2018-12-19',
'2018-12-21',
'2018-12-24',
'2018-12-26',
'2018-12-28',
'2018-12-31',
'2019-01-02',
'2019-01-04',
'2019-01-07',
'2019-01-09',
'2019-01-11',
'2019-01-14',
'2019-01-16',
'2019-01-18',
'2019-01-22',
'2019-01-23',
'2019-01-25',
'2019-01-28',
'2019-01-30',
'2019-02-01',
'2019-02-04',
'2019-02-06',
'2019-02-08',
'2019-02-11',
'2019-02-13',
'2019-02-15',
'2019-02-19',
'2019-02-20',
'2019-02-22',
'2019-02-25',
'2019-02-27',
'2019-03-01',
'2019-03-04',
'2019-03-06',

'2019-03-08',
'2019-03-11',
'2019-03-13',
'2019-03-15',
'2019-03-18',
'2019-03-20',
'2019-03-22',
'2019-03-25',
'2019-03-27',
'2019-03-29',
'2019-04-01',
'2019-04-03',
'2019-04-05',
'2019-04-08',
'2019-04-10',
'2019-04-12',
'2019-04-15',
'2019-04-17',
'2019-04-18',
'2019-04-22',
'2019-04-24',
'2019-04-26',
'2019-04-29',
'2019-05-01',
'2019-05-03',
'2019-05-06',
'2019-05-08',
'2019-05-10',
'2019-05-13',
'2019-05-15',
'2019-05-17',
'2019-05-20',
'2019-05-22',
'2019-05-24',
'2019-05-28',
'2019-05-29',
'2019-05-31',
'2019-06-03',
'2019-06-05',
'2019-06-07',
'2019-06-10',
'2019-06-12',
'2019-06-14',
'2019-06-17',
'2019-06-19',
'2019-06-21',
'2019-06-24',

'2019-06-26',
'2019-06-28',
'2019-07-01',
'2019-07-03',
'2019-07-05',
'2019-07-08',
'2019-07-10',
'2019-07-12',
'2019-07-15',
'2019-07-17',
'2019-07-19',
'2019-07-22',
'2019-07-24',
'2019-07-26',
'2019-07-29',
'2019-07-31',
'2019-08-02',
'2019-08-05',
'2019-08-07',
'2019-08-09',
'2019-08-12',
'2019-08-14',
'2019-08-16',
'2019-08-19',
'2019-08-21',
'2019-08-23',
'2019-08-26',
'2019-08-28',
'2019-08-30',
'2019-09-03',
'2019-09-04',
'2019-09-06',
'2019-09-09',
'2019-09-11',
'2019-09-13',
'2019-09-16',
'2019-09-18',
'2019-09-20',
'2019-09-23',
'2019-09-25',
'2019-09-27',
'2019-09-30',
'2019-10-02',
'2019-10-04',
'2019-10-07',
'2019-10-09',
'2019-10-11',

'2019-10-14',
'2019-10-16',
'2019-10-18',
'2019-10-21',
'2019-10-23',
'2019-10-25',
'2019-10-28',
'2019-10-30',
'2019-11-01',
'2019-11-04',
'2019-11-06',
'2019-11-08',
'2019-11-11',
'2019-11-13',
'2019-11-15',
'2019-11-18',
'2019-11-20',
'2019-11-22',
'2019-11-25',
'2019-11-27',
'2019-11-29',
'2019-12-02',
'2019-12-04',
'2019-12-06',
'2019-12-09',
'2019-12-11',
'2019-12-13',
'2019-12-16',
'2019-12-18',
'2019-12-20',
'2019-12-23',
'2019-12-24',
'2019-12-27',
'2019-12-30',
'2019-12-31',
'2020-01-03',
'2020-01-06',
'2020-01-08',
'2020-01-10',
'2020-01-13',
'2020-01-15',
'2020-01-17',
'2020-01-21',
'2020-01-22',
'2020-01-24',
'2020-01-27',
'2020-01-29',

'2020-01-31',
'2020-02-03',
'2020-02-05',
'2020-02-07',
'2020-02-10',
'2020-02-12',
'2020-02-14',
'2020-02-18',
'2020-02-19',
'2020-02-21',
'2020-02-24',
'2020-02-26',
'2020-02-28',
'2020-03-02',
'2020-03-04',
'2020-03-06',
'2020-03-09',
'2020-03-11',
'2020-03-13',
'2020-03-16',
'2020-03-18',
'2020-03-20',
'2020-03-23',
'2020-03-25',
'2020-03-27',
'2020-03-30',
'2020-03-31',
'2020-04-01',
'2020-04-03',
'2020-04-06',
'2020-04-08',
'2020-04-09',
'2020-04-13',
'2020-04-15',
'2020-04-17',
'2020-04-20',
'2020-04-22',
'2020-04-24',
'2020-04-27',
'2020-04-29',
'2020-05-01',
'2020-05-04',
'2020-05-06',
'2020-05-08',
'2020-05-11',
'2020-05-13',
'2020-05-15',

'2020-05-18',
'2020-05-20',
'2020-05-22',
'2020-05-26',
'2020-05-27',
'2020-05-29',
'2020-06-01',
'2020-06-03',
'2020-06-05',
'2020-06-08',
'2020-06-10',
'2020-06-12',
'2020-06-15',
'2020-06-17',
'2020-06-19',
'2020-06-22',
'2020-06-24',
'2020-06-26',
'2020-06-29',
'2020-06-30',
'2020-07-01',
'2020-07-02',
'2020-07-06',
'2020-07-08',
'2020-07-10',
'2020-07-13',
'2020-07-15',
'2020-07-17',
'2020-07-20',
'2020-07-22',
'2020-07-24',
'2020-07-27',
'2020-07-29',
'2020-07-31',
'2020-08-03',
'2020-08-05',
'2020-08-07',
'2020-08-10',
'2020-08-12',
'2020-08-14',
'2020-08-17',
'2020-08-19',
'2020-08-21',
'2020-08-24',
'2020-08-26',
'2020-08-28',
'2020-08-31',

'2020-09-02',
'2020-09-04',
'2020-09-08',
'2020-09-09',
'2020-09-11',
'2020-09-14',
'2020-09-16',
'2020-09-18',
'2020-09-21',
'2020-09-23',
'2020-09-25',
'2020-09-28',
'2020-09-30',
'2020-10-02',
'2020-10-05',
'2020-10-07',
'2020-10-09',
'2020-10-12',
'2020-10-14',
'2020-10-16',
'2020-10-19',
'2020-10-21',
'2020-10-23',
'2020-10-26',
'2020-10-28',
'2020-10-30',
'2020-11-02',
'2020-11-04',
'2020-11-06',
'2020-11-09',
'2020-11-11',
'2020-11-13',
'2020-11-16',
'2020-11-18',
'2020-11-20',
'2020-11-23',
'2020-11-25',
'2020-11-27',
'2020-11-30',
'2020-12-02',
'2020-12-04',
'2020-12-07',
'2020-12-09',
'2020-12-11',
'2020-12-14',
'2020-12-16',
'2020-12-18',

'2020-12-21',
'2020-12-23',
'2020-12-24',
'2020-12-28',
'2020-12-30',
'2020-12-31',
'2021-01-04',
'2021-01-06',
'2021-01-08',
'2021-01-11',
'2021-01-13',
'2021-01-15',
'2021-01-19',
'2021-01-20',
'2021-01-22',
'2021-01-25',
'2021-01-27',
'2021-01-29',
'2021-02-01',
'2021-02-03',
'2021-02-05',
'2021-02-08',
'2021-02-10',
'2021-02-12',
'2021-02-16',
'2021-02-17',
'2021-02-19',
'2021-02-22',
'2021-02-24',
'2021-02-26',
'2021-03-01',
'2021-03-03',
'2021-03-05',
'2021-03-08',
'2021-03-10',
'2021-03-12',
'2021-03-15',
'2021-03-17',
'2021-03-19',
'2021-03-22',
'2021-03-24',
'2021-03-26',
'2021-03-29',
'2021-03-31',
'2021-04-01',
'2021-04-05',
'2021-04-07',

'2021-04-09',
'2021-04-12',
'2021-04-14',
'2021-04-16',
'2021-04-19',
'2021-04-21',
'2021-04-23',
'2021-04-26',
'2021-04-28',
'2021-04-30',
'2021-05-03',
'2021-05-05',
'2021-05-07',
'2021-05-10',
'2021-05-12',
'2021-05-14',
'2021-05-17',
'2021-05-19',
'2021-05-21',
'2021-05-24',
'2021-05-26',
'2021-05-28',
'2021-06-01',
'2021-06-02',
'2021-06-04',
'2021-06-07',
'2021-06-09',
'2021-06-11',
'2021-06-14',
'2021-06-16',
'2021-06-18',
'2021-06-21',
'2021-06-23',
'2021-06-25',
'2021-06-28',
'2021-06-30',
'2021-07-02',
'2021-07-06',
'2021-07-07',
'2021-07-09',
'2021-07-12',
'2021-07-14',
'2021-07-16',
'2021-07-19',
'2021-07-21',
'2021-07-23',
'2021-07-26',

'2021-07-28',
'2021-07-30',
'2021-08-02',
'2021-08-04',
'2021-08-06',
'2021-08-09',
'2021-08-11',
'2021-08-13',
'2021-08-16',
'2021-08-18',
'2021-08-20',
'2021-08-23',
'2021-08-25',
'2021-08-27',
'2021-08-30',
'2021-09-01',
'2021-09-03',
'2021-09-07',
'2021-09-08',
'2021-09-10',
'2021-09-13',
'2021-09-15',
'2021-09-17',
'2021-09-20',
'2021-09-22',
'2021-09-24',
'2021-09-27',
'2021-09-29',
'2021-09-30',
'2021-10-01',
'2021-10-04',
'2021-10-06',
'2021-10-08',
'2021-10-11',
'2021-10-13',
'2021-10-15',
'2021-10-18',
'2021-10-20',
'2021-10-22',
'2021-10-25',
'2021-10-27',
'2021-10-29',
'2021-11-01',
'2021-11-03',
'2021-11-05',
'2021-11-08',
'2021-11-10',

'2021-11-12',
'2021-11-15',
'2021-11-17',
'2021-11-19',
'2021-11-22',
'2021-11-24',
'2021-11-26',
'2021-11-29',
'2021-12-01',
'2021-12-03',
'2021-12-06',
'2021-12-08',
'2021-12-10',
'2021-12-13',
'2021-12-15',
'2021-12-17',
'2021-12-20',
'2021-12-22',
'2021-12-23',
'2021-12-27',
'2021-12-29',
'2021-12-31',
'2022-01-03',
'2022-01-05',
'2022-01-07',
'2022-01-10',
'2022-01-12',
'2022-01-14',
'2022-01-18',
'2022-01-19',
'2022-01-21',
'2022-01-24',
'2022-01-26',
'2022-01-28',
'2022-01-31',
'2022-02-02',
'2022-02-04',
'2022-02-07',
'2022-02-09',
'2022-02-11',
'2022-02-14',
'2022-02-16',
'2022-02-18',
'2022-02-22',
'2022-02-23',
'2022-02-25',
'2022-02-28',

'2022-03-02',
'2022-03-04',
'2022-03-07',
'2022-03-09',
'2022-03-11',
'2022-03-14',
'2022-03-16',
'2022-03-18',
'2022-03-21',
'2022-03-23',
'2022-03-25',
'2022-03-28',
'2022-03-30',
'2022-03-31',
'2022-04-01',
'2022-04-04',
'2022-04-06',
'2022-04-08',
'2022-04-11',
'2022-04-13',
'2022-04-14',
'2022-04-18',
'2022-04-20',
'2022-04-22',
'2022-04-25',
'2022-04-27',
'2022-04-29',
'2022-05-02',
'2022-05-04',
'2022-05-06',
'2022-05-09',
'2022-05-11',
'2022-05-13',
'2022-05-16',
'2022-05-18',
'2022-05-20',
'2022-05-23',
'2022-05-25',
'2022-05-27',
'2022-05-31',
'2022-06-01',
'2022-06-03',
'2022-06-06',
'2022-06-08',
'2022-06-10',
'2022-06-13',
'2022-06-15',

'2022-06-17',
'2022-06-21',
'2022-06-22',
'2022-06-24',
'2022-06-27',
'2022-06-29',
'2022-06-30',
'2022-07-01',
'2022-07-05',
'2022-07-06',
'2022-07-08',
'2022-07-11',
'2022-07-13',
'2022-07-15',
'2022-07-18',
'2022-07-20',
'2022-07-22',
'2022-07-25',
'2022-07-27',
'2022-07-29',
'2022-08-01',
'2022-08-03',
'2022-08-05',
'2022-08-08',
'2022-08-10',
'2022-08-12',
'2022-08-15',
'2022-08-17',
'2022-08-19',
'2022-08-22',
'2022-08-24',
'2022-08-26',
'2022-08-29',
'2022-08-31',
'2022-09-02',
'2022-09-06',
'2022-09-07',
'2022-09-09',
'2022-09-12',
'2022-09-14',
'2022-09-16',
'2022-09-19',
'2022-09-21',
'2022-09-23',
'2022-09-26',
'2022-09-28',
'2022-09-30',

'2022-10-03',
'2022-10-05',
'2022-10-07',
'2022-10-10',
'2022-10-12',
'2022-10-14',
'2022-10-17',
'2022-10-19',
'2022-10-21',
'2022-10-24',
'2022-10-26',
'2022-10-28',
'2022-10-31',
'2022-11-02',
'2022-11-04',
'2022-11-07',
'2022-11-09',
'2022-11-11',
'2022-11-14',
'2022-11-15',
'2022-11-16',
'2022-11-17',
'2022-11-18',
'2022-11-21',
'2022-11-22',
'2022-11-23',
'2022-11-25',
'2022-11-28',
'2022-11-29',
'2022-11-30',
'2022-12-01',
'2022-12-02',
'2022-12-05',
'2022-12-06',
'2022-12-07',
'2022-12-08',
'2022-12-09',
'2022-12-12',
'2022-12-13',
'2022-12-14',
'2022-12-15',
'2022-12-16',
'2022-12-19',
'2022-12-20',
'2022-12-21',
'2022-12-22',
'2022-12-23',

```
'2022-12-27',
'2022-12-28',
'2022-12-29',
'2022-12-30',
'2023-01-03',
'2023-01-04',
'2023-01-05',
'2023-01-06',
'2023-01-09',
'2023-01-10',
'2023-01-11',
'2023-01-12',
'2023-01-13',
'2023-01-17',
'2023-01-18',
'2023-01-19',
'2023-01-20',
'2023-01-23',
'2023-01-24',
'2023-01-25',
'2023-01-26',
'2023-01-27',
'2023-01-30',
'2023-01-31',
'2023-02-01',
'2023-02-02',
'2023-02-03',
'2023-02-06',
'2023-02-07',
'2023-02-08',
'2023-02-09',
'2023-02-10',
'2023-02-13',
'2023-02-14',
'2023-02-15',
'2023-02-16',
'2023-02-17',
'2023-02-21',
'2023-02-22',
'2023-02-23',
'2023-02-24',
'2023-02-27',
'2023-02-28'}
```

Here, we calculate the time to expiry for each contract, in days.

```
[7]: options_df['TTE'] = (pd.to_datetime(options_df['exdate']) - pd.
    to_datetime(options_df['date'])).dt.days
options_df
```

	date	exdate	last_date	cp_flag	strike_price	best_bid	\
0	2018-01-02	2018-01-03	2017-12-28	C	235.0	33.59	
1	2018-01-02	2018-01-03	2018-01-02	C	240.0	28.59	
2	2018-01-02	2018-01-03	2017-12-27	C	242.5	26.09	
3	2018-01-02	2018-01-03	2018-01-02	C	245.0	23.59	
4	2018-01-02	2018-01-03		NaN	247.5	21.08	
...	
10448189	2023-02-28	2025-03-21	2022-12-28	P	600.0	200.50	
10448190	2023-02-28	2025-03-21		NaN	605.0	205.50	
10448191	2023-02-28	2025-03-21		NaN	610.0	210.00	
10448192	2023-02-28	2025-03-21		NaN	615.0	215.00	
10448193	2023-02-28	2025-03-21		NaN	620.0	220.00	
	best_offer	volume	open_interest	impl_volatility	delta	gamma	\
0	33.81	0.0	187.0		NaN	NaN	NaN
1	28.76	1.0	88.0		NaN	NaN	NaN
2	26.32	0.0	2.0		NaN	NaN	NaN
3	23.81	12.0	58.0		NaN	NaN	NaN
4	21.32	0.0	0.0		NaN	NaN	NaN
...	
10448189	205.50	0.0	0.0		NaN	NaN	NaN
10448190	210.50	0.0	0.0		NaN	NaN	NaN
10448191	215.00	0.0	0.0		NaN	NaN	NaN
10448192	220.00	0.0	0.0		NaN	NaN	NaN
10448193	225.00	0.0	0.0		NaN	NaN	NaN
	vega	theta	expiry_indicator	close	adj_open	adj_close	\
0	NaN	NaN	w	268.77	242.053393	242.893856	
1	NaN	NaN	w	268.77	242.053393	242.893856	
2	NaN	NaN	w	268.77	242.053393	242.893856	
3	NaN	NaN	w	268.77	242.053393	242.893856	
4	NaN	NaN	w	268.77	242.053393	242.893856	
...	
10448189	NaN	NaN		396.26	391.240559	390.285185	
10448190	NaN	NaN		396.26	391.240559	390.285185	
10448191	NaN	NaN		396.26	391.240559	390.285185	
10448192	NaN	NaN		396.26	391.240559	390.285185	
10448193	NaN	NaN		396.26	391.240559	390.285185	
	adj_volume	TTE					
0	86655749.0	1					
1	86655749.0	1					
2	86655749.0	1					

```

3      86655749.0    1
4      86655749.0    1
...
10448189 96141367.0 752
10448190 96141367.0 752
10448191 96141367.0 752
10448192 96141367.0 752
10448193 96141367.0 752

```

[10448194 rows x 20 columns]

Then, we filter for the closest few days to 30 DtE. We will later modify this to be the absolute closest to 30.

```
[8]: filtered_df = options_df[(options_df['TTE'] >= 27) & (options_df['TTE'] <= 33)]
filtered_df
```

	date	exdate	last_date	cp_flag	strike_price	best_bid	\
972	2018-01-02	2018-01-31	NaN	C	165.0	103.82	
973	2018-01-02	2018-01-31	2017-12-28	C	240.0	29.06	
974	2018-01-02	2018-01-31	2017-12-28	C	245.0	24.12	
975	2018-01-02	2018-01-31	2017-12-28	C	250.0	19.20	
976	2018-01-02	2018-01-31	2017-12-28	C	255.0	14.33	
...	
10444209	2023-02-28	2023-03-31	2022-06-23	P	570.0	172.26	
10444210	2023-02-28	2023-03-31	2022-11-18	P	575.0	177.25	
10444211	2023-02-28	2023-03-31	2022-06-23	P	580.0	182.23	
10444212	2023-02-28	2023-03-31	2022-11-15	P	585.0	187.22	
10444213	2023-02-28	2023-03-31	2023-01-04	P	590.0	192.20	
best_offer	volume	open_interest	impl_volatility	delta	\		
972	104.01	0.0	0.0	NaN	NaN		
973	29.26	0.0	12.0	0.193414	0.983311		
974	24.32	0.0	2.0	0.175939	0.972425		
975	19.38	0.0	152.0	0.153960	0.957185		
976	14.52	0.0	52.0	0.133378	0.926605		
...	
10444209	173.45	0.0	0.0	NaN	NaN		
10444210	178.44	0.0	0.0	NaN	NaN		
10444211	183.43	0.0	0.0	NaN	NaN		
10444212	188.42	0.0	0.0	NaN	NaN		
10444213	193.41	0.0	0.0	NaN	NaN		
gamma	vega	theta	expiry_indicator	close	adj_open	\	
972	NaN	NaN	NaN	w	268.77	242.053393	
973	0.002835	3.151210	-7.538074	w	268.77	242.053393	
974	0.004764	4.816286	-9.064763	w	268.77	242.053393	
975	0.007811	6.907079	-10.439730	w	268.77	242.053393	

```

976      0.013784  10.547560 -12.556030          w  268.77  242.053393
...
10444209      ...     NaN     NaN     NaN     ...
10444210      ...     NaN     NaN     NaN     ...
10444211      ...     NaN     NaN     NaN     ...
10444212      ...     NaN     NaN     NaN     ...
10444213      ...     NaN     NaN     NaN     ...

           adj_close  adj_volume   TTE
972        242.893856  86655749.0    29
973        242.893856  86655749.0    29
974        242.893856  86655749.0    29
975        242.893856  86655749.0    29
976        242.893856  86655749.0    29
...
10444209    390.285185  96141367.0    31
10444210    390.285185  96141367.0    31
10444211    390.285185  96141367.0    31
10444212    390.285185  96141367.0    31
10444213    390.285185  96141367.0    31

[686810 rows x 20 columns]

```

We can do the same with the dataset where contracts with nonexistent Greek calculations are dropped (we won't end up using this data set, but it will help us consider which values to initially drop for our calculations before we try to see if mock Black-Scholes values or similar can be used to fill missing Greeks/IV)

```
[9]: exdates1 = set(options_df1['exdate'])
quotedates1 = set(options_df1['date'])
```

```
[10]: len(exdates1 - quotedates1)
```

```
[10]: 31
```

```
[11]: len(quotedates1 - exdates1)
```

```
[11]: 468
```

```
[12]: len(exdates1 & quotedates1)
```

```
[12]: 830
```

Again calculating time to expiry:

```
[13]: options_df1['TTE'] = (pd.to_datetime(options_df1['exdate']) - pd.
    to_datetime(options_df1['date'])).dt.days
options_df1
```

[13]:		date	exdate	last_date	cp_flag	strike_price	best_bid	\
0		2018-01-02	2018-01-03	2018-01-02	C	267.5	1.28	
1		2018-01-02	2018-01-03	2018-01-02	C	268.0	0.85	
2		2018-01-02	2018-01-03	2018-01-02	C	268.5	0.48	
3		2018-01-02	2018-01-03	2018-01-02	C	269.0	0.21	
4		2018-01-02	2018-01-03	2018-01-02	C	269.5	0.07	
...		
9261854		2023-02-28	2025-03-21	2023-02-08	P	450.0	57.50	
9261855		2023-02-28	2025-03-21	2023-01-09	P	455.0	61.00	
9261856		2023-02-28	2025-03-21	2023-01-09	P	460.0	64.50	
9261857		2023-02-28	2025-03-21	2023-01-09	P	465.0	68.50	
9261858		2023-02-28	2025-03-21	2023-02-16	P	470.0	72.50	
		best_offer	volume	open_interest	impl_volatility	delta	\	
0		1.34	5514.0	7184.0	0.062960	0.926657		
1		0.88	14476.0	5520.0	0.060924	0.819616		
2		0.50	12516.0	5895.0	0.058946	0.633232		
3		0.22	14745.0	5761.0	0.055630	0.390490		
4		0.08	1742.0	7512.0	0.055298	0.178472		
...		
9261854		62.50	0.0	75.0	0.161451	-0.714413		
9261855		66.00	0.0	0.0	0.158546	-0.758524		
9261856		69.50	0.0	14.0	0.154637	-0.805469		
9261857		73.50	0.0	21.0	0.152118	-0.846083		
9261858		77.50	0.0	62.0	0.144876	-0.906887		
		gamma	vega	theta	expiry_indicator	close	adj_open	\
0		0.157172	1.959172	-26.221690	w	268.77	242.053393	
1		0.306547	3.697382	-44.389840	w	268.77	242.053393	
2		0.453884	5.294787	-59.509580	w	268.77	242.053393	
3		0.490317	5.399054	-56.377560	w	268.77	242.053393	
4		0.335492	3.670619	-37.772850	w	268.77	242.053393	
...		
9261854		0.007297	154.462000	1.957872		NaN	396.26	391.240559
9261855		0.007626	151.188600	2.874898		NaN	396.26	391.240559
9261856		0.007817	123.774400	4.337152		NaN	396.26	391.240559
9261857		0.007509	79.376680	6.359011		NaN	396.26	391.240559
9261858		0.006313	54.487690	10.982410		NaN	396.26	391.240559
		adj_close	adj_volume	TTE				
0		242.893856	86655749.0	1				
1		242.893856	86655749.0	1				
2		242.893856	86655749.0	1				
3		242.893856	86655749.0	1				
4		242.893856	86655749.0	1				
...					
9261854		390.285185	96141367.0	752				

```

9261855 390.285185 96141367.0 752
9261856 390.285185 96141367.0 752
9261857 390.285185 96141367.0 752
9261858 390.285185 96141367.0 752

```

[9261859 rows x 20 columns]

```

[14]: filtered_df1 = options_df1[(options_df1['TTE'] >= 27) & (options_df1['TTE'] <= 33)]
filtered_df1

```

	date	exdate	last_date	cp_flag	strike_price	best_bid	\
828	2018-01-02	2018-01-31	2017-12-28	C	240.0	29.06	
829	2018-01-02	2018-01-31	2017-12-28	C	245.0	24.12	
830	2018-01-02	2018-01-31	2017-12-28	C	250.0	19.20	
831	2018-01-02	2018-01-31	2017-12-28	C	255.0	14.33	
832	2018-01-02	2018-01-31	2017-12-28	C	260.0	9.58	
...	
9258523	2023-02-28	2023-03-31	2023-02-28	P	411.0	16.09	
9258524	2023-02-28	2023-03-31	2023-02-28	P	412.0	16.82	
9258525	2023-02-28	2023-03-31	2023-02-28	P	413.0	17.57	
9258526	2023-02-28	2023-03-31	2023-02-28	P	414.0	18.34	
9258527	2023-02-28	2023-03-31	2023-02-28	P	415.0	19.17	
	best_offer	volume	open_interest	impl_volatility	delta	\	
828	29.26	0.0	12.0	0.193414	0.983311		
829	24.32	0.0	2.0	0.175939	0.972425		
830	19.38	0.0	152.0	0.153960	0.957185		
831	14.52	0.0	52.0	0.133378	0.926605		
832	9.74	0.0	31.0	0.110864	0.868021		
...	
9258523	17.26	102.0	906.0	0.127514	-0.854706		
9258524	18.00	5.0	2911.0	0.121468	-0.882473		
9258525	18.77	7.0	2126.0	0.114016	-0.911503		
9258526	19.56	1.0	1120.0	0.103528	-0.943419		
9258527	20.31	25.0	3530.0	0.076447	-0.987661		
	gamma	vega	theta	expiry_indicator	close	adj_open	\
828	0.002835	3.151210	-7.538074	w	268.77	242.053393	
829	0.004764	4.816286	-9.064763	w	268.77	242.053393	
830	0.007811	6.907079	-10.439730	w	268.77	242.053393	
831	0.013784	10.547560	-12.556030	w	268.77	242.053393	
832	0.025454	16.200190	-14.825650	w	268.77	242.053393	
...	
9258523	0.016339	25.346960	-6.080472	m	396.26	391.240559	
9258524	0.014863	21.527260	-1.950742	m	396.26	391.240559	
9258525	0.012890	17.372680	2.619810	m	396.26	391.240559	

```

9258526  0.009995  11.793440  7.923930      m  396.26  391.240559
9258527  0.002897  2.215591  15.766890      m  396.26  391.240559

          adj_close  adj_volume   TTE
828        242.893856  86655749.0  29
829        242.893856  86655749.0  29
830        242.893856  86655749.0  29
831        242.893856  86655749.0  29
832        242.893856  86655749.0  29
...
9258523  390.285185  96141367.0  31
9258524  390.285185  96141367.0  31
9258525  390.285185  96141367.0  31
9258526  390.285185  96141367.0  31
9258527  390.285185  96141367.0  31

```

[627236 rows x 20 columns]

We need strikes for each day where we evaluate positions to cover the range of 30 days of stock movements. We'll look at calls and puts separately. We need the ATM option to have data for the next 21-30 days.

First, calculate the closest possible date to 30.

```
[15]: puts_df = filtered_df[filtered_df['cp_flag'] == 'P']
calls_df = filtered_df[filtered_df['cp_flag'] == 'C']

puts_df1 = filtered_df1[filtered_df1['cp_flag'] == 'P']
calls_df1 = filtered_df1[filtered_df1['cp_flag'] == 'C']
```

Note: Resolve DeprecationWarning by adding ', include_groups=False' to apply function.

```
[16]: puts_result_df = puts_df.groupby('date', as_index=False).
    ↪apply(find_closest_tte).reset_index(drop=True)[['date', 'TTE']].
    ↪drop_duplicates()

calls_result_df = calls_df.groupby('date', as_index=False).
    ↪apply(find_closest_tte).reset_index(drop=True)[['date', 'TTE']].
    ↪drop_duplicates()

puts_result_df1 = puts_df1.groupby('date', as_index=False).
    ↪apply(find_closest_tte).reset_index(drop=True)[['date', 'TTE']].
    ↪drop_duplicates()

calls_result_df1 = calls_df1.groupby('date', as_index=False).
    ↪apply(find_closest_tte).reset_index(drop=True)[['date', 'TTE']].
    ↪drop_duplicates()
```

```
[17]: calls_result_df
```

```
[17]:      date    TTE
0     2018-01-02    29
1     2018-01-03    30
2     2018-01-04    29
3     2018-01-05    28
4     2018-01-08    30
...
1293   2023-02-22    30
1294   2023-02-23    29
1295   2023-02-24    28
1296   2023-02-27    32
1297   2023-02-28    31
```

[1298 rows x 2 columns]

Check if there are any days where the TtE is less than 28, then check that the data matches.

```
[18]: calls_result_df1.loc[calls_result_df1['TTE'] < 28]
```

```
[18]: Empty DataFrame
Columns: [date, TTE]
Index: []
```

```
[19]: set(options_df['date']) - set(calls_result_df['date'])
```

```
[19]: set()
```

Now, we'll iterate and filter based on date and time to expiry.

```
[20]: original_dfs = {'puts': puts_df, 'calls': calls_df, 'puts1': puts_df1, 'calls1': calls_df1}
result_dfs = {'puts': puts_result_df, 'calls': calls_result_df, 'puts1': puts_result_df1, 'calls1': calls_result_df1}
filtered_dfs = merge_dataframes(original_dfs, result_dfs)
```

```
[21]: filtered_dfs['puts']
```

```
[21]:      date      exdate    last_date cp_flag  strike_price  best_bid \
0     2018-01-02  2018-01-31        NaN       P      165.0      0.00
1     2018-01-02  2018-01-31  2018-01-02       P      240.0      0.13
2     2018-01-02  2018-01-31  2018-01-02       P      245.0      0.17
3     2018-01-02  2018-01-31  2018-01-02       P      250.0      0.24
4     2018-01-02  2018-01-31  2018-01-02       P      255.0      0.36
...
128920  2023-02-28  2023-03-31  2022-06-23       P      570.0     172.26
128921  2023-02-28  2023-03-31  2022-11-18       P      575.0     177.25
128922  2023-02-28  2023-03-31  2022-06-23       P      580.0     182.23
128923  2023-02-28  2023-03-31  2022-11-15       P      585.0     187.22
```

128924	2023-02-28	2023-03-31	2023-01-04	P	590.0	192.20	
	best_offer	volume	open_interest	impl_volatility	delta	\	
0	0.01	0.0	0.0	0.535503	-0.000456		
1	0.14	147.0	1282.0	0.205954	-0.022718		
2	0.18	66.0	182.0	0.182066	-0.031889		
3	0.25	269.0	166.0	0.159122	-0.048143		
4	0.37	59.0	749.0	0.135986	-0.077530		
...		
128920	173.45	0.0	0.0	NaN	NaN		
128921	178.44	0.0	0.0	NaN	NaN		
128922	183.43	0.0	0.0	NaN	NaN		
128923	188.42	0.0	0.0	NaN	NaN		
128924	193.41	0.0	0.0	NaN	NaN		
	gamma	vega	theta	expiry_indicator	close	adj_open	\
0	0.000040	0.123239	-0.414822	w	268.77	242.053393	
1	0.003462	4.080427	-5.205224	w	268.77	242.053393	
2	0.005197	5.423385	-6.083812	w	268.77	242.053393	
3	0.008314	7.569383	-7.395249	w	268.77	242.053393	
4	0.014119	10.996340	-9.096192	w	268.77	242.053393	
...	
128920	NaN	NaN	NaN	m	396.26	391.240559	
128921	NaN	NaN	NaN	m	396.26	391.240559	
128922	NaN	NaN	NaN	m	396.26	391.240559	
128923	NaN	NaN	NaN	m	396.26	391.240559	
128924	NaN	NaN	NaN	m	396.26	391.240559	
	adj_close	adj_volume	TTE				
0	242.893856	86655749.0	29				
1	242.893856	86655749.0	29				
2	242.893856	86655749.0	29				
3	242.893856	86655749.0	29				
4	242.893856	86655749.0	29				
...				
128920	390.285185	96141367.0	31				
128921	390.285185	96141367.0	31				
128922	390.285185	96141367.0	31				
128923	390.285185	96141367.0	31				
128924	390.285185	96141367.0	31				

[128925 rows x 20 columns]

We'll choose the one closest to the money:

```
[22]: final_dfs = filter_dfs_on_criteria(filtered_dfs, select_row_with_smallest_diff)
final_dfs['calls1']
```

[22]:		date	exdate	last_date	cp_flag	strike_price	best_bid	\
0	2018-01-02	2018-01-31	2018-01-02	C	269.0	2.11		
1	2018-01-03	2018-02-02	2018-01-03	C	270.0	2.65		
2	2018-01-04	2018-02-02	2018-01-04	C	272.0	2.13		
3	2018-01-05	2018-02-02	2018-01-05	C	273.0	2.60		
4	2018-01-08	2018-02-07	2018-01-08	C	275.0	1.93		
...	
1293	2023-02-22	2023-03-24	2023-02-22	C	399.0	9.08		
1294	2023-02-23	2023-03-24	2023-02-23	C	401.0	8.79		
1295	2023-02-24	2023-03-24	2023-02-24	C	396.0	9.05		
1296	2023-02-27	2023-03-31	2023-02-27	C	398.0	8.92		
1297	2023-02-28	2023-03-31	2023-02-28	C	396.0	9.55		
	best_offer	volume	open_interest	impl_volatility	delta	gamma	\	
0	2.16	477.0	37.0	0.068871	0.512031	0.076400		
1	2.69	289.0	719.0	0.072398	0.562218	0.070177		
2	2.17	769.0	6178.0	0.071179	0.500445	0.073186		
3	2.66	1387.0	1307.0	0.074068	0.557338	0.070366		
4	1.94	924.0	3846.0	0.072545	0.454108	0.069544		
...	
1293	9.12	1875.0	344.0	0.202075	0.517410	0.018019		
1294	8.82	936.0	923.0	0.196720	0.518997	0.018739		
1295	9.08	1151.0	225.0	0.201088	0.531177	0.018872		
1296	8.96	976.0	5244.0	0.194675	0.511410	0.018207		
1297	9.58	611.0	3758.0	0.207305	0.519384	0.017402		
	vega	theta	expiry_indicator	close	adj_open	adj_close	\	
0	30.21236	-15.22501	w	268.77	242.053393	242.893856		
1	30.55966	-15.81073	w	270.47	243.065564	244.430187		
2	30.54542	-15.79170	w	271.61	245.089905	245.460432		
3	29.90136	-16.79527	w	273.42	246.273783	247.096172		
4	31.12119	-15.69601	w	273.92	246.996762	247.548034		
...	
1293	45.10776	-66.59158	w	398.54	393.496030	392.530807		
1294	44.17986	-66.44084	w	400.66	395.505271	394.618841		
1295	43.16854	-68.26598	w	396.38	389.457850	390.403375		
1296	46.44293	-62.65579	m	397.73	393.840753	391.733020		
1297	45.37965	-66.87704	m	396.26	391.240559	390.285185		
	adj_volume	TTE						
0	86655749.0	29						
1	90070416.0	30						
2	80595402.0	29						
3	83468662.0	28						
4	57288979.0	30						
...						
1293	83574386.0	30						

```

1294 95842681.0 29
1295 108144866.0 28
1296 80318244.0 32
1297 96141367.0 31

```

[1298 rows x 20 columns]

Then, we'll find the close date of each of the contracts that we simulate (ie. approximately one week prior to expiry. We choose this because close to expiry, delta and gamma are far more difficult to hedge on a day-to-day basis and we do not have minute bars).

```
[23]: final_dfs = calculate_closest_dates(final_dfs)
final_dfs['calls1']
```

	date	exdate	last_date	cp_flag	strike_price	best_bid	\
0	2018-01-02	2018-01-31	2018-01-02	C	269.0	2.11	
1	2018-01-03	2018-02-02	2018-01-03	C	270.0	2.65	
2	2018-01-04	2018-02-02	2018-01-04	C	272.0	2.13	
3	2018-01-05	2018-02-02	2018-01-05	C	273.0	2.60	
4	2018-01-08	2018-02-07	2018-01-08	C	275.0	1.93	
...	
1293	2023-02-22	2023-03-24	2023-02-22	C	399.0	9.08	
1294	2023-02-23	2023-03-24	2023-02-23	C	401.0	8.79	
1295	2023-02-24	2023-03-24	2023-02-24	C	396.0	9.05	
1296	2023-02-27	2023-03-31	2023-02-27	C	398.0	8.92	
1297	2023-02-28	2023-03-31	2023-02-28	C	396.0	9.55	
	best_offer	volume	open_interest	impl_volatility	...	gamma	\
0	2.16	477.0	37.0	0.068871	...	0.076400	
1	2.69	289.0	719.0	0.072398	...	0.070177	
2	2.17	769.0	6178.0	0.071179	...	0.073186	
3	2.66	1387.0	1307.0	0.074068	...	0.070366	
4	1.94	924.0	3846.0	0.072545	...	0.069544	
...	
1293	9.12	1875.0	344.0	0.202075	...	0.018019	
1294	8.82	936.0	923.0	0.196720	...	0.018739	
1295	9.08	1151.0	225.0	0.201088	...	0.018872	
1296	8.96	976.0	5244.0	0.194675	...	0.018207	
1297	9.58	611.0	3758.0	0.207305	...	0.017402	
	vega	theta	expiry_indicator	close	adj_open	adj_close	\
0	30.21236	-15.22501	w	268.77	242.053393	242.893856	
1	30.55966	-15.81073	w	270.47	243.065564	244.430187	
2	30.54542	-15.79170	w	271.61	245.089905	245.460432	
3	29.90136	-16.79527	w	273.42	246.273783	247.096172	
4	31.12119	-15.69601	w	273.92	246.996762	247.548034	
...	
1293	45.10776	-66.59158	w	398.54	393.496030	392.530807	

```

1294 44.17986 -66.44084           w  400.66  395.505271  394.618841
1295 43.16854 -68.26598           w  396.38  389.457850  390.403375
1296 46.44293 -62.65579           m  397.73  393.840753  391.733020
1297 45.37965 -66.87704           m  396.26  391.240559  390.285185

      adj_volume  TTE  close_date
0     86655749.0   29  2018-01-23
1     90070416.0   30  2018-01-24
2     80595402.0   29  2018-01-25
3     83468662.0   28  2018-01-26
4     57288979.0   30  2018-01-29
...
...
1293 83574386.0   30  2023-02-28
1294 95842681.0   29  2023-02-28
1295 108144866.0  28  2023-02-28
1296 80318244.0   32  2023-02-28
1297 96141367.0   31  2023-02-28

```

[1298 rows x 21 columns]

Check for dates where values are not all present for either calls or puts for the entirety of the 20-30 days or so that we hold the contracts (straddle)

```
[24]: options_df1['date'] = pd.to_datetime(options_df1['date'])
options_df1['exdate'] = pd.to_datetime(options_df1['exdate'])

final_dfs = merge_with_options(final_dfs, options_df1)

[25]: len(final_dfs['calls1'].loc[final_dfs['calls1']['is_present'] == False])

[25]: 48

[26]: len(final_dfs['calls'].loc[final_dfs['calls']['is_present'] == False])

[26]: 48

[27]: len(final_dfs['puts'].loc[final_dfs['puts']['is_present'] == False])

[27]: 110

[28]: len(final_dfs['puts1'].loc[final_dfs['puts1']['is_present'] == False])

[28]: 110
```

So we see that missing values for puts and calls is slightly different.

```
[29]: final_dfs['puts1']
```

[29]:

	date_x	exdate	last_date_x	cp_flag	strike_price	best_bid_x	\
0	2018-01-02	2018-01-31	2018-01-02	P	269.0	2.05	
1	2018-01-03	2018-02-02	2018-01-03	P	270.0	1.77	
2	2018-01-04	2018-02-02	2018-01-04	P	272.0	2.12	
3	2018-01-05	2018-02-02	2018-01-05	P	273.0	1.81	
4	2018-01-08	2018-02-07	2018-01-08	P	275.0	2.53	
...	
1293	2023-02-22	2023-03-24	2023-02-22	P	399.0	9.67	
1294	2023-02-23	2023-03-24	2023-02-23	P	401.0	8.83	
1295	2023-02-24	2023-03-24	2023-02-24	P	396.0	8.81	
1296	2023-02-27	2023-03-31	2023-02-27	P	398.0	9.23	
1297	2023-02-28	2023-03-31	2023-02-28	P	396.0	8.15	
	best_offer_x	volume_x	open_interest_x	impl_volatility_x	...	\	
0	2.10	198.0	33.0	0.069577	...		
1	1.80	1060.0	330.0	0.069568	...		
2	2.15	416.0	304.0	0.068201	...		
3	1.85	509.0	149.0	0.071740	...		
4	2.56	82.0	269.0	0.067924	...		
...	
1293	9.71	2678.0	1317.0	0.201626	...		
1294	8.86	1233.0	1566.0	0.186139	...		
1295	8.84	3156.0	1086.0	0.198990	...		
1296	9.26	1568.0	4088.0	0.188579	...		
1297	8.17	1375.0	6481.0	0.174174	...		
	vega_x	theta_x	expiry_indicator_x	close_x	adj_open_x	\	
0	30.11151	-11.46749	w	268.77	242.053393		
1	30.52431	-11.33461	w	270.47	243.065564		
2	30.40449	-11.28616	w	271.61	245.089905		
3	29.86764	-12.34094	w	273.42	246.273783		
4	30.79770	-10.80407	w	273.92	246.996762		
...	
1293	45.30915	-47.38424	w	398.54	393.496030		
1294	44.77837	-44.24160	w	400.66	395.505271		
1295	43.53449	-48.60954	w	396.38	389.457850		
1296	46.60089	-42.60824	m	397.73	393.840753		
1297	45.68817	-39.53407	m	396.26	391.240559		
	adj_close_x	adj_volume_x	TTE_x	close_date	is_present		
0	242.893856	86655749.0	29	2018-01-23	True		
1	244.430187	90070416.0	30	2018-01-24	True		
2	245.460432	80595402.0	29	2018-01-25	True		
3	247.096172	83468662.0	28	2018-01-26	True		
4	247.548034	57288979.0	30	2018-01-29	True		
...	
1293	392.530807	83574386.0	30	2023-02-28	True		

```

1294 394.618841    95842681.0      29  2023-02-28      True
1295 390.403375    108144866.0     28  2023-02-28      True
1296 391.733020    80318244.0      32  2023-02-28      True
1297 390.285185    96141367.0      31  2023-02-28      True

```

[1298 rows x 22 columns]

Checkpoint to save the potential call/put options (which we will open in tandem as long or short positions on straddles).

```
[30]: for key, df in final_dfs.items():
    csv_filename = f"{key}.csv"
    df.to_csv(csv_filename, index=False)
    print(f"DataFrame '{key}' has been saved to '{csv_filename}'")
```

```

DataFrame 'puts' has been saved to 'puts.csv'
DataFrame 'calls' has been saved to 'calls.csv'
DataFrame 'puts1' has been saved to 'puts1.csv'
DataFrame 'calls1' has been saved to 'calls1.csv'
```

1.6.4 3.1: Checking which dates are missing intermediate data - continued data from C3

```
[31]: calls_in_puts = check_dates_in_calls_and_puts('calls', 'puts', final_dfs)
calls1_in_puts1 = check_dates_in_calls_and_puts('calls1', 'puts1', final_dfs)

print(f"All dates in 'calls' with is_present=False are in 'puts' with "
      "is_present=False: {calls_in_puts}")
print(f"All dates in 'calls1' with is_present=False are in 'puts1' with "
      "is_present=False: {calls1_in_puts1}")
```

```

All dates in 'calls' with is_present=False are in 'puts' with is_present=False:
False
All dates in 'calls1' with is_present=False are in 'puts1' with
is_present=False: False
```

```
[32]: count_unique_in_calls, count_unique_in_puts, count_overlapping = ↴
      count_dates_comparison('calls', 'puts', final_dfs)
count_unique_in_calls1, count_unique_in_puts1, count_overlapping1 = ↴
      count_dates_comparison('calls1', 'puts1', final_dfs)

print(f"Unique dates in 'calls' not in 'puts' where is_present=False: "
      "↪{count_unique_in_calls}")
print(f"Unique dates in 'puts' not in 'calls' where is_present=False: "
      "↪{count_unique_in_puts}")
print(f"Overlapping dates in 'calls' and 'puts' where is_present=False: "
      "↪{count_overlapping}")
```

```

print(f"Unique dates in 'calls1' not in 'puts1' where is_present=False: {count_unique_in_calls1}")
print(f"Unique dates in 'puts1' not in 'calls1' where is_present=False: {count_unique_in_puts1}")
print(f"Overlapping dates in 'calls1' and 'puts1' where is_present=False: {count_overlapping1}")

```

```

Unique dates in 'calls' not in 'puts' where is_present=False: 48
Unique dates in 'puts' not in 'calls' where is_present=False: 110
Overlapping dates in 'calls' and 'puts' where is_present=False: 0
Unique dates in 'calls1' not in 'puts1' where is_present=False: 48
Unique dates in 'puts1' not in 'calls1' where is_present=False: 110
Overlapping dates in 'calls1' and 'puts1' where is_present=False: 0

```

With this, unfortunately we do see that these sets are actually exclusive, so we will ignore them for now and for the final version, we'll try to fill them in with Black-Scholes calculations if they are close enough to the other provided data values and ignore these dates as starting points otherwise.

This concludes the initial data structuring. Further data structuring will be done in the simulation section. We now have the following important files to be retrieved when necessary:

- * combinedata.csv
- * puts.csv
- * calls.csv

1.7 5. Initial Graphs and Basic Data Exploration

We graph several different aspects of the data to take an initial look at the structures within our data.

Note that each individual graphing was done by a different group member working asynchronously, so each subsection has data imports such that it should work as a standalone if the correct data processing functions had been run to obtain the .csv files. Do note that because of this, if running the notebook directly you may end up importing multiple copies of the same data under different variable names.

1.7.1 At-The-Money, over time

We graph various greeks and metrics for ATM options over time.

```
[2]: calls_filepath = 'calls.csv'
puts_filepath = 'puts.csv'
calls_df = pd.read_csv(calls_filepath)
puts_df = pd.read_csv(puts_filepath)
```

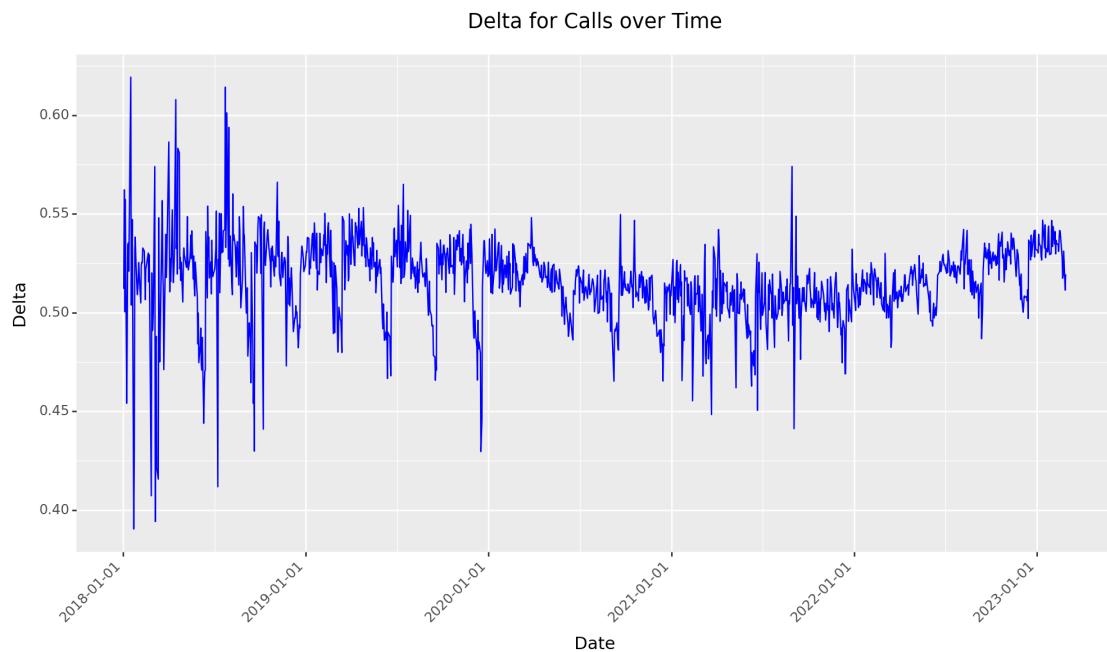
```
[3]: # Add a 'type' column to distinguish between calls and puts
calls_df['Type'] = 'Calls'
puts_df['Type'] = 'Puts'

# Combine into a single DataFrame
combined_df = pd.concat([calls_df, puts_df], ignore_index=True)

combined_df['date_x'] = pd.to_datetime(combined_df['date_x'])
```

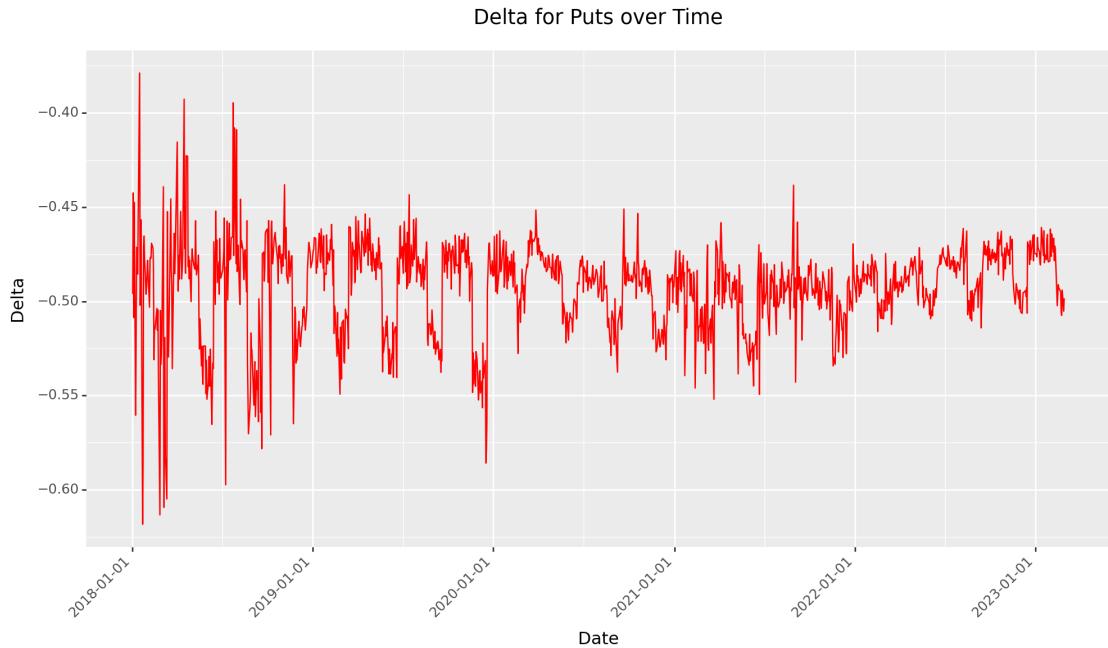
```
calls_df['date_x'] = pd.to_datetime(calls_df['date_x'])
puts_df['date_x'] = pd.to_datetime(puts_df['date_x'])
```

```
[4]: p_delta_calls = (ggplot(calls_df, aes(x='date_x', y='delta_x')) +
                      geom_line(color='blue') +
                      labs(title='Delta for Calls over Time', x='Date', y='Delta',
                           color='Type') +
                      theme(axis_text_x=element_text(rotation=45, hjust=1),
                            figure_size=(10, 6),
                            plot_title=element_text(ha='center')))
p_delta_calls
```



```
[4]: <Figure Size: (1000 x 600)>
```

```
[5]: p_delta_puts = (ggplot(puts_df, aes(x='date_x', y='delta_x')) +
                     geom_line(color='red') +
                     labs(title='Delta for Puts over Time', x='Date', y='Delta',
                          color='Type') +
                     theme(axis_text_x=element_text(rotation=45, hjust=1),
                           figure_size=(10, 6),
                           plot_title=element_text(ha='center')))
p_delta_puts
```



[5]: <Figure Size: (1000 x 600)>

We can see that at-the-money calls and puts center around a magnitude of 0.5 delta. There also appears seasonality.

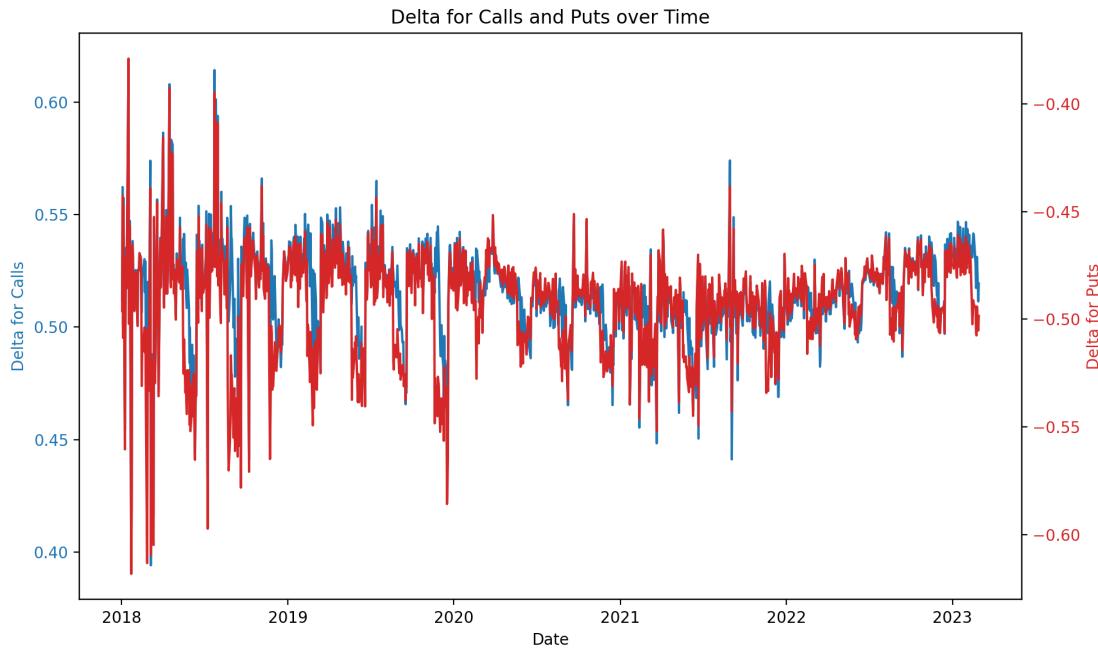
```
[6]: # Create the plot
fig, ax1 = plt.subplots(figsize=(10, 6))

# Plot Delta for Calls on the left y-axis
color = 'tab:blue'
ax1.set_xlabel('Date')
ax1.set_ylabel('Delta for Calls', color=color)
ax1.plot(calls_df['date_x'], calls_df['delta_x'], color=color)
ax1.tick_params(axis='y', labelcolor=color)

# Create a second y-axis for Delta for Puts
ax2 = ax1.twinx()
color = 'tab:red'
ax2.set_ylabel('Delta for Puts', color=color)
ax2.plot(puts_df['date_x'], puts_df['delta_x'], color=color)
ax2.tick_params(axis='y', labelcolor=color)

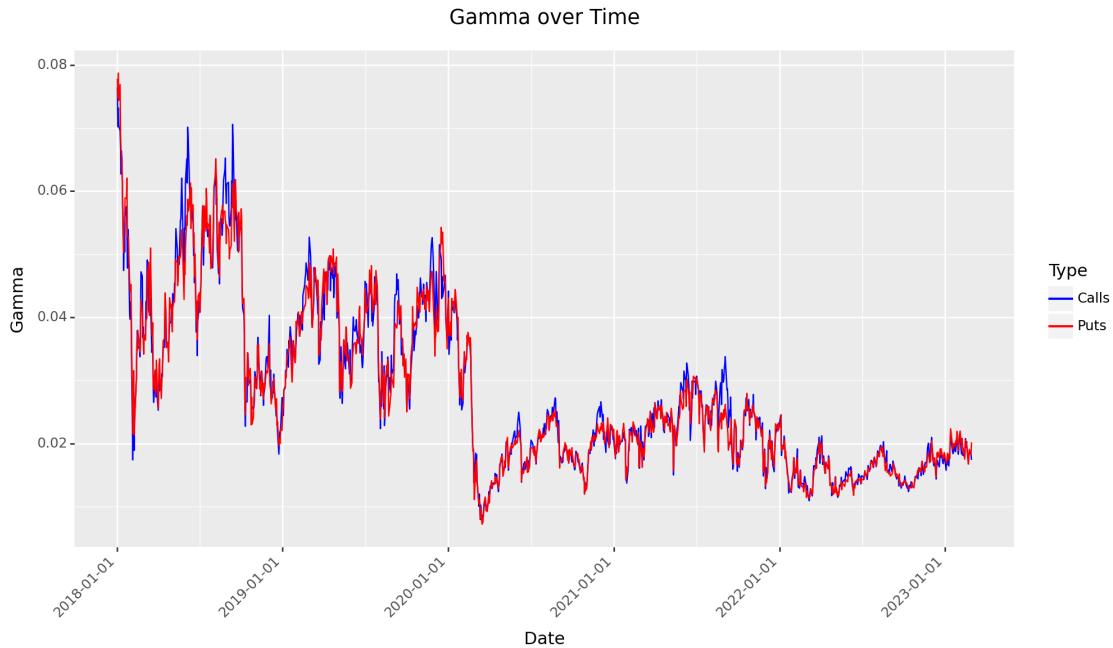
# Title and layout adjustments
plt.title('Delta for Calls and Puts over Time')
fig.tight_layout()
```

```
plt.show()
```



We can see that at-the-money calls and puts have similar gamma across multiple times. There also appears to be a regime shift, where gamma is noticeably lower post-COVID.

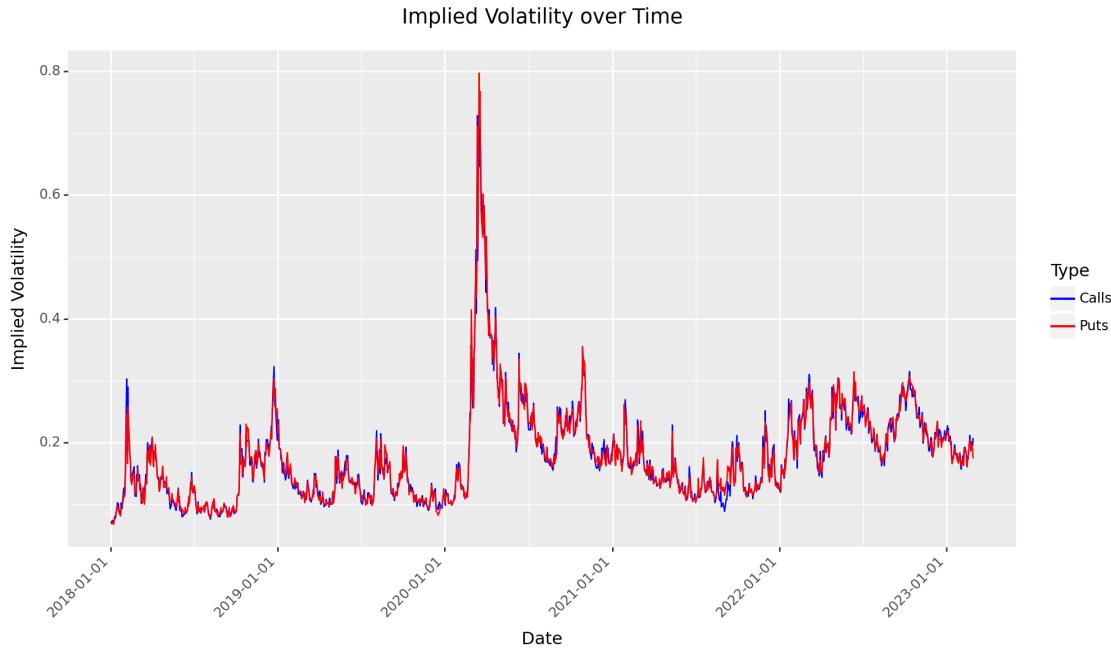
```
[7]: p_gamma = (ggplot(combined_df, aes(x='date_x', y='gamma_x', color='Type')) +  
    geom_line() +  
    labs(title='Gamma over Time', x='Date', y='Gamma', color='Type') +  
    theme(axis_text_x=element_text(rotation=45, hjust=1),  
          figure_size=(10, 6),  
          plot_title=element_text(ha='center')) +  
    scale_color_manual(values={"Calls": "blue", "Puts": "red"}))  
p_gamma
```



[7]: <Figure Size: (1000 x 600)>

We can see that implied volatility has remained largely flat until the spike during COVID, and it has largely regressed back to pre-pandemic levels albeit perhaps slightly elevated.

```
[8]: p_impl_volatility = (ggplot(combined_df, aes(x='date_x', y='impl_volatility_x',
    ↴color='Type')) +
    geom_line() +
    labs(title='Implied Volatility over Time', x='Date',
    ↴y='Implied Volatility', color='Type') +
    theme(axis_text_x=element_text(rotation=45, hjust=1),
        figure_size=(10, 6),
        plot_title=element_text(ha='center')) +
    scale_color_manual(values={"Calls": "blue", "Puts": ↴
    ↴"red"}))
p_impl_volatility
```



[8]: <Figure Size: (1000 x 600)>

1.7.2 Delta, Gamma, and IV for Individual Pairs opened ATM

We'll select a few examples of individual option pairs that we may use for straddles from the beginning, middle, and end sections of the data, and take a look at their progression over time (of the same contracts for our holding period).

```
[2]: calls = pd.read_csv('calls.csv')
puts = pd.read_csv('puts.csv')
datapull = pd.read_csv('spy_tickerdata.csv')
csv_file_path = 'option_data.csv'
spydata = pd.read_csv(csv_file_path)
```

```
[3]: spydata['date']=pd.to_datetime(spydata['date'], format = '%Y-%m-%d')
spydata['exdate']=pd.to_datetime(spydata['exdate'], format = '%Y-%m-%d')
spydata['dte'] = (spydata['exdate'] - spydata['date']).dt.days
mapping = {'C': 'Call', 'P': 'Put'}
spydata['Type'] = spydata['cp_flag'].map(mapping)
```

```
[4]: datapull[datapull['date'] == '2018-03-06']
```

	ticker	date	open	high	low	close	volume	dividend
6104	SPY	2018-03-06	273.3	273.39	271.18	272.88	78407902.0	0.0
	split	adj_open	adj_high	adj_low	adj_close	adj_volume		

```
6104    1.0  246.987725  247.06906  245.071831  246.608161  78407902.0
```

```
[5]: calls[calls['date_x'] == '2021-02-08']
```

```
[5]:      date_x      exdate last_date_x cp_flag  strike_price  best_bid_x  \
780  2021-02-08  2021-03-10  2021-02-08       C          391.0        6.8

      best_offer_x  volume_x  open_interest_x  impl_volatility_x  ...  \
780        6.88      19.0            3.0           0.157962  ...

      vega_x  theta_x  expiry_indicator_x  close_x  adj_open_x  adj_close_x  \
780  44.66742 -43.18171                  w      390.51   372.256242   373.442045

      adj_volume_x  TTE_x  close_date  is_present
780     37887680.0      30  2021-03-01       True
```

```
[1 rows x 22 columns]
```

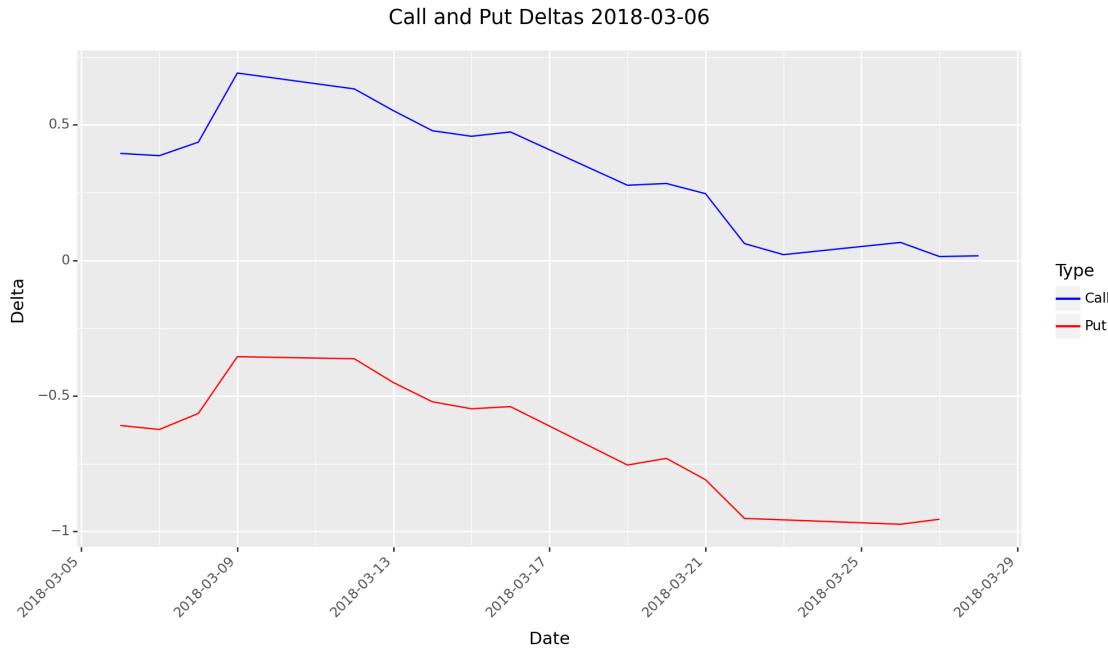
```
[6]: first_options = spydata[(spydata['exdate'] == '2018-04-04') & (spydata['dte'] <
    ↪30) & (spydata['dte'] >=7) & (spydata['strike_price'] == 275000)]
middle_options = spydata[(spydata['exdate'] == '2021-03-10') & (spydata['dte'] <
    ↪30) & (spydata['dte'] >=7) & (spydata['strike_price'] == 391000)]
last_options = spydata[(spydata['exdate'] == '2022-07-06') & (spydata['dte'] <
    ↪30) & (spydata['dte'] >=7) & (spydata['strike_price'] == 412000)]
```

```
[7]: first_options = first_options.dropna()
middle_options = middle_options.dropna()
last_options = last_options.dropna()
```

From the Delta Graphs below we can see that our positions open with call and put deltas each close to one half. As we hold the positions, the underlying moves up or down, and we can see the delta values diverge from one half and go either up or down. Our trading strategy will rebalance by buying and selling the underlying as the delta values change to keep a consistent portfolio Gamma.

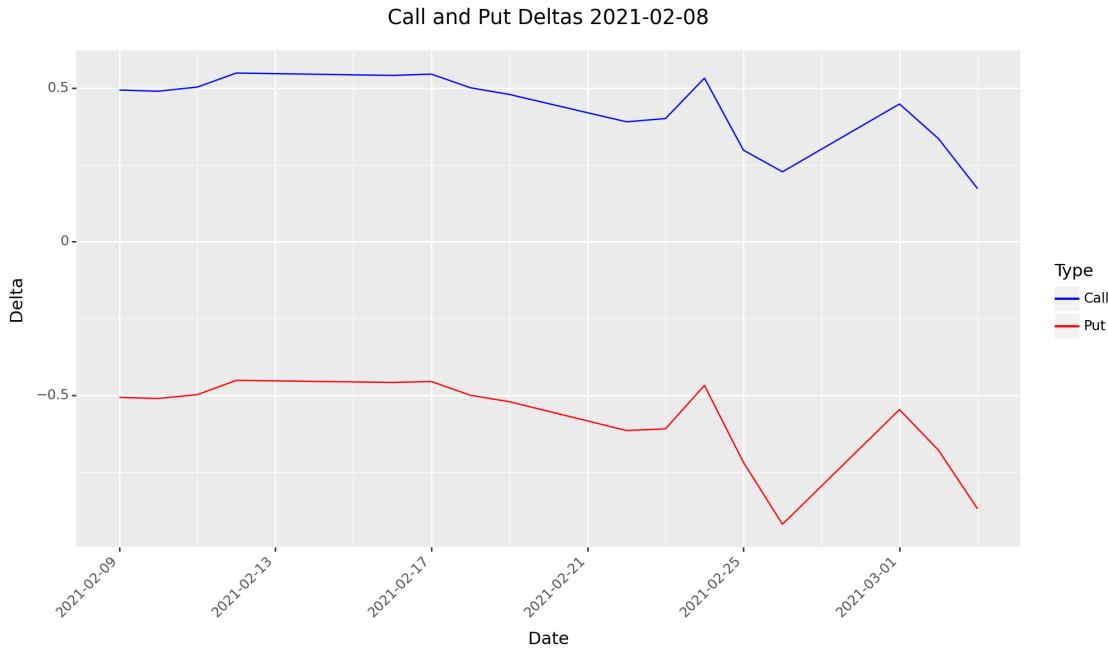
Note that under put-call parity, the Delta of a call option is always exactly 1 greater than the delta of a put option. In practice, this is almost always true, but with slight delays or lags/exceptions between time points.

```
[8]: p_d_1 = (ggplot(first_options, aes(x='date', y='delta', color='Type')) +
    geom_line() +
    labs(title='Call and Put Deltas 2018-03-06', x='Date', y='Delta', ↪
    ↪color='Type') +
    theme(axis_text_x=element_text(rotation=45, hjust=1),
          figure_size=(10, 6),
          plot_title=element_text(ha='center')) +
    scale_color_manual(values={"Call": "blue", "Put": "red"}))
p_d_1
```



[8]: <Figure Size: (1000 x 600)>

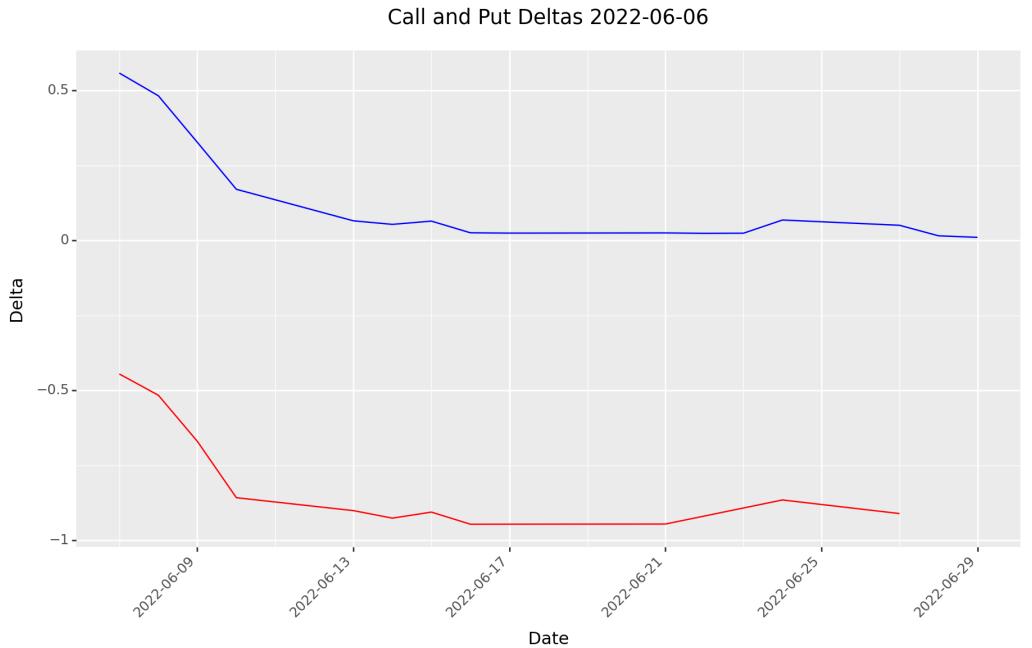
```
[9]: p_d_2 = (ggplot(middle_options, aes(x='date', y='delta', color='Type')) +
            geom_line() +
            labs(title='Call and Put Deltas 2021-02-08', x='Date', y='Delta', color='Type') +
            theme(axis_text_x=element_text(rotation=45, hjust=1),
                  figure_size=(10, 6),
                  plot_title=element_text(ha='center')) +
            scale_color_manual(values={"Call": "blue", "Put": "red"}))
p_d_2
```



[9]: <Figure Size: (1000 x 600)>

```
[10]: p_d_3 = (ggplot(last_options, aes(x='date', y='delta', color='Type')) +
            geom_line() +
            labs(title='Call and Put Deltas 2022-06-06', x='Date', y='Delta', color='Type') +
            theme(axis_text_x=element_text(rotation=45, hjust=1),
                  figure_size=(10, 6),
                  plot_title=element_text(ha='center')) +
            scale_color_manual(values={"Call": "blue", "Put": "red"}))
```

p_d_3



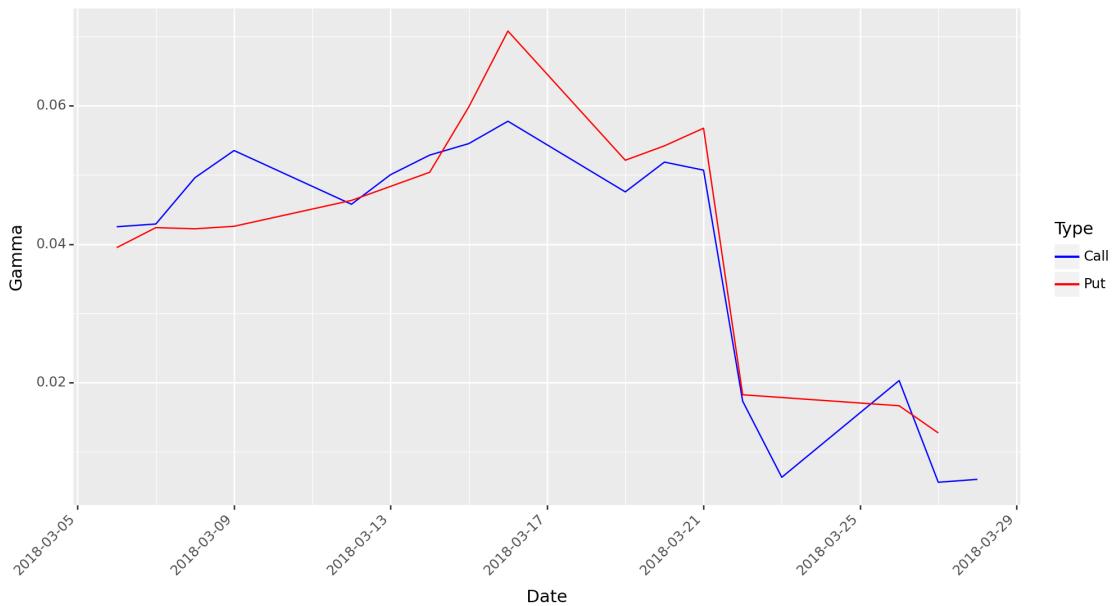
[10]: <Figure Size: (1000 x 600)>

The graphs below show how Call and Put Gammas may change while we hold them. If the price of the underlying remains constant, the Gamma value will increase; if the price of the underlying decreases, the Gamma value will decrease. Our strategy hopes to identify times when Gamma will increase or decrease with a greater magnitude than the rest of the market assumes. If we anticipate more volatility in the underlying than the market does, we will go long Gamma. Conversely, if we anticipate less volatility than the market does, we will go short Gamma.

```
[11]: p_g_1 = (ggplot(first_options, aes(x='date', y='gamma', color='Type')) +
  geom_line() +
  labs(title='Call and Put Gammas 2018-03-06', x='Date', y='Gamma', color='Type') +
  theme(axis_text_x=element_text(rotation=45, hjust=1),
        figure_size=(10, 6),
        plot_title=element_text(ha='center')) +
  scale_color_manual(values={"Call": "blue", "Put": "red"}))
```

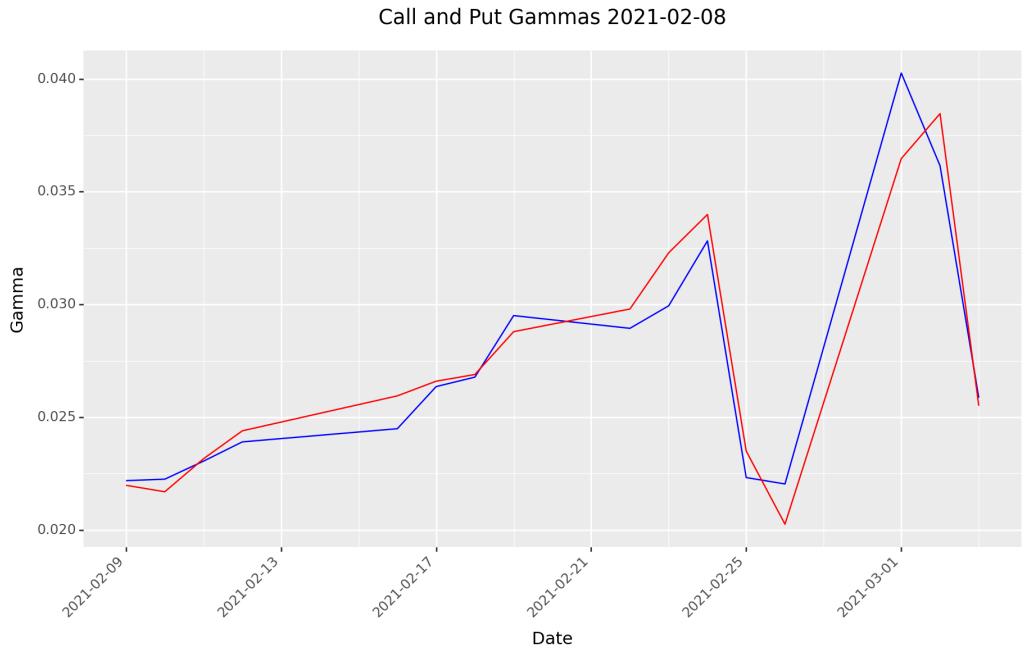
p_g_1

Call and Put Gammas 2018-03-06



[11]: <Figure Size: (1000 x 600)>

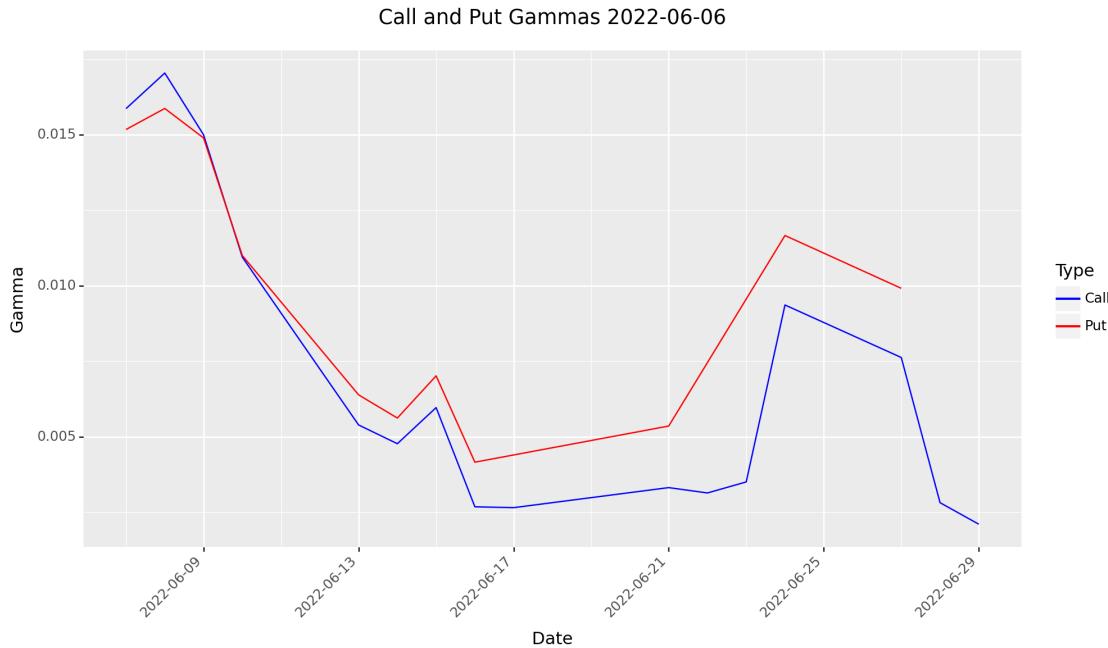
```
[12]: p_g_2 = (ggplot(middle_options, aes(x='date', y='gamma', color='Type')) +  
    geom_line() +  
    labs(title='Call and Put Gammas 2021-02-08', x='Date', y='Gamma',  
    color='Type') +  
    theme(axis_text_x=element_text(rotation=45, hjust=1),  
    figure_size=(10, 6),  
    plot_title=element_text(ha='center')) +  
    scale_color_manual(values={"Call": "blue", "Put": "red"}))  
p_g_2
```



[12]: <Figure Size: (1000 x 600)>

```
[13]: p_g_3 = (ggplot(last_options, aes(x='date', y='gamma', color='Type')) +
             geom_line() +
             labs(title='Call and Put Gammas 2022-06-06', x='Date', y='Gamma', color='Type') +
             theme(axis_text_x=element_text(rotation=45, hjust=1),
                   figure_size=(10, 6),
                   plot_title=element_text(ha='center')) +
             scale_color_manual(values={"Call": "blue", "Put": "red"}))
```

p_g_3



[13]: <Figure Size: (1000 x 600)>

1.7.3 Volume and Open Interest for Individual Days

We can also visualize the volume and open interest for various strikes on single days - for consistency, we can take a look at the same few days we visualized above.

```
[2]: data = pd.read_csv("option_data.csv")

data['date'] = pd.to_datetime(data['date'])

specific_dates = ['2018-03-06', '2021-02-08', '2022-06-06']

for specific_date in specific_dates:
    data_on_specific_date = data[data['date'] == specific_date]

    calls = data_on_specific_date[data_on_specific_date['cp_flag'] == 'C']
    puts = data_on_specific_date[data_on_specific_date['cp_flag'] == 'P']

    puts_volume = puts.copy()
    puts_oi = puts.copy()
    puts_volume['volume'] = -puts_volume['volume']
    puts_oi['open_interest'] = -puts_oi['open_interest']

    combined_volume = pd.concat([calls, puts_volume]).sort_values(by='strike_price')
```

```

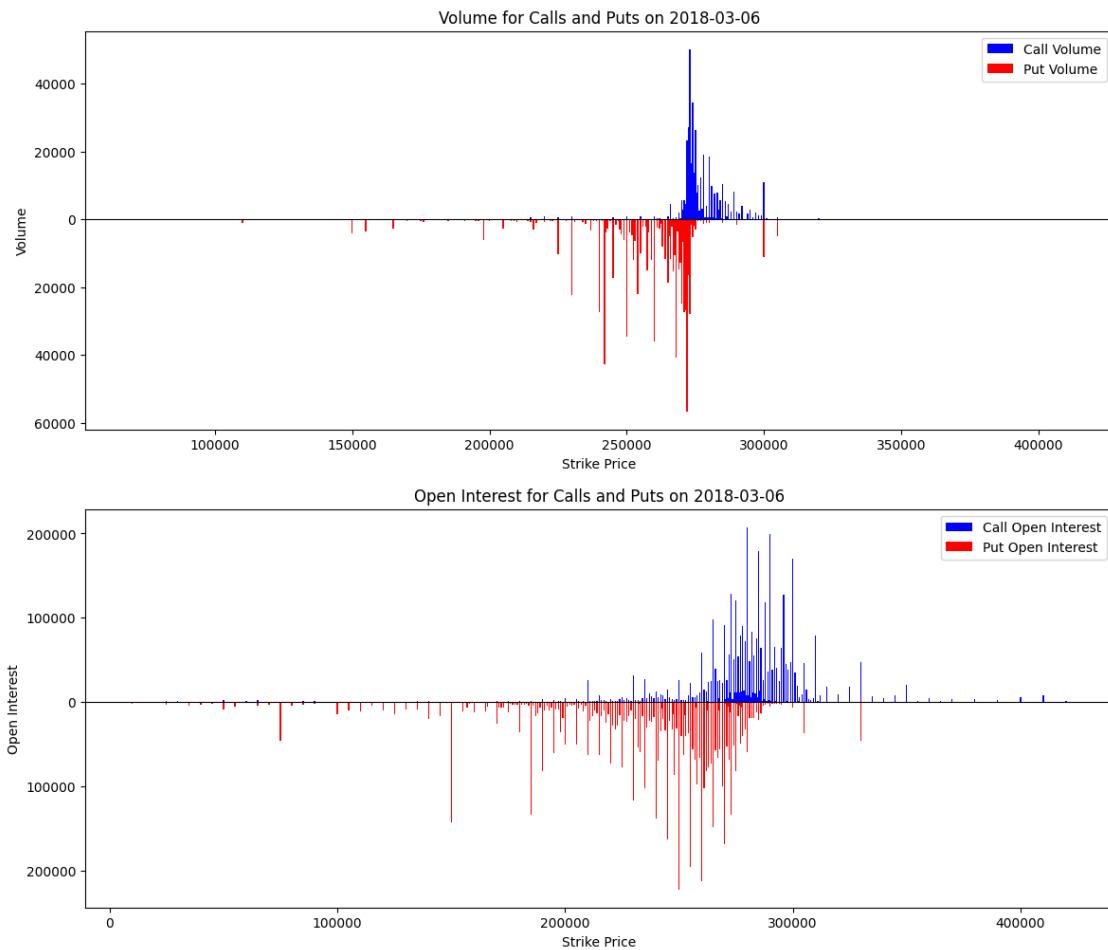
combined_oi = pd.concat([calls, puts_oi]).sort_values(by='strike_price')

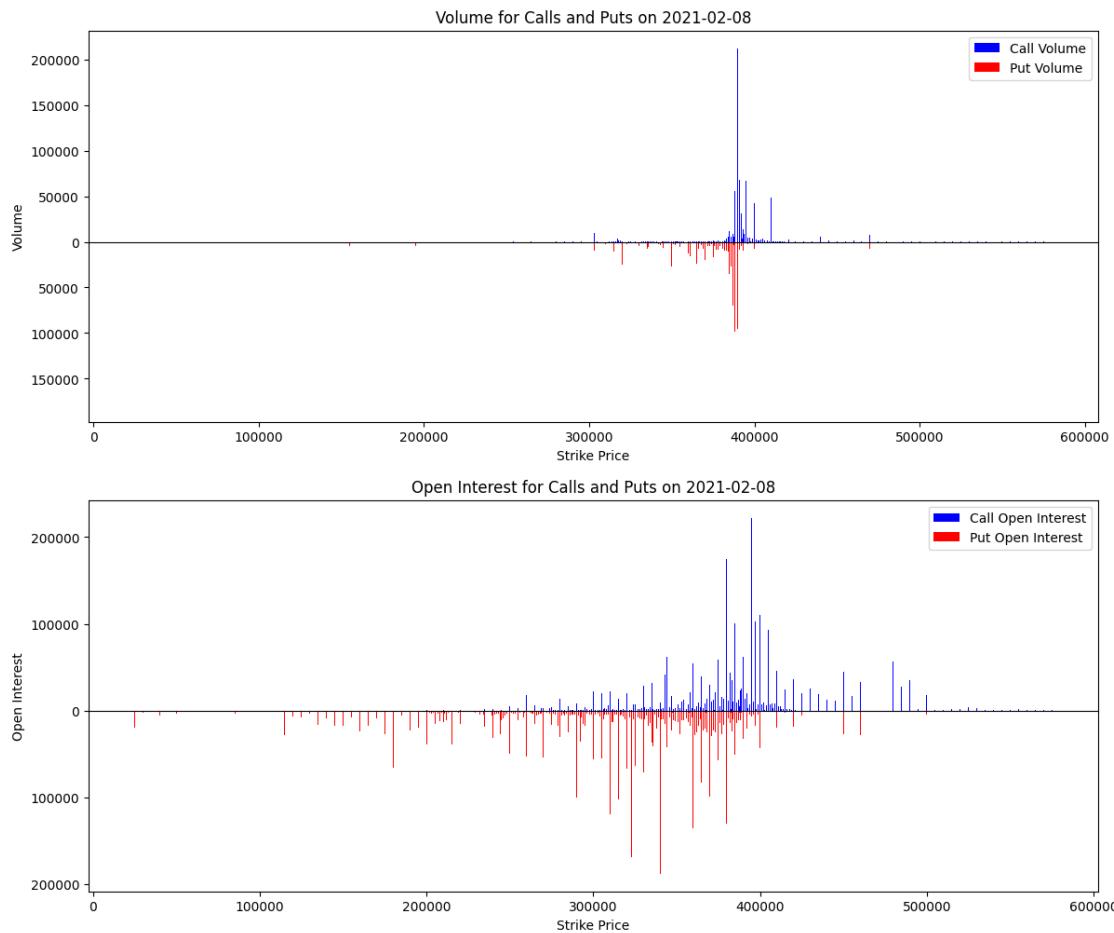
fig, axs = plt.subplots(2, 1, figsize=(14, 12))

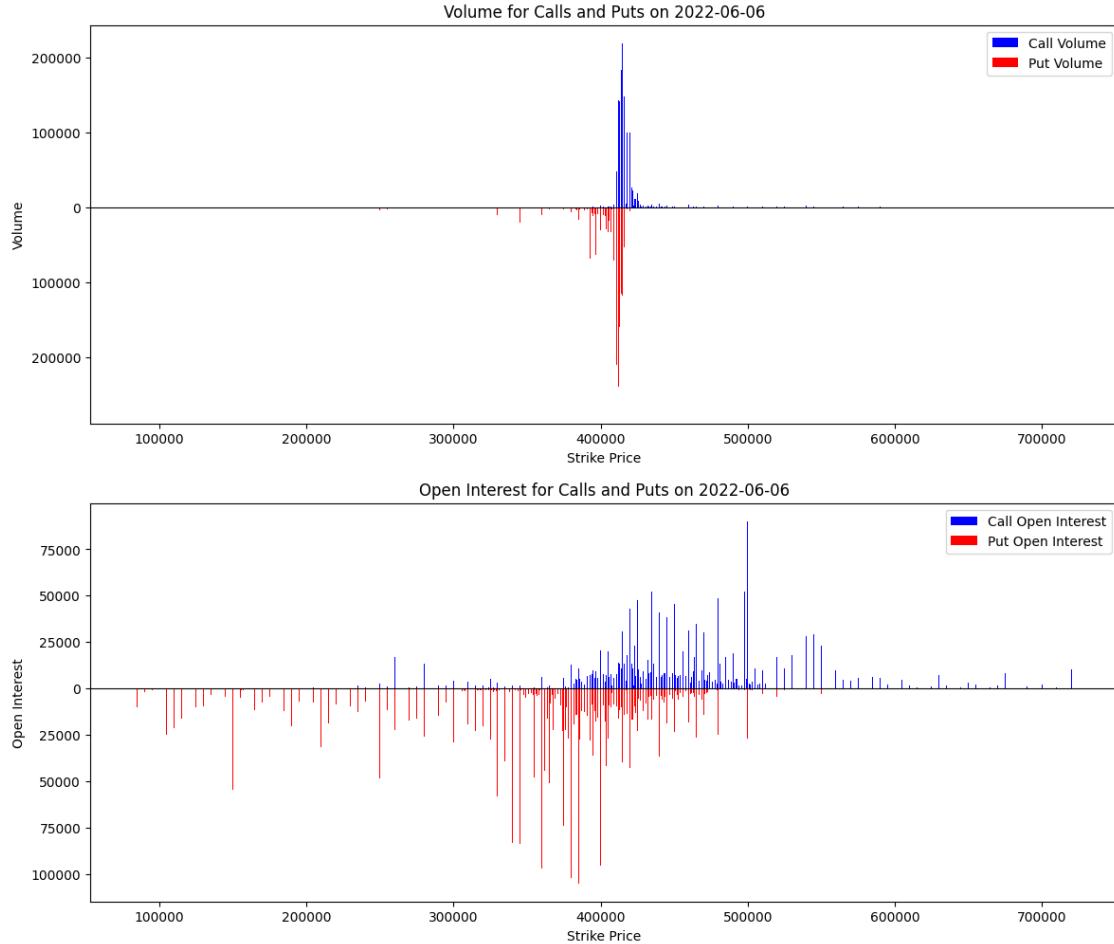
axs[0].bar(combined_volume['strike_price'][combined_volume['volume'] > 0], □
            ↵combined_volume['volume'][combined_volume['volume'] > 0], color='blue', □
            ↵width=500, label='Call Volume')
axs[0].bar(combined_volume['strike_price'][combined_volume['volume'] < 0], □
            ↵combined_volume['volume'][combined_volume['volume'] < 0], color='red', □
            ↵width=500, label='Put Volume')
axs[0].set_title(f'Volume for Calls and Puts on {specific_date}')
axs[0].set_xlabel('Strike Price')
axs[0].set_ylabel('Volume')
axs[0].axhline(0, color='black', linewidth=0.8)
axs[0].get_yaxis().set_major_formatter(plt.FuncFormatter(lambda x, _: □
            ↵int(abs(x))))
axs[0].legend()

axs[1].bar(combined_oi['strike_price'][combined_oi['open_interest'] > 0], □
            ↵combined_oi['open_interest'][combined_oi['open_interest'] > 0], □
            ↵color='blue', width=500, label='Call Open Interest')
axs[1].bar(combined_oi['strike_price'][combined_oi['open_interest'] < 0], □
            ↵combined_oi['open_interest'][combined_oi['open_interest'] < 0], color='red', □
            ↵width=500, label='Put Open Interest')
axs[1].set_title(f'Open Interest for Calls and Puts on {specific_date}')
axs[1].set_xlabel('Strike Price')
axs[1].set_ylabel('Open Interest')
axs[1].axhline(0, color='black', linewidth=0.8)
axs[1].get_yaxis().set_major_formatter(plt.FuncFormatter(lambda x, _: □
            ↵int(abs(x))))
axs[1].legend()

```







The plots above show the volume and open interest quantities for both calls and puts for three different dates throughout the five-year time period. We first observed that the volume quantity for our chosen date in 2018 was nearly 1/5th the quantities observed in the chosen 2021 and 2022 dates near the middle and end of the time span. Individually this does not say too much other than noting that different days will have different distributions, but can provide, considering the other graphs, some additional context for different time periods. Additionally, as consistently observed for all 3 dates, quantities for volume and open interest for calls are right-skewed while quantities for volume and open interest for puts are left-skewed, ie. there are certain moves far out of the money. Additionally, the open interest values were more spread out than those for volume. Moreover, volume and open interest quantities are heavily concentrated at-the-money while the tails are heavily spread out towards out-of-the-money.

1.8 6. Further Data Manipulations and Logic for Simulating All Individual Contracts

As part of the outlined strategy, we'll be simulating the strategy and also considering using a calculation of IV found in a paper to open trades relative to the gamma.

1.8.1 Further data restructuring

This uses the previous checkpoint to load the data. For the final version, we will clean up the variables to consistently use a single variable. For now, we'll allow different variables.

```
[2]: csv_file_path = 'combinedata.csv'
data = pd.read_csv(csv_file_path)
calls_df = pd.read_csv('calls.csv')
puts_df = pd.read_csv('puts.csv')
```

```
[3]: calls_df, puts_df = preprocess_options_data(calls_df, puts_df)
```

To make our analysis easier, we'll merge the dataframes from puts and calls on the same strike for the days that we may potentially open positions at (ie. every day with data filled for the next holding period for that contract).

We'll check that they match, of course.

```
[4]: dates_strike_prices_dont_match = compare_strike_prices(calls_df, puts_df)
print(dates_strike_prices_dont_match)
```

```
Series([], Name: date, dtype: datetime64[ns])
```

```
[5]: dates_data_dont_match = compare_strike_prices_and_exdates(calls_df, puts_df)
print(dates_data_dont_match)
```

```
Series([], Name: date, dtype: datetime64[ns])
```

Now merge,

```
[6]: calls_df.rename(columns={
    'close_x': 'close',
    'adj_open_x': 'adj_open',
    'adj_close_x': 'adj_close',
    'adj_volume_x': 'adj_volume',
    'TTE_x': 'TTE',
    'last_date_x': 'last_date_c',
    'best_bid_x': 'best_bid_c',
    'best_offer_x': 'best_offer_c',
    'volume_x': 'volume_c',
    'open_interest_x': 'open_interest_c',
    'impl_volatility_x': 'impl_volatility_c',
    'delta_x': 'delta_c',
    'gamma_x': 'gamma_c',
    'vega_x': 'vega_c',
    'theta_x': 'theta_c',
    'expiry_indicator_x': 'expiry_indicator_c',
    'is_present': 'is_present_c'
}, inplace=True)
```

```

calls_df = calls_df[['date', 'exdate', 'strike_price', 'close_date', 'close',  

    ↴'adj_open', 'adj_close', 'adj_volume', 'TTE', 'last_date_c', 'best_bid_c',  

    ↴'best_offer_c', 'volume_c', 'open_interest_c', 'impl_volatility_c',  

    ↴'delta_c', 'gamma_c', 'vega_c', 'theta_c', 'expiry_indicator_c',  

    ↴'is_present_c']]

```

```

[7]: puts_df.rename(columns={  

    'last_date_x': 'last_date_p',  

    'best_bid_x': 'best_bid_p',  

    'best_offer_x': 'best_offer_p',  

    'volume_x': 'volume_p',  

    'open_interest_x': 'open_interest_p',  

    'impl_volatility_x': 'impl_volatility_p',  

    'delta_x': 'delta_p',  

    'gamma_x': 'gamma_p',  

    'vega_x': 'vega_p',  

    'theta_x': 'theta_p',  

    'expiry_indicator_x': 'expiry_indicator_p',  

    'is_present': 'is_present_p'  

}, inplace=True)  
  

puts_df_selected = puts_df[['date', 'last_date_p', 'best_bid_p',  

    ↴'best_offer_p', 'volume_p', 'open_interest_p', 'impl_volatility_p',  

    ↴'delta_p', 'gamma_p', 'vega_p', 'theta_p', 'expiry_indicator_p',  

    ↴'is_present_p']]

```

```

[8]: option_df = pd.merge(calls_df, puts_df_selected, on='date', how='left')
display(option_df)

```

	date	exdate	strike_price	close_date	close	adj_open	\	
0	2018-01-02	2018-01-31	269.0	2018-01-23	268.77	242.053393		
1	2018-01-03	2018-02-02	270.0	2018-01-24	270.47	243.065564		
2	2018-01-04	2018-02-02	272.0	2018-01-25	271.61	245.089905		
3	2018-01-05	2018-02-02	273.0	2018-01-26	273.42	246.273783		
4	2018-01-08	2018-02-07	275.0	2018-01-29	273.92	246.996762		
...		
1293	2023-02-22	2023-03-24	399.0	2023-02-28	398.54	393.496030		
1294	2023-02-23	2023-03-24	401.0	2023-02-28	400.66	395.505271		
1295	2023-02-24	2023-03-24	396.0	2023-02-28	396.38	389.457850		
1296	2023-02-27	2023-03-31	398.0	2023-02-28	397.73	393.840753		
1297	2023-02-28	2023-03-31	396.0	2023-02-28	396.26	391.240559		
	adj_close	adj_volume	TTE	last_date_c	...	best_offer_p	volume_p	\
0	242.893856	86655749.0	29	2018-01-02	...	2.10	198.0	
1	244.430187	90070416.0	30	2018-01-03	...	1.80	1060.0	
2	245.460432	80595402.0	29	2018-01-04	...	2.15	416.0	
3	247.096172	83468662.0	28	2018-01-05	...	1.85	509.0	
4	247.548034	57288979.0	30	2018-01-08	...	2.56	82.0	

```

...
1293    ...    ...
1294  392.530807  83574386.0   30  2023-02-22 ...
1295  394.618841  95842681.0   29  2023-02-23 ...
1296  390.403375  108144866.0   28  2023-02-24 ...
1297  391.733020  80318244.0   32  2023-02-27 ...
1297  390.285185  96141367.0   31  2023-02-28 ...

      open_interest_p  impl_volatility_p  delta_p  gamma_p  vega_p \
0            33.0          0.069577 -0.496183  0.077797 30.11151
1            330.0          0.069568 -0.442522  0.074853 30.52431
2            304.0          0.068201 -0.508652  0.078707 30.40449
3            149.0          0.071740 -0.447556  0.074347 29.86764
4            269.0          0.067924 -0.560465  0.076891 30.79770
...
1293        ...    ...
1294        1317.0          0.201626 -0.506205  0.017387 45.30915
1295        1566.0          0.186139 -0.507514  0.019064 44.77837
1296        1086.0          0.198990 -0.494048  0.018342 43.53449
1297        4088.0          0.188579 -0.505251  0.018253 46.60089
1297        6481.0          0.174174 -0.498412  0.020189 45.68817

      theta_p  expiry_indicator_p  is_present_p
0     -11.46749           w      True
1     -11.33461           w      True
2     -11.28616           w      True
3     -12.34094           w      True
4     -10.80407           w      True
...
1293    ...    ...
1294    -47.38424          w      True
1295    -44.24160          w      True
1296    -48.60954          w      True
1297    -42.60824          m      True
1297    -39.53407          m      True

```

[1298 rows x 33 columns]

Creating a new dataframe where start dates with missing values are removed, according to our initial analysis.

```
[9]: option_df1 = option_df[(option_df['is_present_c'] == True) &
                           (option_df['is_present_p'] == True)]
display(option_df1)
```

	date	exdate	strike_price	close_date	close	adj_open	\
0	2018-01-02	2018-01-31	269.0	2018-01-23	268.77	242.053393	
1	2018-01-03	2018-02-02	270.0	2018-01-24	270.47	243.065564	
2	2018-01-04	2018-02-02	272.0	2018-01-25	271.61	245.089905	
4	2018-01-08	2018-02-07	275.0	2018-01-29	273.92	246.996762	
5	2018-01-09	2018-02-07	275.0	2018-01-30	274.54	247.981821	

...
1293	2023-02-22	2023-03-24		399.0	2023-02-28	398.54	393.496030				
1294	2023-02-23	2023-03-24		401.0	2023-02-28	400.66	395.505271				
1295	2023-02-24	2023-03-24		396.0	2023-02-28	396.38	389.457850				
1296	2023-02-27	2023-03-31		398.0	2023-02-28	397.73	393.840753				
1297	2023-02-28	2023-03-31		396.0	2023-02-28	396.26	391.240559				
	adj_close	adj_volume	TTE	last_date_c	...	best_offer_p	volume_p	\			
0	242.893856	86655749.0	29	2018-01-02	...	2.10	198.0				
1	244.430187	90070416.0	30	2018-01-03	...	1.80	1060.0				
2	245.460432	80595402.0	29	2018-01-04	...	2.15	416.0				
4	247.548034	57288979.0	30	2018-01-08	...	2.56	82.0				
5	248.108343	57253957.0	29	2018-01-09	...	2.47	453.0				
...	\			
1293	392.530807	83574386.0	30	2023-02-22	...	9.71	2678.0				
1294	394.618841	95842681.0	29	2023-02-23	...	8.86	1233.0				
1295	390.403375	108144866.0	28	2023-02-24	...	8.84	3156.0				
1296	391.733020	80318244.0	32	2023-02-27	...	9.26	1568.0				
1297	390.285185	96141367.0	31	2023-02-28	...	8.17	1375.0				
	open_interest_p	impl_volatility_p	delta_p	gamma_p	vega_p	\					
0	33.0	0.069577	-0.496183	0.077797	30.11151						
1	330.0	0.069568	-0.442522	0.074853	30.52431						
2	304.0	0.068201	-0.508652	0.078707	30.40449						
4	269.0	0.067924	-0.560465	0.076891	30.79770						
5	229.0	0.076654	-0.510832	0.069026	30.74049						
...	\					
1293	1317.0	0.201626	-0.506205	0.017387	45.30915						
1294	1566.0	0.186139	-0.507514	0.019064	44.77837						
1295	1086.0	0.198990	-0.494048	0.018342	43.53449						
1296	4088.0	0.188579	-0.505251	0.018253	46.60089						
1297	6481.0	0.174174	-0.498412	0.020189	45.68817						
	theta_p	expiry_indicator_p	is_present_p								
0	-11.46749	w	True								
1	-11.33461	w	True								
2	-11.28616	w	True								
4	-10.80407	w	True								
5	-12.99855	w	True								
...								
1293	-47.38424	w	True								
1294	-44.24160	w	True								
1295	-48.60954	w	True								
1296	-42.60824	m	True								
1297	-39.53407	m	True								

[1140 rows x 33 columns]

Creating a checkpoint,

```
[10]: option_df1.to_csv('option_df.csv', index=False)
```

1.8.2 Simulating Individual Contracts and Hedging

In the final version, we will simulate this later on - however, this section explains the simulation process for each individual possible contract position.

`source.py` contains the original code to implement this, as `create_simulations_original` and calculates metrics as `calculate_realized_PL`. We have updated these metrics in the full strategy section; however, these simulations can still be run and saved (and returned to) via the appendix.

This is another checkpoint. If the data has been run until this point, we can directly run from this point. This section simulates ALL possible contracts that we may open and trade/balance, individually. This way we can save them as an intermediate or retrieve them, mapping them onto a single timeline when we evaluate which positions will be opened at which time using various strategies later on. Because each individual simulation will NOT change depending on which we open, we can simulate each individual contract's progress ahead of time. Then, when we map depending on where we decide to open positions, we get the sum progression of our strategy.

```
[115]: data = pd.read_csv('combinedata.csv')
options = pd.read_csv('option_df.csv')
```

Pseudocode format for mass simulation of individual date-strike rebalance. Essentially, we will:

1. Pull in `calls_df` and `puts_df`
2. For dates in both dataframes where `is_present` is true for both, perform additional analysis
3. Align calls/puts on that day, specify the contract
4. For the contract in the main dataframe, find the period of info until the end date, inclusive, for that strike, contract, and expiry
5. For each day, calculate the delta sum as the sum of the call delta and put delta
6. Based on this, calculate how many shares we must hold on that day (+ or -) and use this to calculate our share movement daily
7. Calculate the share price times shares as stock position value
8. Calculate PL of day-to-day as change in shares times diff in share price
9. Calculate current available closing price of the option straddle, which will also be used to add to the total PL with open positions (ie. price based on the closest book level, should we choose to close the position)
10. On the end date, close ALL positions

We will save each dataframe as an item in a variable () then store it under a same-directory data folder, naming each simulation after the date it was opened and whether we were long or short gamma. The function call to do this is in the appendix.

```
[ ]: data['exdate'] = pd.to_datetime(data['exdate'])
options['exdate'] = pd.to_datetime(options['exdate'])

data['exdate_str'] = data['exdate'].dt.strftime('%Y%m%d')
data['strikeID'] = data['exdate_str'] + '_' + data['strike_price'].astype(str)
data.drop(columns=['exdate_str'], inplace=True)

options['exdate_str'] = options['exdate'].dt.strftime('%Y%m%d')
options['strikeID'] = options['exdate_str'] + '_' + options['strike_price'].
    astype(str)
```

```
options.drop(columns=['exdate_str'], inplace=True)
```

```
[119]: options['date'] = pd.to_datetime(options['date'])
data['date'] = pd.to_datetime(data['date'])
```

Simulate a small subset, just to show what we expect; note that this simulates with a single option contract (for one share, 1:1). We go into much more detail later with our modified simulation, but this outlines the method used to get our results.

```
[120]: options_subset = options.head(2)
simulations1 = create_simulations_original(options_subset, data, ↴
    dropna_greeks=True)
```

```
[ ]: for date, df in simulations1.items():
    simulations1[date] = calculate_realized_PL(df, long_op=True)
```

```
[ ]: for key, df in list(simulations1.items())[:2]:
    print(f"DataFrame for {key}:")
    print(df.columns)
    display(df)
    print("\n")
```

DataFrame for 2018-01-02:

```
Index(['date', 'exdate', 'strike_price', 'expiry_indicator', 'close',
       'adj_open', 'adj_close', 'adj_volume', 'strikeID', 'cp_flag_c',
       'best_bid_c', 'best_offer_c', 'volume_c', 'open_interest_c',
       'impl_volatility_c', 'delta_c', 'gamma_c', 'vega_c', 'theta_c',
       'cp_flag_p', 'best_bid_p', 'best_offer_p', 'volume_p',
       'open_interest_p', 'impl_volatility_p', 'delta_p', 'gamma_p', 'vega_p',
       'theta_p', 'delta_sum', 'shares_held', 'sharechange', 'stock_pos',
       'avg_cost', 'realized_PL', 'option_PL', 'stock_PL'],
      dtype='object')
```

	date	exdate	strike_price	expiry_indicator	close	adj_open	\
0	2018-01-02	2018-01-31	269.0		w 268.77	242.053393	
1	2018-01-03	2018-01-31	269.0		w 270.47	243.065564	
2	2018-01-04	2018-01-31	269.0		w 271.61	245.089905	
3	2018-01-05	2018-01-31	269.0		w 273.42	246.273783	
4	2018-01-08	2018-01-31	269.0		w 273.92	246.996762	
5	2018-01-09	2018-01-31	269.0		w 274.54	247.981821	
6	2018-01-10	2018-01-31	269.0		w 274.12	247.331140	
7	2018-01-11	2018-01-31	269.0		w 276.12	248.298125	
8	2018-01-12	2018-01-31	269.0		w 277.92	249.807344	
9	2018-01-16	2018-01-31	269.0		w 276.97	252.455254	
10	2018-01-17	2018-01-31	269.0		w 279.61	251.262339	
11	2018-01-18	2018-01-31	269.0		w 279.14	252.572738	
12	2018-01-19	2018-01-31	269.0		w 280.41	252.861930	
13	2018-01-23	2018-01-31	269.0		w 283.29	255.518878	

	adj_close	adj_volume	strikeID	cp_flag_c	...	vega_p	\
0	242.893856	86655749.0	20180131_269.0	C	...	30.111510	
1	244.430187	90070416.0	20180131_269.0	C	...	28.250190	
2	245.460432	80595402.0	20180131_269.0	C	...	25.781000	
3	247.096172	83468662.0	20180131_269.0	C	...	21.481350	
4	247.548034	57288979.0	20180131_269.0	C	...	18.362690	
5	248.108343	57253957.0	20180131_269.0	C	...	17.498160	
6	247.728779	69499524.0	20180131_269.0	C	...	17.468620	
7	249.536226	62306557.0	20180131_269.0	C	...	13.118760	
8	251.162929	90789911.0	20180131_269.0	C	...	10.328780	
9	250.304392	106555142.0	20180131_269.0	C	...	11.123210	
10	252.690223	113201396.0	20180131_269.0	C	...	7.665309	
11	252.265472	100682250.0	20180131_269.0	C	...	8.089207	
12	253.413202	140895374.0	20180131_269.0	C	...	5.106381	
13	256.015926	97084700.0	20180131_269.0	C	...	2.290031	

	theta_p	delta_sum	shares_held	sharechange	stock_pos	avg_cost	\
0	-11.467490	0.015848	-0.015848	NaN	-0.015848	268.770000	
1	-11.921280	0.259359	-0.259359	-0.243511	-0.259359	270.366122	
2	-12.327160	0.391860	-0.391860	-0.132501	-0.391860	270.786719	
3	-11.877760	0.560376	-0.560376	-0.168516	-0.560376	271.578598	
4	-11.728420	0.613018	-0.613018	-0.052642	-0.613018	271.779662	
5	-13.020330	0.646666	-0.646666	-0.033648	-0.646666	271.923291	
6	-13.232750	0.626662	-0.626662	0.020004	-0.626662	271.923291	
7	-11.482330	0.780802	-0.780802	-0.154140	-0.780802	272.751774	
8	-10.552090	0.840416	-0.840416	-0.059614	-0.840416	273.118376	
9	-16.564950	0.747087	-0.747087	0.093329	-0.747087	273.118376	
10	-13.657280	0.866499	-0.866499	-0.119412	-0.866499	274.012985	
11	-16.159490	0.815875	-0.815875	0.050624	-0.815875	274.012985	
12	-10.722630	0.916563	-0.916563	-0.100688	-0.916563	274.715722	
13	-9.210225	0.921621	-0.921621	-0.005058	0.000000	274.762779	

	realized_PL	option_PL	stock_PL
0	0.000000	-0.10	0.000000
1	0.000000	0.32	-0.026942
2	0.000000	0.90	-0.322611
3	0.000000	2.00	-1.031877
4	0.000000	2.19	-1.312065
5	0.000000	2.67	-1.692137
6	-0.043943	2.29	-1.420537
7	0.000000	3.68	-2.673861
8	0.000000	5.29	-4.035362
9	-0.359468	4.73	-3.236966
10	0.000000	6.84	-5.209276
11	-0.259550	6.63	-4.442553
12	0.000000	7.41	-5.478715
13	-7.858866	10.33	-7.858866

[14 rows x 37 columns]

DataFrame for 2018-01-03:

```
Index(['date', 'exdate', 'strike_price', 'expiry_indicator', 'close',
       'adj_open', 'adj_close', 'adj_volume', 'strikeID', 'cp_flag_c',
       'best_bid_c', 'best_offer_c', 'volume_c', 'open_interest_c',
       'impl_volatility_c', 'delta_c', 'gamma_c', 'vega_c', 'theta_c',
       'cp_flag_p', 'best_bid_p', 'best_offer_p', 'volume_p',
       'open_interest_p', 'impl_volatility_p', 'delta_p', 'gamma_p', 'vega_p',
       'theta_p', 'delta_sum', 'shares_held', 'sharechange', 'stock_pos',
       'avg_cost', 'realized_PL', 'option_PL', 'stock_PL'],
      dtype='object')
```

	date	exdate	strike_price	expiry_indicator	close	adj_open	\
0	2018-01-03	2018-02-02	270.0		w 270.47	243.065564	
1	2018-01-04	2018-02-02	270.0		w 271.61	245.089905	
2	2018-01-05	2018-02-02	270.0		w 273.42	246.273783	
3	2018-01-08	2018-02-02	270.0		w 273.92	246.996762	
4	2018-01-09	2018-02-02	270.0		w 274.54	247.981821	
5	2018-01-10	2018-02-02	270.0		w 274.12	247.331140	
6	2018-01-11	2018-02-02	270.0		w 276.12	248.298125	
7	2018-01-12	2018-02-02	270.0		w 277.92	249.807344	
8	2018-01-16	2018-02-02	270.0		w 276.97	252.455254	
9	2018-01-17	2018-02-02	270.0		w 279.61	251.262339	
10	2018-01-18	2018-02-02	270.0		w 279.14	252.572738	
11	2018-01-19	2018-02-02	270.0		w 280.41	252.861930	
12	2018-01-22	2018-02-02	270.0		w 282.69	253.196308	
13	2018-01-23	2018-02-02	270.0		w 283.29	255.518878	
14	2018-01-24	2018-02-02	270.0		w 283.18	256.675645	

	adj_close	adj_volume	strikeID	cp_flag_c	...	vega_p	\
0	244.430187	90070416.0	20180202_270.0	C ...	30.524310		
1	245.460432	80595402.0	20180202_270.0	C ...	28.778260		
2	247.096172	83468662.0	20180202_270.0	C ...	25.017800		
3	247.548034	57288979.0	20180202_270.0	C ...	22.018190		
4	248.108343	57253957.0	20180202_270.0	C ...	21.022120		
5	247.728779	69499524.0	20180202_270.0	C ...	21.179100		
6	249.536226	62306557.0	20180202_270.0	C ...	16.636580		
7	251.162929	90789911.0	20180202_270.0	C ...	13.466870		
8	250.304392	106555142.0	20180202_270.0	C ...	14.348100		
9	252.690223	113201396.0	20180202_270.0	C ...	10.380940		
10	252.265472	100682250.0	20180202_270.0	C ...	10.897700		
11	253.413202	140895374.0	20180202_270.0	C ...	7.613017		
12	255.473692	91322408.0	20180202_270.0	C ...	4.818615		
13	256.015926	97084700.0	20180202_270.0	C ...	3.783040		
14	255.916517	134816117.0	20180202_270.0	C ...	3.228602		

	theta_p	delta_sum	shares_held	sharechange	stock_pos	avg_cost	\
0	-11.33461	0.119696	-0.119696	NaN	-0.119696	270.470000	
1	-12.15496	0.265768	-0.265768	-0.146072	-0.265768	271.096569	
2	-12.28042	0.457539	-0.457539	-0.191771	-0.457539	272.070402	
3	-12.28724	0.514544	-0.514544	-0.057005	-0.514544	272.275315	
4	-13.72542	0.549888	-0.549888	-0.035344	-0.549888	272.420877	
5	-14.17493	0.523363	-0.523363	0.026525	-0.523363	272.420877	
6	-12.78389	0.692574	-0.692574	-0.169211	-0.692574	273.324654	
7	-12.09646	0.768158	-0.768158	-0.075584	-0.768158	273.776819	
8	-18.15760	0.670163	-0.670163	0.097995	-0.670163	273.776819	
9	-15.57961	0.802791	-0.802791	-0.132628	-0.802791	274.740511	
10	-18.16232	0.753469	-0.753469	0.049322	-0.753469	274.740511	
11	-13.19071	0.854576	-0.854576	-0.101107	-0.854576	275.411282	
12	-12.65994	0.921580	-0.921580	-0.067004	-0.921580	275.940486	
13	-11.24559	0.891718	-0.891718	0.029862	-0.891718	275.940486	
14	-10.78056	0.898643	-0.898643	-0.006925	0.000000	275.996274	

	realized_PL	option_PL	stock_PL
0	0.000000	-0.07	0.000000
1	0.000000	0.39	-0.136453
2	0.000000	1.33	-0.617494
3	0.000000	1.49	-0.846263
4	0.000000	2.01	-1.165280
5	-0.045069	1.63	-0.934327
6	0.000000	2.83	-1.981053
7	0.000000	4.39	-3.182617
8	-0.312916	3.93	-2.452867
9	0.000000	5.92	-4.222098
10	-0.216992	5.76	-3.531870
11	0.000000	6.47	-4.488776
12	0.000000	8.47	-6.220217
13	-0.219471	9.25	-6.773165
14	-6.455605	9.08	-6.455605

[15 rows x 37 columns]

1.9 7. Calculate model-free IV to compare with IV given in data

As per [Jiang et al.] we calculate IV in a model-free way, to experiment with how using the model-free IV to open positions compares against taking a long or short gamma position through this period.

We've downloaded some additional data, with which to calculate an output `iv_calculations.csv`, and for this we require t-bill data.

[2]: `tbills = load_and_transform_tbills('tbill_data.csv')`

Finding the calls and puts we calculated earlier, to implement the model-free IV.

```
[3]: calls = load_and_transform_calls('calls.csv')
puts = load_and_transform_puts('puts.csv')
option_data = load_and_transform_option_data('option_data.csv')
```

We also perform some small manipulations - we will use the option midprices, and find days to expiry based on the expiry date.

```
[4]: option_data
```

```
[4]:
```

	date	exdate	last_date	cp_flag	strike_price	best_bid	\
0	2018-01-02	2018-01-03	2017-12-28	C	235000.0	33.59	
1	2018-01-02	2018-01-03	2018-01-02	C	240000.0	28.59	
2	2018-01-02	2018-01-03	2017-12-27	C	242500.0	26.09	
3	2018-01-02	2018-01-03	2018-01-02	C	245000.0	23.59	
4	2018-01-02	2018-01-03		NaN	247500.0	21.08	
...	\
10448189	2023-02-28	2025-03-21	2022-12-28	P	600000.0	200.50	
10448190	2023-02-28	2025-03-21		NaN	605000.0	205.50	
10448191	2023-02-28	2025-03-21		NaN	610000.0	210.00	
10448192	2023-02-28	2025-03-21		NaN	615000.0	215.00	
10448193	2023-02-28	2025-03-21		NaN	620000.0	220.00	
							\
	best_offer	volume	open_interest	impl_volatility	delta	gamma	\
0	33.81	0.0	187.0		NaN	NaN	NaN
1	28.76	1.0	88.0		NaN	NaN	NaN
2	26.32	0.0	2.0		NaN	NaN	NaN
3	23.81	12.0	58.0		NaN	NaN	NaN
4	21.32	0.0	0.0		NaN	NaN	NaN
...	\
10448189	205.50	0.0	0.0		NaN	NaN	NaN
10448190	210.50	0.0	0.0		NaN	NaN	NaN
10448191	215.00	0.0	0.0		NaN	NaN	NaN
10448192	220.00	0.0	0.0		NaN	NaN	NaN
10448193	225.00	0.0	0.0		NaN	NaN	NaN
							\
	vega	theta	expiry_indicator	midpt	dte		
0	NaN	NaN	w	33.700	1		
1	NaN	NaN	w	28.675	1		
2	NaN	NaN	w	26.205	1		
3	NaN	NaN	w	23.700	1		
4	NaN	NaN	w	21.200	1		
...		\
10448189	NaN	NaN		NaN	203.000	752	
10448190	NaN	NaN		NaN	208.000	752	
10448191	NaN	NaN		NaN	212.500	752	
10448192	NaN	NaN		NaN	217.500	752	

```
10448193      NaN      NaN          NaN  222.500  752
```

```
[10448194 rows x 17 columns]
```

Reading in SPY daily data.

```
[5]: spydata = pd.read_csv('spy_tickerdata.csv')
spydata['date'] = pd.to_datetime(spydata['date'])
```

And now, we calculate the model-free IV, according to the formula first presented in the introduction:

$$\sum_{i=1}^m [g(T, K_i) + g(T, K_{i-1})] \Delta K$$

Where m is the number of tradable strike prices, $\Delta K = \frac{K_{\max} - K_{\min}}{m}$, $K_i = K_{\min} + i\Delta K$, and $g(T, K_i) = \frac{C^F(T, K_i) - \max(0, F_0 - K_i)}{K_i^2}$.

We also wrote a function `calc_s0`, which calculates the discounted price of the underlying SPY by subtracting the present value of any dividends that will be disbursed before the option expires.

Below we calculate model-free implied volatility for each day in our analysis period. We filter our options that are trading at less than \$0.375, and options that are in the money. We define ‘in-the-money’ as call options with a strike price less than 97% of the spot price of the underlying. We filter out these options because the paper suggested to do this, as their volatilities behave differently than normal for various reasons.

```
[6]: our_ivs, sizes = calculate_iv_for_calls(calls, option_data, tbills, spydata)
```

Next, we cleaned the results and plotted our findings against the Black-Scholes Implied volatility values for the analysis period.

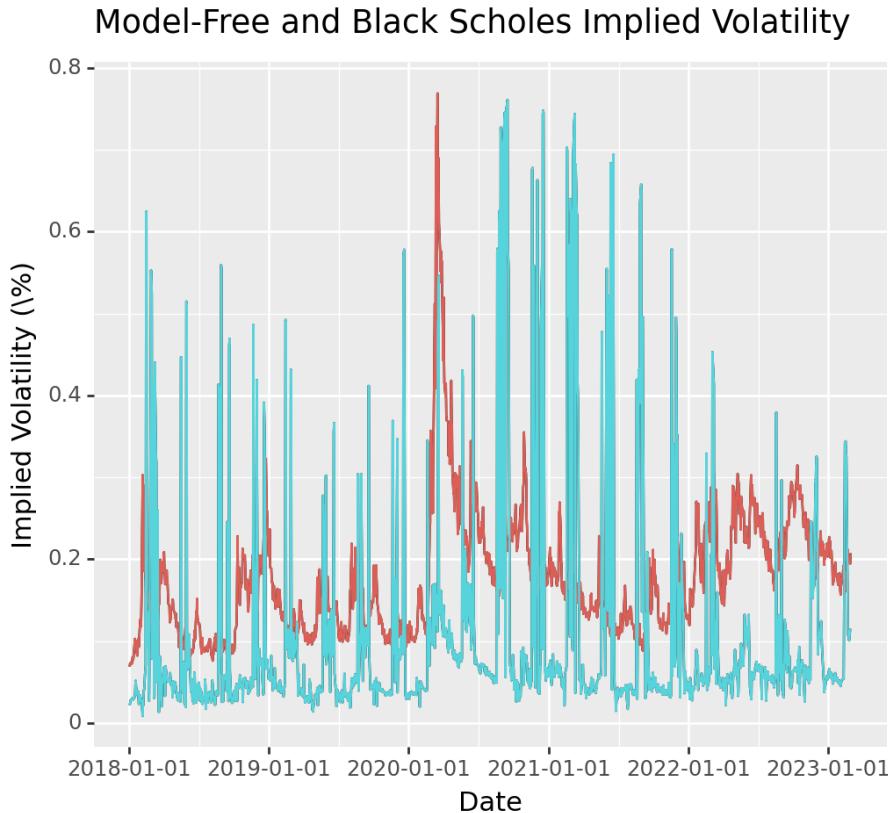
```
[7]: our_ivs = our_ivs.reset_index().rename(columns = {'index':'date'})
our_ivs['BS'] = calls['impl_volatility_x']
our_ivs['iv'] = our_ivs['iv']**0.5
```

```
[8]: plotting_df = our_ivs.melt('date')

plotting_df = plotting_df.rename(columns = {'variable':'type', 'value':'IV'})

mapper = {'iv':'Model-Free IV', 'BS':'BS IV'}
plotting_df['type'] = plotting_df['type'].map(mapper)
```

```
[9]: ggplot(plotting_df, aes(x ='date', y = 'IV', group = 'type')) +
  geom_line() +
  labs(x = 'Date', y = 'Implied Volatility (\%)') +
  ggtitle('Model-Free and Black Scholes Implied Volatility') +
  geom_line(aes(color = 'type'))
```



[9]: <Figure Size: (640 x 480)>

Our hardware is generally lacking, so we save a checkpoint:

```
[10]: our_ivs.to_csv('iv_calculations.csv')
```

1.10 8. STRATEGY SIMULATION

Now we have our IV, we can implement our strategy, which assesses whether we open positions depending on the model-free IV, and sizing positions according to market volume. Note we simulate several additional considerations:

- * SPY dividends
- * Costs of borrowing, with initial capital set to \$1MM and borrowing limit set to \\$9MM
- * Long-short, long only, and short only positions
- * Cash available to trade
- * Costs of shorting
- * Interest from holding cash
- * Costs associated with trading/crossing the book
- * Percentage trading fees

1.10.1 Dividends

We will need dividends for SPY, which will need to be factored into our strategy depending on the amount of SPY we hold at the ex-dividend date. However, as we are keeping track of cash flows, we will not receive the cash until the payment date, which is on the last trading day of the following month for SPY.

```
[3]: spy_divdata, trading_days = process_spy_dividends('spy_tickerdata.csv',  
    ↪'2018-01-01', '2023-02-28')  
spy_divdata
```

```
[3]:      date  dividend  pay_date  
 51  2018-03-16  1.096780 2018-04-30  
114  2018-06-15  1.246000 2018-07-31  
182  2018-09-21  1.323000 2018-10-31  
245  2018-12-21  1.435400 2019-01-31  
301  2019-03-15  1.233100 2019-04-30  
369  2019-06-21  1.431600 2019-07-31  
432  2019-09-20  1.383620 2019-10-31  
496  2019-12-20  1.570000 2020-01-31  
557  2020-03-20  1.405560 2020-04-30  
620  2020-06-19  1.366200 2020-07-31  
683  2020-09-18  1.339200 2020-10-30  
747  2020-12-18  1.580000 2021-01-29  
808  2021-03-19  1.277790 2021-04-30  
871  2021-06-18  1.375870 2021-07-30  
934  2021-09-17  1.428117 2021-10-29  
998  2021-12-17  1.633485 2022-01-31  
1060 2022-03-18  1.366009 2022-04-29  
1123 2022-06-17  1.576871 2022-07-29  
1185 2022-09-16  1.596000 2022-10-31  
1249 2022-12-16  1.781000 2023-01-31
```

1.10.2 Simulations

This loads a new checkpoint to simulate the data from earlier checkpoints. We manipulate the data a bit first.

```
[4]: data_file_path = 'combinedata.csv'  
options_file_path = 'option_df.csv'  
data, options = prepare_dataframes(data_file_path, options_file_path)
```

Let's simulate the ATM options we are going to be trading now:

```
[5]: simulations = create_simulations(options, data, dropna_greeks=True)  
  
[6]: len(trading_days) - len(simulations) # Adds up to contracts with missing greeks  
  
[6]: 158
```

This gets rid of simulations missing trading days (could occur due to various issues, mainly due to bad data):

```
[7]: filtered_simulations = filter_simulations(simulations, trading_days)  
  
[8]: len(trading_days) - len(filtered_simulations)
```

[8]: 325

Now, again showing an example of the simulations.

```
[9]: for key, df in list(filtered_simulations.items())[0:1]:  
    print(f"DataFrame for {key}:")  
    print(df.columns)  
    display(df)  
    print("\n")
```

DataFrame for 2018-01-03:

```
Index(['date', 'exdate', 'strike_price', 'close', 'strikeID', 'cp_flag_c',  
       'best_bid_c', 'best_offer_c', 'impl_volatility_c', 'delta_c',  
       'cp_flag_p', 'best_bid_p', 'best_offer_p', 'impl_volatility_p',  
       'delta_p', 'delta_sum', 'shares_held', 'sharechange'],  
      dtype='object')
```

```
   date      exdate  strike_price  close      strikeID cp_flag_c \\\n0 2018-01-03 2018-02-02      270.0  270.47 20180202_270.0      C  
1 2018-01-04 2018-02-02      270.0  271.61 20180202_270.0      C  
2 2018-01-05 2018-02-02      270.0  273.42 20180202_270.0      C  
3 2018-01-08 2018-02-02      270.0  273.92 20180202_270.0      C  
4 2018-01-09 2018-02-02      270.0  274.54 20180202_270.0      C  
5 2018-01-10 2018-02-02      270.0  274.12 20180202_270.0      C  
6 2018-01-11 2018-02-02      270.0  276.12 20180202_270.0      C  
7 2018-01-12 2018-02-02      270.0  277.92 20180202_270.0      C  
8 2018-01-16 2018-02-02      270.0  276.97 20180202_270.0      C  
9 2018-01-17 2018-02-02      270.0  279.61 20180202_270.0      C  
10 2018-01-18 2018-02-02      270.0  279.14 20180202_270.0      C  
11 2018-01-19 2018-02-02      270.0  280.41 20180202_270.0      C  
12 2018-01-22 2018-02-02      270.0  282.69 20180202_270.0      C  
13 2018-01-23 2018-02-02      270.0  283.29 20180202_270.0      C  
14 2018-01-24 2018-02-02      270.0  283.18 20180202_270.0      C
```

```
   best_bid_c  best_offer_c  impl_volatility_c  delta_c cp_flag_p \\\n0        2.65          2.69         0.072398  0.562218      P  
1        3.45          3.49         0.077848  0.632690      P  
2        4.80          4.84         0.083820  0.727658      P  
3        5.17          5.20         0.089623  0.749240      P  
4        5.67          5.72         0.093639  0.773171      P  
5        5.27          5.33         0.092791  0.759070      P  
6        6.76          6.91         0.090857  0.854846      P  
7        8.45          8.60         0.102281  0.890014      P  
8        7.82          7.97         0.130638  0.827166      P  
9       10.03         10.25         0.131682  0.904279      P  
10       9.81          9.97         0.152184  0.867476      P  
11      10.72         10.90         0.135815  0.927491      P  
12      12.81         12.99         0.144638  0.968565      P  
13      13.63         13.78         0.202072  0.928614      P
```

14	13.48	13.65	0.208464	0.931044	P
best_bid_p best_offer_p impl_volatility_p delta_p delta_sum \					
0	1.77	1.80	0.069568	-0.442522	0.119696
1	1.43	1.46	0.074706	-0.366922	0.265768
2	1.02	1.04	0.081783	-0.270119	0.457539
3	0.81	0.82	0.082307	-0.234696	0.514544
4	0.83	0.84	0.091492	-0.223283	0.549888
5	0.85	0.87	0.089866	-0.235707	0.523363
6	0.56	0.58	0.097319	-0.162272	0.692574
7	0.43	0.44	0.107499	-0.121856	0.768158
8	0.60	0.61	0.121851	-0.157003	0.670163
9	0.38	0.39	0.135023	-0.101488	0.802791
10	0.44	0.45	0.140239	-0.114007	0.753469
11	0.24	0.25	0.135983	-0.072915	0.854576
12	0.15	0.16	0.160805	-0.046985	0.921580
13	0.11	0.12	0.164982	-0.036896	0.891718
14	0.09	0.10	0.166801	-0.032401	0.898643
shares_held sharechange					
0	-0.119696	NaN			
1	-0.265768	-0.146072			
2	-0.457539	-0.191771			
3	-0.514544	-0.057005			
4	-0.549888	-0.035344			
5	-0.523363	0.026525			
6	-0.692574	-0.169211			
7	-0.768158	-0.075584			
8	-0.670163	0.097995			
9	-0.802791	-0.132628			
10	-0.753469	0.049322			
11	-0.854576	-0.101107			
12	-0.921580	-0.067004			
13	-0.891718	0.029862			
14	-0.898643	-0.006925			

1.10.3 PL and Various Metrics

We calculate simulation metrics, adding more concise metrics and altering some columns between daily and cumulative for better mapping onto the single timelines. Here, *Cash - total cost basis + net realized PL* gives how much cash is available to use, i.e. deployable capital. Here we assume that our leverage is completely dependent on how much money we choose to borrow from the \$9MM in our flexible credit facility.

```
[ ]: simulations_long = {date: calculate_realized_PL(df.copy(), long_op=True) for date, df in filtered_simulations.items()}
simulations_short = {date: calculate_realized_PL(df.copy(), long_op=False) for date, df in filtered_simulations.items()}
```

Let's take a look:

```
[35]: for key, df in list(simulations_short.items())[-5:]:
    print(f"DataFrame for {key}:")
    print(df.columns)
    display(df)
    print("\n")
```

DataFrame for 2023-02-22:

```
Index(['date', 'exdate', 'strike_price', 'close', 'strikeID', 'cp_flag_c',
       'best_bid_c', 'best_offer_c', 'impl_volatility_c', 'delta_c',
       'cp_flag_p', 'best_bid_p', 'best_offer_p', 'impl_volatility_p',
       'delta_p', 'delta_sum', 'stock_pos', 'pos_change', 'change_cost_basis',
       'stock_cost_basis', 'daily_stock_value', 'stock_PL',
       'option_cost_basis', 'change_cost_basis_op', 'daily_option_value',
       'option_PL', 'total_cost_basis', 'total_pos_value', 'total_PL',
       'realized_stock_PL', 'realized_option_PL', 'realized_PL', 'UID',
       'to_open'],
      dtype='object')

   date      exdate  strike_price     close      strikeID cp_flag_c \
0 2023-02-22 2023-03-24        399.0  398.54  20230324_399.0      C
1 2023-02-23 2023-03-24        399.0  400.66  20230324_399.0      C
2 2023-02-24 2023-03-24        399.0  396.38  20230324_399.0      C
3 2023-02-27 2023-03-24        399.0  397.73  20230324_399.0      C
4 2023-02-28 2023-03-24        399.0  396.26  20230324_399.0      C

   best_bid_c  best_offer_c  impl_volatility_c  delta_c ... \
0      9.08          9.12         0.202075  0.517410 ...
1      9.99         10.03         0.200472  0.555948 ...
2      7.37          7.41         0.195788  0.473339 ...
3      7.54          7.58         0.197377  0.495326 ...
4      7.03          7.05         0.207028  0.465189 ...

   daily_option_value  option_PL  total_cost_basis  total_pos_value  total_PL \
0           -18.83     -0.08        -14.284359      -14.364359 -0.080000
1           -18.06      0.69         15.854488       16.568242  0.713755
2           -17.62      1.13        -49.236657      -48.452815  0.783843
3           -16.65      2.10        -32.535975      -30.887143  1.648832
4            0.00      0.00         0.000000       0.000000  0.000000

   realized_stock_PL  realized_option_PL  realized_PL \
0            0.000000          0.00      0.000000
1            0.000000          0.00      0.000000
```

```

2      0.000000      0.00      0.000000
3      0.000000      0.00      0.000000
4     -0.398548      2.83     2.431452

```

```

          UID  to_open
0 20230324_399.0_2023-02-22      1
1 20230324_399.0_2023-02-22      0
2 20230324_399.0_2023-02-22      0
3 20230324_399.0_2023-02-22      0
4 20230324_399.0_2023-02-22      0

```

[5 rows x 34 columns]

DataFrame for 2023-02-23:

```

Index(['date', 'exdate', 'strike_price', 'close', 'strikeID', 'cp_flag_c',
       'best_bid_c', 'best_offer_c', 'impl_volatility_c', 'delta_c',
       'cp_flag_p', 'best_bid_p', 'best_offer_p', 'impl_volatility_p',
       'delta_p', 'delta_sum', 'stock_pos', 'pos_change', 'change_cost_basis',
       'stock_cost_basis', 'daily_stock_value', 'stock_PL',
       'option_cost_basis', 'change_cost_basis_op', 'daily_option_value',
       'option_PL', 'total_cost_basis', 'total_pos_value', 'total_PL',
       'realized_stock_PL', 'realized_option_PL', 'realized_PL', 'UID',
       'to_open'],
      dtype='object')

      date      exdate    strike_price    close      strikeID cp_flag_c \
0 2023-02-23 2023-03-24        401.0  400.66  20230324_401.0      C
1 2023-02-24 2023-03-24        401.0  396.38  20230324_401.0      C
2 2023-02-27 2023-03-24        401.0  397.73  20230324_401.0      C
3 2023-02-28 2023-03-24        401.0  396.26  20230324_401.0      C

      best_bid_c  best_offer_c  impl_volatility_c  delta_c ... \
0         8.79        8.82        0.196720  0.518997 ...
1         6.35        6.39        0.192120  0.433775 ...
2         6.48        6.51        0.193449  0.453692 ...
3         5.99        6.01        0.202116  0.424099 ...

      daily_option_value  option_PL  total_cost_basis  total_pos_value  total_PL \
0            -17.68      -0.06        -13.019221      -13.079221 -0.060000
1            -17.61       0.01        -79.673293      -79.712440 -0.039147
2            -16.55       1.07        -64.680463      -63.871120  0.809343
3             0.00       0.00         0.000000       0.000000  0.000000

      realized_stock_PL  realized_option_PL  realized_PL \
0            0.000000        0.00        0.000000
1            0.000000        0.00        0.000000
2            0.000000        0.00        0.000000

```

```
3           -0.085759          1.74      1.654241
```

```
          UID  to_open
0  20230324_401.0_2023-02-23      1
1  20230324_401.0_2023-02-23      0
2  20230324_401.0_2023-02-23      0
3  20230324_401.0_2023-02-23      0
```

```
[4 rows x 34 columns]
```

```
DataFrame for 2023-02-24:
```

```
Index(['date', 'exdate', 'strike_price', 'close', 'strikeID', 'cp_flag_c',
       'best_bid_c', 'best_offer_c', 'impl_volatility_c', 'delta_c',
       'cp_flag_p', 'best_bid_p', 'best_offer_p', 'impl_volatility_p',
       'delta_p', 'delta_sum', 'stock_pos', 'pos_change', 'change_cost_basis',
       'stock_cost_basis', 'daily_stock_value', 'stock_PL',
       'option_cost_basis', 'change_cost_basis_op', 'daily_option_value',
       'option_PL', 'total_cost_basis', 'total_pos_value', 'total_PL',
       'realized_stock_PL', 'realized_option_PL', 'realized_PL', 'UID',
       'to_open'],
      dtype='object')
```

```
      date      exdate    strike_price    close      strikeID cp_flag_c \
0 2023-02-24 2023-03-24        396.0  396.38  20230324_396.0      C
1 2023-02-27 2023-03-24        396.0  397.73  20230324_396.0      C
2 2023-02-28 2023-03-24        396.0  396.26  20230324_396.0      C
```

```
      best_bid_c  best_offer_c  impl_volatility_c  delta_c ... \
0          9.05          9.08          0.201088  0.531177 ...
1          9.27          9.31          0.202705  0.555327 ...
2          8.74          8.77          0.214299  0.524633 ...
```

```
      daily_option_value  option_PL  total_cost_basis  total_pos_value  total_PL \
0            -17.92      -0.06          -3.142807      -3.202807 -0.060000
1            -17.07      0.79          15.532207      16.372332  0.840124
2             0.00      0.00          0.000000      0.000000  0.000000
```

```
      realized_stock_PL  realized_option_PL  realized_PL \
0            0.000000          0.00      0.000000
1            0.000000          0.00      0.000000
2           -0.073478          1.56      1.486522
```

```
          UID  to_open
0  20230324_396.0_2023-02-24      1
1  20230324_396.0_2023-02-24      0
2  20230324_396.0_2023-02-24      0
```

[3 rows x 34 columns]

DataFrame for 2023-02-27:

```
Index(['date', 'exdate', 'strike_price', 'close', 'strikeID', 'cp_flag_c',
       'best_bid_c', 'best_offer_c', 'impl_volatility_c', 'delta_c',
       'cp_flag_p', 'best_bid_p', 'best_offer_p', 'impl_volatility_p',
       'delta_p', 'delta_sum', 'stock_pos', 'pos_change', 'change_cost_basis',
       'stock_cost_basis', 'daily_stock_value', 'stock_PL',
       'option_cost_basis', 'change_cost_basis_op', 'daily_option_value',
       'option_PL', 'total_cost_basis', 'total_pos_value', 'total_PL',
       'realized_stock_PL', 'realized_option_PL', 'realized_PL', 'UID',
       'to_open'],
      dtype='object')

  date      exdate  strike_price    close      strikeID cp_flag_c \
0 2023-02-27 2023-03-31        398.0  397.73  20230331_398.0      C
1 2023-02-28 2023-03-31        398.0  396.26  20230331_398.0      C

  best_bid_c  best_offer_c  impl_volatility_c   delta_c ... \
0      8.92          8.96         0.194675  0.511410 ...
1      8.39          8.42         0.202880  0.483923 ...

  daily_option_value  option_PL  total_cost_basis  total_pos_value  total_PL \
0        -18.22      -0.07        -15.700381      -15.770381     -0.07
1         0.00        0.00         0.000000        0.000000      0.00

  realized_stock_PL  realized_option_PL  realized_PL \
0        0.000000          0.0        0.000000
1      -0.009054          0.7        0.690946

  UID  to_open
0 20230331_398.0_2023-02-27      1
1 20230331_398.0_2023-02-27      0
```

[2 rows x 34 columns]

DataFrame for 2023-02-28:

```
Index(['date', 'exdate', 'strike_price', 'close', 'strikeID', 'cp_flag_c',
       'best_bid_c', 'best_offer_c', 'impl_volatility_c', 'delta_c',
       'cp_flag_p', 'best_bid_p', 'best_offer_p', 'impl_volatility_p',
       'delta_p', 'delta_sum', 'stock_pos', 'pos_change', 'change_cost_basis',
       'stock_cost_basis', 'daily_stock_value', 'stock_PL',
       'option_cost_basis', 'change_cost_basis_op', 'daily_option_value',
       'option_PL', 'total_cost_basis', 'total_pos_value', 'total_PL',
       'realized_stock_PL', 'realized_option_PL', 'realized_PL', 'UID',
```

```

    'to_open'],
    dtype='object')

      date      exdate  strike_price   close       strikeID cp_flag_c \
0 2023-02-28 2023-03-31        396.0  396.26  20230331_396.0          C

      best_bid_c  best_offer_c  impl_volatility_c   delta_c ... \
0         9.55           9.58          0.207305  0.519384 ...

      daily_option_value  option_PL  total_cost_basis  total_pos_value  total_PL \
0            0.0           0.0            0.0            0.0            0.0

      realized_stock_PL  realized_option_PL  realized_PL \
0            0.0           0.0            0.0

      UID  to_open
0 20230331_396.0_2023-02-28           1

[1 rows x 34 columns]

```

1.10.4 IV data

Load calculated IV data.

```
[36]: iv_data = pd.read_csv('iv_calculations.csv')
iv_data['iv'] = iv_data['iv']
iv_data
```

```
[36]:      Unnamed: 0      date      iv      BS
0            0 2018-01-02  0.021847  0.068871
1            1 2018-01-03  0.023454  0.072398
2            2 2018-01-04  0.027852  0.071179
3            3 2018-01-05  0.027788  0.074068
4            4 2018-01-08  0.029469  0.072545
...
1293        1293 2023-02-22  0.107065  0.202075
1294        1294 2023-02-23  0.111236  0.196720
1295        1295 2023-02-24  0.100687  0.201088
1296        1296 2023-02-27  0.114114  0.194675
1297        1297 2023-02-28  0.112473  0.207305
```

[1298 rows x 4 columns]

Create the IV comparisons to trade, depending on whether model-free IV is more or less than the IV we will go long or short gamma. We also simulate going only long or only short gamma.

```
[37]: IV_compare = compare_iv(filtered_simulations, iv_data)
IV_compare
```

```
[37]:      date  BS_Call_IV  MF_Call_IV   IV_diff
 0  2018-01-03    0.072398    0.023454 -0.048944
 1  2018-01-04    0.071179    0.027852 -0.043327
 2  2018-01-10    0.080641    0.030803 -0.049838
 3  2018-01-11    0.076890    0.028318 -0.048572
 4  2018-01-12    0.079350    0.029221 -0.050129
 ..
 ..
 968 2023-02-22    0.202075    0.107065 -0.095010
 969 2023-02-23    0.196720    0.111236 -0.085484
 970 2023-02-24    0.201088    0.100687 -0.100401
 971 2023-02-27    0.194675    0.114114 -0.080561
 972 2023-02-28    0.207305    0.112473 -0.094832
```

[973 rows x 4 columns]

Quick Look at IV Taking a look at the differences in IV, based on quantile:

```
[38]: (IV_compare['IV_diff'] > 0).sum()
```

```
[38]: 124
```

```
[39]: IV_compare[IV_compare['IV_diff'] > 0]['IV_diff'].quantile(0.5)
```

```
[39]: 0.31982084842669284
```

```
[41]: IV_compare[IV_compare['IV_diff'] > 0]['IV_diff'].quantile(0.25)
```

```
[41]: 0.16191266523189107
```

```
[42]: IV_compare[IV_compare['IV_diff'] > 0]['IV_diff'].quantile(0.2)
```

```
[42]: 0.12497667184676832
```

```
[43]: IV_compare[IV_compare['IV_diff'] < 0]['IV_diff'].quantile(0.75)
```

```
[43]: -0.0785731074996423
```

```
[44]: IV_compare[IV_compare['IV_diff'] < 0]['IV_diff'].quantile(0.80)
```

```
[44]: -0.07246443270542066
```

```
[45]: IV_compare[IV_compare['IV_diff'] < 0]['IV_diff'].quantile(0.70)
```

```
[45]: -0.08366817795015688
```

```
[46]: IV_compare[IV_compare['IV_diff'] < 0]['IV_diff'].quantile(0.50)
```

```
[46]: -0.11270668550109746
```

```
[47]: IV_compare['IV_diff'].max()
```

```
[47]: 0.5826495838418329
```

```
[48]: IV_compare['IV_diff'].min()
```

```
[48]: -0.6570148924085712
```

1.10.5 Strategies

Essentially, we set a threshold for which we will go long and short gamma, which we use to evaluate on each day whether there is a trading opportunity. We set these thresholds based on the quantiles we looked at above.

We simulate three strategies:

- * 1: Long-Short - if the model-free IV is greater than the IV by a set threshold, we will take a long gamma position. If it is less than the IV by another set threshold, we will take a short gamma position.
- * 2: Long only - we only take long gamma positions, dependent on a threshold.
- * 3: Short only - we only take short gamma positions, opening positions depending on a threshold.

```
[ ]: strat_dict = {'trade_1': trade_strategy_1, 'trade_2': trade_strategy_2,  
    ↪'trade_3': trade_strategy_3}
```

```
[50]: for key, func in strat_dict.items():  
    IV_compare[key] = IV_compare['IV_diff'].apply(func)  
IV_compare
```

```
[50]:      date  BS_Call_IV  MF_Call_IV  IV_diff  trade_1  trade_2  trade_3  
0  2018-01-03  0.072398  0.023454 -0.048944  0  0  0  
1  2018-01-04  0.071179  0.027852 -0.043327  0  0  0  
2  2018-01-10  0.080641  0.030803 -0.049838  0  0  0  
3  2018-01-11  0.076890  0.028318 -0.048572  0  0  0  
4  2018-01-12  0.079350  0.029221 -0.050129  0  0  0  
..  ...  ...  ...  ...  ...  ...  
968 2023-02-22  0.202075  0.107065 -0.095010  0  0 -1  
969 2023-02-23  0.196720  0.111236 -0.085484  0  0 -1  
970 2023-02-24  0.201088  0.100687 -0.100401 -1  0 -1  
971 2023-02-27  0.194675  0.114114 -0.080561  0  0 -1  
972 2023-02-28  0.207305  0.112473 -0.094832  0  0 -1
```

[973 rows x 7 columns]

```
[51]: IV_compare['date'] = IV_compare['date'].dt.strftime('%Y-%m-%d')
```

```
[52]: abs(IV_compare['trade_2']).sum()
```

[52]: 53

Here, generate the trade dataframes, by mapping each individual position onto the timeline based on whether we opened the position long or short (either 1 or -1, with additional position sizing added later) or did nothing (0).

```
[54]: trades_dfs = generate_trades_dfs(strat_dict, IV_compare, simulations_long, simulations_short)
volumes = preprocess_options(options)
```

Looking at volumes:

```
[55]: volumes.loc[volumes['volume_med'] > 0, 'volume_med'].quantile(0.10)
```

[55]: 49.0

```
[56]: (volumes['volume_c'].quantile(.1) + volumes['volume_p'].quantile(.1))/2
```

[56]: 28.5

We assume an initial capital, from which we can get additional leverage, on which we will also pay costs and fees. We also assume that when we borrow, we will get a rate 25 bps above the risk-free rate.

```
[57]: # 10 million dollars: 1 million initial capital, up to 9 million as a lending facility
KAPITAL = 1e7
INITIAL = 1e6
LEVERAGE = KAPITAL - INITIAL
```

Due to low liquidity on some days in the month-out ATM options, we instead assume that the market can handle options volume at around the 10th percentile of typical volume, which determines our position sizing on those low-liquidity days. Otherwise, we determine position sizing based on the attractiveness of the trade (absolute value of IV differential), capped at a maximum 80% differential as a risk control, in addition to not allowing a trade to exceed 10% of our capital as a capital remains bounded.

```
[59]: trades_dfs = update_trades_with_pos_size(trades_dfs, volumes, KAPITAL)
```

Let's take a look at our trades after sizing them accordingly:

```
[60]: trades_dfs['trade_1'].tail(6)
```

```
[60]:      date    exdate  strike_price   close       strikeID delta_sum \
8730 2023-02-24 2023-03-24      399.0  396.38  20230324_399.0 -0.077786
8731 2023-02-27 2023-03-10      410.0  397.73  20230310_410.0 -0.688320
8732 2023-02-27 2023-03-24      396.0  397.73  20230324_396.0  0.084083
8733 2023-02-27 2023-03-24      399.0  397.73  20230324_399.0 -0.035796
8734 2023-02-28 2023-03-24      396.0  396.26  20230324_396.0  0.020843
8735 2023-02-28 2023-03-24      399.0  396.26  20230324_399.0 -0.108683
```

	stock_pos	pos_change	change_cost_basis	stock_cost_basis	...	\
8730	-0.077786	-0.164214	-65.091145	-30.468441	...	
8731	0.000000	0.688026	273.648581	0.000000	...	
8732	0.084083	0.046954	18.675014	33.392207	...	
8733	-0.035796	0.041990	16.700683	-13.767759	...	
8734	0.000000	-0.084083	-33.318730	0.000000	...	
8735	0.000000	0.035796	14.184523	0.000000	...	
	sized_option_cost_basis	sized_change_cost_basis_op	\			
8730	-21681.0		0.0			
8731	0.0		21000.0			
8732	-19646.0		0.0			
8733	-21681.0		0.0			
8734	0.0		17930.0			
8735	0.0		17512.0			
	sized_daily_option_value	sized_option_PL	sized_total_cost_basis	\		
8730	-19382.0	2299.0	-55196.285584			
8731	0.0	0.0	0.000000			
8732	-18777.0	869.0	17085.428184			
8733	-18315.0	3366.0	-36825.534614			
8734	0.0	0.0	0.000000			
8735	0.0	0.0	0.000000			
	sized_total_pos_value	sized_total_PL	sized_realized_stock_PL	\		
8730	-53298.096148	1898.189436	0.000000			
8731	0.000000	0.000000	-2223.604995			
8732	18009.564749	924.136565	0.000000			
8733	-33975.857388	2849.677226	0.000000			
8734	0.000000	0.000000	-80.825646			
8735	0.000000	0.000000	-458.440642			
	sized_realized_option_PL	sized_realized_PL				
8730	0.0	0.000000				
8731	4680.0	2456.395005				
8732	0.0	0.000000				
8733	0.0	0.000000				
8734	1716.0	1635.174354				
8735	4169.0	3710.559358				

[6 rows x 43 columns]

1.10.6 Dataframe Metrics

We calculate the PL metrics and costs for each dataframe's trades, minimizing trading transactions to get to the same point wherever possible. To do so, we consolidate the underlying stock position while treating the options as separate securities.

```
[61]: PL_temp_dfs = summarize_pl_by_date(trades_dfs, trading_days)
PL_temp_dfs['trade_1'][50:84]
```

	date	sized_stock_pos	sized_change_cost_basis	\	
50	2018-03-15	-4905.3341	-1.995678e+05		
51	2018-03-16	-5069.7366	-4.507917e+04		
52	2018-03-19	2211.3897	1.969472e+06		
53	2018-03-20	-984.7594	-8.659966e+05		
54	2018-03-21	-1481.1900	-1.342497e+05		
55	2018-03-22	27.8096	3.978779e+05		
56	2018-03-23	-497.3449	-1.355161e+05		
57	2018-03-26	662.8197	3.075712e+05		
58	2018-03-27	-437.0603	-2.866287e+05		
59	2018-03-28	-684.9435	-6.440749e+04		
60	2018-03-29	1013.4659	4.469364e+05		
61	2018-04-02	-2161.4125	-8.174359e+05		
62	2018-04-03	148.3848	6.023258e+05		
63	2018-04-04	2361.1328	5.831919e+05		
64	2018-04-05	4496.0433	5.671176e+05		
65	2018-04-06	-1933.1882	-1.669800e+06		
66	2018-04-09	-207.4741	4.504114e+05		
67	2018-04-10	5231.9108	1.442253e+06		
68	2018-04-11	3458.1284	-4.678528e+05		
69	2018-04-12	6930.2940	9.233530e+05		
70	2018-04-13	5479.8851	-3.845759e+05		
71	2018-04-16	8871.3407	9.066378e+05		
72	2018-04-17	10728.5716	5.018052e+05		
73	2018-04-18	9717.9262	-2.732684e+05		
74	2018-04-19	7435.5754	-6.137013e+05		
75	2018-04-20	5223.2715	-5.898223e+05		
76	2018-04-23	4220.6426	-2.672708e+05		
77	2018-04-24	-467.6853	-1.232936e+06		
78	2018-04-25	317.1996	2.069192e+05		
79	2018-04-26	2751.5291	6.482863e+05		
80	2018-04-27	2301.8278	-1.198724e+05		
81	2018-04-30	-239.9211	-6.723180e+05		
82	2018-05-01	113.5991	9.367578e+04		
83	2018-05-02	-982.3976	-2.884663e+05		
		sized_stock_cost_basis	sized_daily_stock_value	sized_stock_PL	\
50		-1.378472e+06	-1.348967e+06	29504.771606	
51		-1.423551e+06	-1.390122e+06	33429.038886	
52		5.459210e+05	5.981588e+05	52237.761672	
53		-2.992710e+05	-2.668206e+05	32450.461897	
54		-4.127809e+05	-4.005582e+05	12222.726100	
55		7.332557e+03	7.332557e+03	0.000000	
56		-1.281836e+05	-1.283399e+05	-156.289952	

57	1.793877e+05	1.757201e+05	-3667.544946
58	-1.072411e+05	-1.138979e+05	-6656.861793
59	-1.716485e+05	-1.779689e+05	-6320.325362
60	2.752879e+05	2.666936e+05	-8594.337782
61	-5.421481e+05	-5.564989e+05	-14350.824094
62	6.017779e+04	3.869430e+04	-21483.485344
63	6.433697e+05	6.223002e+05	-21069.491752
64	1.210487e+06	1.194329e+06	-16158.335528
65	-4.593127e+05	-5.020876e+05	-42774.911864
66	-8.901347e+03	-5.415074e+04	-45249.392760
67	1.433352e+06	1.387241e+06	-46110.410275
68	9.654987e+05	9.121159e+05	-53382.766287
69	1.878820e+06	1.842973e+06	-35847.403339
70	1.487542e+06	1.452992e+06	-34550.641821
71	2.384537e+06	2.371576e+06	-12961.411492
72	2.886392e+06	2.898753e+06	12360.606702
73	2.613378e+06	2.627630e+06	14252.045613
74	1.995010e+06	1.999352e+06	4341.476402
75	1.405188e+06	1.392576e+06	-12611.635510
76	1.139984e+06	1.125097e+06	-14887.627570
77	-9.833869e+04	-1.229919e+05	-24653.185979
78	1.015270e+05	8.362333e+04	-17903.698298
79	7.425357e+05	7.327597e+05	-9775.994516
80	6.219180e+05	6.135752e+05	-8342.783195
81	-5.313651e+04	-6.346153e+04	-10325.021220
82	3.693444e+04	3.010149e+04	-6832.954647
83	-2.550280e+05	-2.585670e+05	-3539.017475

50	sized_option_cost_basis	sized_change_cost_basis_op	\
51	157363.0	11052.0	
52	157363.0	0.0	
53	157363.0	0.0	
54	102036.0	-30927.0	
55	48700.0	-26963.0	
56	-14053.0	-46353.0	
57	-29120.0	-15067.0	
58	-48481.0	-19361.0	
59	-69343.0	-20862.0	
60	-88638.0	-19295.0	
61	-105995.0	-17357.0	
62	-124490.0	-18495.0	
63	-141035.0	-16545.0	
64	-161023.0	-19988.0	
65	-178699.0	-17676.0	
66	-193735.0	-15036.0	
67	-210820.0	-17085.0	
	-226330.0	-15510.0	

68	-241855.0	-15525.0	
69	-241992.0	-6702.0	
70	-226925.0	9425.0	
71	-207564.0	9975.0	
72	-186702.0	21071.0	
73	-167407.0	19278.0	
74	-150050.0	12512.0	
75	-150050.0	0.0	
76	-131555.0	18495.0	
77	-115010.0	9465.0	
78	-108168.0	-1993.0	
79	-90492.0	8802.0	
80	-88699.0	-2057.0	
81	-71614.0	10170.0	
82	-56104.0	6420.0	
83	-40579.0	7200.0	
	sized_daily_option_value	sized_option_PL	sized_total_cost_basis \
50	104763.0	-52600.0	-1.221109e+06
51	100406.0	-56957.0	-1.266188e+06
52	98938.0	-58425.0	7.032840e+05
53	61341.0	-40695.0	-1.972350e+05
54	29150.0	-19550.0	-3.640809e+05
55	-14261.0	-208.0	-6.720443e+03
56	-30420.0	-1300.0	-1.573036e+05
57	-48179.0	302.0	1.309067e+05
58	-69539.0	-196.0	-1.765841e+05
59	-89346.0	-708.0	-2.602865e+05
60	-102344.0	3651.0	1.692929e+05
61	-126734.0	-2244.0	-6.666381e+05
62	-134427.0	6608.0	-8.085721e+04
63	-155875.0	5148.0	4.823467e+05
64	-172010.0	6689.0	1.031788e+06
65	-176793.0	16942.0	-6.530477e+05
66	-194893.0	15927.0	-2.197213e+05
67	-210457.0	15873.0	1.207022e+06
68	-208101.0	33754.0	7.236437e+05
69	-213918.0	28074.0	1.636828e+06
70	-181459.0	45466.0	1.260617e+06
71	-173373.0	34191.0	2.176973e+06
72	-183979.0	2723.0	2.699690e+06
73	-163757.0	3650.0	2.445971e+06
74	-134698.0	15352.0	1.844960e+06
75	-119254.0	30796.0	1.255138e+06
76	-92942.0	38613.0	1.008429e+06
77	-76894.0	38116.0	-2.133487e+05
78	-77865.0	30303.0	-6.640971e+03

79	-66818.0	23674.0	6.520437e+05
80	-64705.0	23994.0	5.332190e+05
81	-47017.0	24597.0	-1.247505e+05
82	-37783.0	18321.0	-1.916956e+04
83	-30180.0	10399.0	-2.956070e+05
	sized_total_pos_value	sized_total_PL	sized_realized_stock_PL \
50	-1.244204e+06	-23095.228394	-5556.721428
51	-1.289716e+06	-23527.961114	0.000000
52	6.970968e+05	-6187.238328	0.000000
53	-2.054796e+05	-8244.538103	20804.539037
54	-3.714082e+05	-7327.273900	20739.810685
55	-6.928443e+03	-208.000000	22235.570500
56	-1.587599e+05	-1456.289952	0.000000
57	1.275411e+05	-3365.544946	0.000000
58	-1.834369e+05	-6852.861793	0.000000
59	-2.673149e+05	-7028.325362	0.000000
60	1.643496e+05	-4943.337782	0.000000
61	-6.832329e+05	-16594.824094	0.000000
62	-9.573270e+04	-14875.485344	0.000000
63	4.664252e+05	-15921.491752	0.000000
64	1.022319e+06	-9469.335528	0.000000
65	-6.788806e+05	-25832.911864	0.000000
66	-2.490437e+05	-29322.392760	0.000000
67	1.176784e+06	-30237.410275	0.000000
68	7.040149e+05	-19628.766287	0.000000
69	1.629055e+06	-7773.403339	-10031.224320
70	1.271533e+06	10915.358179	-6702.390838
71	2.198203e+06	21229.588508	-9643.080811
72	2.714774e+06	15083.606702	50.016208
73	2.463873e+06	17902.045613	254.275409
74	1.864654e+06	19693.476402	-4666.320089
75	1.273322e+06	18184.364490	0.000000
76	1.032155e+06	23725.372430	2067.061200
77	-1.998859e+05	13462.814021	-5386.548525
78	5.758331e+03	12399.301702	-7053.483126
79	6.659417e+05	13898.005484	-7277.608854
80	5.488702e+05	15651.216805	-745.329046
81	-1.104785e+05	14271.978780	-2736.508965
82	-7.681510e+03	11488.045353	-3604.829490
83	-2.887470e+05	6859.982525	-3496.143570
	sized_realized_option_PL	sized_realized_PL	
50	5130.0	-426.721428	
51	0.0	0.000000	
52	0.0	0.000000	
53	-24400.0	-3595.460963	

```

54           -26373.0      -5633.189315
55           -16400.0       5835.570500
56             0.0        0.000000
57             0.0        0.000000
58             0.0        0.000000
59             0.0        0.000000
60             0.0        0.000000
61             0.0        0.000000
62             0.0        0.000000
63             0.0        0.000000
64             0.0        0.000000
65             0.0        0.000000
66             0.0        0.000000
67             0.0        0.000000
68             0.0        0.000000
69            6565.0      -3466.224320
70            5642.0      -1060.390838
71            9386.0      -257.080811
72           -209.0      -158.983792
73            17.0       271.275409
74            4845.0      178.679911
75             0.0        0.000000
76             0.0       2067.061200
77            7080.0      1693.451475
78            8835.0      1781.516874
79            8874.0      1596.391146
80            3850.0      3104.670954
81            6915.0      4178.491035
82            9090.0      5485.170510
83            8325.0      4828.856430

```

Apply dividends:

```
[64]: divvies = calculate_dividends(PL_temp_dfs, spy_divdata)
divvies['trade_1'].head(50)
```

```
[64]:      date   dividend   pay_date   sized_stock_pos      div
 0  2018-03-16    1.096780  2018-04-30      -5069.7366  -5560.385708
 1  2018-06-15    1.246000  2018-07-31        0.0000     0.000000
 2  2018-09-21    1.323000  2018-10-31     -1631.3276  -2158.246415
 3  2018-12-21    1.435400  2019-01-31     -1342.4627  -1926.970960
 4  2019-03-15    1.233100  2019-04-30     -3734.6172  -4605.156469
 5  2019-06-21    1.431600  2019-07-31      659.6096    944.297103
 6  2019-09-20    1.383620  2019-10-31        0.0000     0.000000
 7  2019-12-20    1.570000  2020-01-31     -1650.5574  -2591.375118
 8  2020-03-20    1.405560  2020-04-30     -1472.6726  -2069.929700
 9  2020-06-19    1.366200  2020-07-31      822.8804    1124.219202
10 2020-09-18    1.339200  2020-10-30    12863.3701   17226.625238
```

```

11 2020-12-18 1.580000 2021-01-29      -5687.4344 -8986.146352
12 2021-03-19 1.277790 2021-04-30      2314.1840  2957.041173
13 2021-06-18 1.375870 2021-07-30      16110.8271 22166.403682
14 2021-09-17 1.428117 2021-10-29      0.0000    0.000000
15 2021-12-17 1.633485 2022-01-31      494.4044   807.602171
16 2022-03-18 1.366009 2022-04-29      1542.7890  2107.463659
17 2022-06-17 1.576871 2022-07-29      -2586.1145 -4077.968958
18 2022-09-16 1.596000 2022-10-31      -2510.7063 -4007.087255
19 2022-12-16 1.781000 2023-01-31      -2067.5956 -3682.387764

```

```
[65]: PL_temp_dfs = merge_dividends_with_pl(PL_temp_dfs, divvies)
PL_temp_dfs['trade_1'][-2:]
```

```

[65]:          date  sized_stock_pos  sized_change_cost_basis \
1296 2023-02-27           53.1157        449386.138302
1297 2023-02-28           0.0000       -21047.627282

          sized_stock_cost_basis  sized_daily_stock_value  sized_stock_PL \
1296            21586.89357        21125.707361     -461.186209
1297            0.00000          0.000000      0.000000

          sized_option_cost_basis  sized_change_cost_basis_op \
1296            -41327.0          21000.0
1297             0.0            35442.0

          sized_daily_option_value  sized_option_PL  sized_total_cost_basis \
1296            -37092.0          4235.0      -19740.10643
1297             0.0            0.0          0.00000

          sized_total_pos_value  sized_total_PL  sized_realized_stock_PL \
1296            -15966.292639      3773.813791     -2223.604995
1297            0.000000        0.000000      -539.266288

          sized_realized_option_PL  sized_realized_PL  div
1296                 4680.0        2456.395005  0.0
1297                 5885.0        5345.733712  0.0

```

We use smallest DTE to approximate an overnight risk-free rate.

```
[66]: tbills_data = process_tbills_data('tbill_data.csv', '2018-01-01', '2023-02-28', ↴
                                         trading_days)
tbills_data
```

```

[66]:          date  dte maturity_date      price      rate  leverage_rate
0    2018-01-02  2.0    2018-01-04  99.994083  0.000030    0.000036
1    2018-01-03  1.0    2018-01-04  99.996944  0.000031    0.000037
2    2018-01-04  7.0    2018-01-11  99.976375  0.000034    0.000041
3    2018-01-05  6.0    2018-01-11  99.979583  0.000034    0.000041

```

```

4    2018-01-08  3.0    2018-01-11  99.989750  0.000034    0.000041
...
1283  ...  ...    2023-02-23  99.987410  0.000126    0.000132
1284  2023-02-23  5.0    2023-02-28  99.935972  0.000128    0.000135
1285  2023-02-24  4.0    2023-02-28  99.950389  0.000124    0.000131
1286  2023-02-27  1.0    2023-02-28  99.987639  0.000124    0.000130
1287  2023-02-28  2.0    2023-03-02  99.975833  0.000121    0.000127

```

[1288 rows x 6 columns]

Calculate the rate:

```
[67]: rfr = calculate_rfr(trading_days, tbills_data)
rfr
```

```
[67]:      date      rate  leverage_rate
0    2018-01-02  0.000030      0.000036
1    2018-01-03  0.000031      0.000037
2    2018-01-04  0.000034      0.000041
3    2018-01-05  0.000034      0.000041
4    2018-01-08  0.000034      0.000041
...
1293  ...  ...      ...
1294  2023-02-22  0.000126      0.000132
1294  2023-02-23  0.000128      0.000135
1295  2023-02-24  0.000124      0.000131
1296  2023-02-27  0.000124      0.000130
1297  2023-02-28  0.000121      0.000127
```

[1298 rows x 3 columns]

1.10.7 Final Dataframes

In this simulation, when we are short shares, we do not need to borrow money, hence we don't go above initial capital of 1 million, which is why lever cash is 0. To account for the fees associated with opening a short position, we pay to borrow the shares under short fees (hence not double-paying fees).

Here, we take the simulations for each strategy, and calculate all of the associated metrics.

```
[68]: PL_dfs = process_pl_dfs(PL_temp_dfs, rfr, INITIAL, KAPITAL)
```

```
[69]: PL_dfs['trade_1']
```

```
[69]:      date  gross_stock_trades  gross_option_trades  gross_trades_value \
0    2018-01-02        0.000000            0.0          0.000000
1    2018-01-03        0.000000            0.0          0.000000
2    2018-01-04        0.000000            0.0          0.000000
3    2018-01-05        0.000000            0.0          0.000000
4    2018-01-08        0.000000            0.0          0.000000
```

...
1293	2023-02-22	364328.728590	20355.0	384683.728590	
1294	2023-02-23	62357.239958	0.0	62357.239958	
1295	2023-02-24	174798.030680	19646.0	194444.030680	
1296	2023-02-27	449386.138302	21000.0	470386.138302	
1297	2023-02-28	21047.627282	35442.0	56489.627282	
		stock_trading_costs	option_trading_costs	net_trading_costs	\
0		0.000000	0.0000	0.000000	
1		0.000000	0.0000	0.000000	
2		0.000000	0.0000	0.000000	
3		0.000000	0.0000	0.000000	
4		0.000000	0.0000	0.000000	
...	
1293		36.432873	2.0355	38.468373	
1294		6.235724	0.0000	6.235724	
1295		17.479803	1.9646	19.444403	
1296		44.938614	2.1000	47.038614	
1297		2.104763	3.5442	5.648963	
		stock_pos_value	option_pos_value	gross_pos_value	... initial_cash \
0		0.000000	0.0	0.000000	... 1.000000e+06
1		0.000000	0.0	0.000000	... 1.000030e+06
2		0.000000	0.0	0.000000	... 1.000060e+06
3		0.000000	0.0	0.000000	... 1.000094e+06
4		0.000000	0.0	0.000000	... 1.000128e+06
...
1293		-315409.338480	-43198.0	-358607.338480	... 1.468668e+06
1294		-254729.891962	-39426.0	-294155.891962	... 1.406455e+06
1295		-426806.802646	-62314.0	-489120.802646	... 1.601004e+06
1296		21125.707361	-37092.0	-15966.292639	... 1.130770e+06
1297		0.000000	0.0	0.000000	... 1.116510e+06
		interest	lever_cash	leverage_fee	end_kapital net_short_fees \
0		29.584646	0.0	0.0	1.000030e+06 0.000000
1		30.557393	0.0	0.0	1.000060e+06 0.000000
2		33.756587	0.0	0.0	1.000094e+06 0.000000
3		34.035027	0.0	0.0	1.000128e+06 0.000000
4		34.173373	0.0	0.0	1.000162e+06 0.000000
...
1293		184.932685	0.0	0.0	1.108357e+06 13455.044279
1294		180.173640	0.0	0.0	1.108497e+06 13489.339657
1295		198.630604	0.0	0.0	1.108620e+06 13545.082913
1296		139.792991	0.0	0.0	1.111169e+06 13545.082913
1297		134.936030	0.0	0.0	1.116645e+06 13545.082913
		net_interest_paid	net_interest_earned	net_pos_value	tot_cash

```

0          29.584646      0.000000  1.000030e+06  9.999970e+06
1          60.142039      0.000000  1.000060e+06  9.999940e+06
2          93.898626      0.000000  1.000094e+06  9.999906e+06
3         127.933653      0.000000  1.000128e+06  9.999872e+06
4         162.107026      0.000000  1.000162e+06  9.999838e+06
...
1293        ...           ...       ...           ...
1294        38074.710424  12983.858946  1.110246e+06  1.043213e+07
1294        38254.884065  12983.858946  1.112480e+06  1.036958e+07
1295        38453.514668  12983.858946  1.112082e+06  1.056381e+07
1296        38593.307659  12983.858946  1.114943e+06  1.009324e+07
1297        38728.243689  12983.858946  1.116645e+06  1.007870e+07

```

[1298 rows x 26 columns]

[70]: PL_dfs['trade_2'][-2:]

```

[70]:      date  gross_stock_trades  gross_option_trades  gross_trades_value \
1296  2023-02-27          0.0                  0.0          0.0
1297  2023-02-28          0.0                  0.0          0.0

      stock_trading_costs  option_trading_costs  net_trading_costs \
1296            0.0                  0.0          0.0
1297            0.0                  0.0          0.0

      stock_pos_value  option_pos_value  gross_pos_value  ...  initial_cash \
1296            0.0                  0.0          0.0  ...  683225.069753
1297            0.0                  0.0          0.0  ...  683309.534404

      interest  lever_cash  leverage_fee  end_kapital  net_short_fees \
1296  84.464651          0.0          0.0  683309.534404    4719.494988
1297  82.581537          0.0          0.0  683392.115941    4719.494988

      net_interest_paid  net_interest_earned  net_pos_value      tot_cash
1296      39039.947929          1262.251172  683309.534404  9.612474e+06
1297      39122.529466          1262.251172  683392.115941  9.612391e+06

```

[2 rows x 26 columns]

Noting some position lows, drawdowns and maximum PL values, on each of the strategies.

[71]: PL_dfs['trade_1']['net_pos_value'][PL_dfs['trade_1']['net_pos_value'].idxmin()]

[71]: 868579.2717882802

[73]: PL_dfs['trade_2']['net_pos_value'][PL_dfs['trade_2']['net_pos_value'].idxmin()]

[73]: 596810.1024514363

```
[74]: PL_dfs['trade_3']['net_pos_value'][PL_dfs['trade_3']['net_pos_value'].idxmin()]

[74]: 981452.1037887249

[75]: PL_dfs['trade_1']['tot_cash'][PL_dfs['trade_1']['tot_cash'].idxmin()]

[75]: 2987330.136676361

[76]: PL_dfs['trade_1']['tot_cash'][PL_dfs['trade_1']['tot_cash'].idxmax()]

[76]: 15620853.359602839

[77]: PL_dfs['trade_2']['tot_cash'][PL_dfs['trade_2']['tot_cash'].idxmin()]

[77]: 2468237.6778471903

[78]: PL_dfs['trade_2']['tot_cash'][PL_dfs['trade_2']['tot_cash'].idxmax()]

[78]: 15140576.721228467

[79]: PL_dfs['trade_3']['tot_cash'][PL_dfs['trade_3']['tot_cash'].idxmin()]

[79]: 5649798.585902663

[80]: PL_dfs['trade_3']['tot_cash'][PL_dfs['trade_3']['tot_cash'].idxmax()]

[80]: 15511034.440640306
```

Showing some examples of the final positions and closing them:

```
[81]: for key in strat_dict.keys():
    formatted_df = PL_dfs[key][-2:][['date', 'net_pos_value']].copy()

    formatted_df['net_pos_value'] = formatted_df['net_pos_value'].apply(lambda x: "${:,.2f}".format(x))

    print(f"Final positions for {key}:")
    display(formatted_df)
```

Final positions for trade_1:

	date	net_pos_value
1296	2023-02-27	\$1,114,943.29
1297	2023-02-28	\$1,116,644.50

Final positions for trade_2:

	date	net_pos_value
1296	2023-02-27	\$683,309.53
1297	2023-02-28	\$683,392.12

Final positions for trade_3:

```
    date  net_pos_value
1296 2023-02-27 $1,755,802.90
1297 2023-02-28 $1,759,887.75
```

As you can see above, the strategies trade_1 (long-short) and trade_3 (short only) were profitable, while strategy trade_2 (long only) was not. This corresponds with general knowledge that realized volatility is historically lower than implied volatility, which allows short gamma strategies to be profitable.

```
[82]: os.makedirs('simdata', exist_ok=True)

for strat, df in PL_dfs.items():
    csv_path = f'simdata/PL_{strat}.csv'
    df.to_csv(csv_path, index=False)
```

1.11 9. Analysis

We visualize and analyze trade data for the 3 trading strategies developed. Additionally, we also import Fama-French factors data available from Ken French's website and set the date column as the index to determine correlations / regression analyses using market factors.

1.11.1 Load Data

```
[ ]: PL_trade_1 = pd.read_csv("./simdata/PL_trade_1.csv")
PL_trade_1['date'] = pd.to_datetime(PL_trade_1['date'])
PL_trade_1.set_index('date', inplace=True)

PL_trade_2 = pd.read_csv("./simdata/PL_trade_2.csv")
PL_trade_2['date'] = pd.to_datetime(PL_trade_2['date'])
PL_trade_2.set_index('date', inplace=True)

PL_trade_3 = pd.read_csv("./simdata/PL_trade_3.csv")
PL_trade_3['date'] = pd.to_datetime(PL_trade_3['date'])
PL_trade_3.set_index('date', inplace=True)

PL_trade_1 = calculate_daily_pl(PL_trade_1)
PL_trade_2 = calculate_daily_pl(PL_trade_2)
PL_trade_3 = calculate_daily_pl(PL_trade_3)
```

Display an example:

```
[4]: PL_trade_3
```

```
[4]:      gross_stock_trades  gross_option_trades  gross_trades_value \
date
2018-01-02          0.000000                 0.0          0.000000
2018-01-03          0.000000                 0.0          0.000000
2018-01-04          0.000000                 0.0          0.000000
```

2018-01-05	0.000000	0.0	0.000000	
2018-01-08	0.000000	0.0	0.000000	
...	
2023-02-22	368794.369290	1605.0	370399.369290	
2023-02-23	97096.865918	17620.0	114716.865918	
2023-02-24	306543.247660	19646.0	326189.247660	
2023-02-27	483284.308245	4665.0	487949.308245	
2023-02-28	38086.609152	82947.0	121033.609152	
stock_trading_costs option_trading_costs net_trading_costs \				
date				
2018-01-02	0.000000	0.0000	0.000000	
2018-01-03	0.000000	0.0000	0.000000	
2018-01-04	0.000000	0.0000	0.000000	
2018-01-05	0.000000	0.0000	0.000000	
2018-01-08	0.000000	0.0000	0.000000	
...	
2023-02-22	36.879437	0.1605	37.039937	
2023-02-23	9.709687	1.7620	11.471687	
2023-02-24	30.654325	1.9646	32.618925	
2023-02-27	48.328431	0.4665	48.794931	
2023-02-28	3.808661	8.2947	12.103361	
stock_pos_value option_pos_value gross_pos_value stock_PL \				
date				
2018-01-02	0.000000	0.0	0.000000	0.000000e+00
2018-01-03	0.000000	0.0	0.000000	0.000000e+00
2018-01-04	0.000000	0.0	0.000000	0.000000e+00
2018-01-05	0.000000	0.0	0.000000	0.000000e+00
2018-01-08	0.000000	0.0	0.000000	0.000000e+00
...
2023-02-22	-310943.697780	-62028.0	-372971.697780	-1.704122e+06
2023-02-23	-215500.870702	-75166.0	-290666.870702	-1.704131e+06
2023-02-24	-519742.057446	-97544.0	-617286.057446	-1.704162e+06
2023-02-27	-38227.898496	-86690.0	-124917.898496	-1.706434e+06
2023-02-28	0.000000	0.0	0.000000	-1.707469e+06
... leverage_fee end_kapital net_short_fees \				
date	...			
2018-01-02	...	0.0	1.000030e+06	0.000000
2018-01-03	...	0.0	1.000060e+06	0.000000
2018-01-04	...	0.0	1.000094e+06	0.000000
2018-01-05	...	0.0	1.000128e+06	0.000000
2018-01-08	...	0.0	1.000162e+06	0.000000
...
2023-02-22	...	0.0	1.746584e+06	8707.858452
2023-02-23	...	0.0	1.746805e+06	8736.872258

2023-02-24	...	0.0	1.746999e+06	8804.753357
2023-02-27	...	0.0	1.749634e+06	8809.729350
2023-02-28	...	0.0	1.759888e+06	8809.729350
		net_interest_paid	net_interest_earned	net_pos_value \
date				
2018-01-02		29.584646	0.000000	1.000030e+06
2018-01-03		60.142039	0.000000	1.000060e+06
2018-01-04		93.898626	0.000000	1.000094e+06
2018-01-05		127.933653	0.000000	1.000128e+06
2018-01-08		162.107026	0.000000	1.000162e+06
...	
2023-02-22		39678.615408	10919.221232	1.748393e+06
2023-02-23		39940.185789	10919.221232	1.751442e+06
2023-02-24		40233.999182	10919.221232	1.751205e+06
2023-02-27		40466.477170	10919.221232	1.755803e+06
2023-02-28		40679.143123	10919.221232	1.759888e+06
		tot_cash	daily_option_PL	daily_stock_PL
date				daily_net_PL
2018-01-02	9.999970e+06	0.0000	0.000000	0.000000
2018-01-03	9.999940e+06	0.0000	0.000000	0.000000
2018-01-04	9.999906e+06	0.0000	0.000000	0.000000
2018-01-05	9.999872e+06	0.0000	0.000000	0.000000
2018-01-08	9.999838e+06	0.0000	0.000000	0.000000
...
2023-02-22	1.107255e+07	2669.8395	-2990.039297	-320.199797
2023-02-23	1.099280e+07	-1.7620	-9.709687	-11.471687
2023-02-24	1.131867e+07	-1.9646	-30.654325	-32.618925
2023-02-27	1.083044e+07	4679.5335	-2271.933426	2407.600074
2023-02-28	1.070918e+07	11076.7053	-1035.531006	10041.174294

[1298 rows x 28 columns]

```
[5]: fff_data = pd.read_csv('F-F_Research_Data_Factors_daily.CSV')
fff_data['date'] = pd.to_datetime(fff_data['date'])
fff_data.set_index('date', inplace=True)
fff_data
```

	Mkt-RF	SMB	HML	RF
date				
2026-07-01	0.10	-0.25	-0.27	0.009
2026-07-02	0.45	-0.33	-0.06	0.009
2026-07-06	0.17	0.30	-0.39	0.009
2026-07-07	0.09	-0.58	0.02	0.009
2026-07-08	0.21	-0.38	0.19	0.009
...

```

2024-01-25    0.46  0.04  0.56  0.022
2024-01-26   -0.02  0.40 -0.27  0.022
2024-01-29    0.85  1.07 -0.59  0.022
2024-01-30   -0.13 -1.26  0.84  0.022
2024-01-31   -1.74 -0.92 -0.30  0.022

```

[25670 rows x 4 columns]

```

[6]: start_date = '2018-01-01'
end_date = '2023-02-28'
spydata = pd.read_csv('spy_tickerdata.csv')[['date','adj_close']].\
    sort_values(by='date').reset_index(drop=True)
spydata = spydata.loc[(spydata['date'] >= start_date) & (spydata['date'] <=\
    end_date)].copy().reset_index(drop=True)
spydata['spy_return1'] = spydata['adj_close'] / spydata.loc[0, 'adj_close']
spydata['spy_return'] = (spydata['spy_return1'] * 1000000 ) - 1000000
spydata['returns'] = spydata['spy_return1'].diff() *1000000
spydata.set_index('date', inplace=True)

```

1.11.2 Data Analysis

To re-iterate, this analysis will focus on the 3 different trading strategies developed in the previous section: 1. Comparing the Model-Free Implied Vol vs BS Implied Vol to Enter Trades 2. Enter Only Long Gamma Trades 3. Enter Only Short Gamma Trades

```
[7]: performance_summary(PL_trade_1)
```

	Mean	Median	Volatility	Sharpe Ratio	\
gross_stock_trades	5.924315e+05	3.432981e+05	7.339366e+05	0.807197	
gross_option_trades	1.750643e+04	9.078500e+03	2.449568e+04	0.714674	
gross_trades_value	6.099379e+05	3.611404e+05	7.407474e+05	0.823409	
stock_trading_costs	5.924315e+01	3.432981e+01	7.339366e+01	0.807197	
option_trading_costs	1.750643e+00	9.078500e-01	2.449568e+00	0.714674	
net_trading_costs	6.099379e+01	3.611404e+01	7.407474e+01	0.823409	
stock_pos_value	3.752555e+05	1.469597e+05	1.604310e+06	0.233905	
option_pos_value	-8.080070e+04	-6.545700e+04	2.097862e+05	-0.385157	
gross_pos_value	2.944548e+05	8.302307e+04	1.556962e+06	0.189121	
stock_PL	-3.469488e+04	-4.172414e+04	2.496436e+05	-0.138978	
option_PL	1.016892e+05	7.503284e+04	2.210585e+05	0.460011	
net_PL	6.699437e+04	3.729012e+04	1.221955e+05	0.548256	
start_cash	1.079254e+06	1.051728e+06	1.240356e+05	8.701157	
initial_kapital	1.079324e+06	1.051700e+06	1.240365e+05	8.701661	
short_fee	1.043535e+01	0.000000e+00	2.737500e+01	0.381200	
initial_cash	1.052367e+06	9.307892e+05	1.142975e+06	0.920726	
interest	2.983686e+01	3.541307e+00	4.619027e+01	0.645956	
lever_cash	2.669067e+05	0.000000e+00	7.212898e+05	0.370041	
leverage_fee	1.000297e+01	0.000000e+00	3.897554e+01	0.256647	

end_kapital	1.079343e+06	1.051819e+06	1.240204e+05	8.702949
net_short_fees	6.414806e+03	6.976890e+03	3.387926e+03	1.893431
net_interest_paid	2.134996e+04	2.599523e+04	9.355687e+03	2.282031
net_interest_earned	2.586065e+03	2.055672e+03	2.199032e+03	1.176001
net_pos_value	1.079935e+06	1.054309e+06	1.247928e+05	8.653825
tot_cash	9.754367e+06	9.890187e+06	1.546236e+06	6.308458
daily_option_PL	5.726654e+02	0.000000e+00	8.968539e+03	0.063853
daily_stock_PL	-4.921991e+02	-2.651589e+01	8.369149e+03	-0.058811
daily_net_PL	8.046626e+01	-1.157372e+01	5.188377e+03	0.015509

	Skewness	Excess	Kurtosis	Min	Max	\
gross_stock_trades	2.354224	8.253499	0.000000e+00	6.412519e+06		
gross_option_trades	2.145489	5.345669	0.000000e+00	1.578750e+05		
gross_trades_value	2.336678	8.116444	0.000000e+00	6.412519e+06		
stock_trading_costs	2.354224	8.253499	0.000000e+00	6.412519e+02		
option_trading_costs	2.145489	5.345669	0.000000e+00	1.578750e+01		
net_trading_costs	2.336678	8.116444	0.000000e+00	6.412519e+02		
stock_pos_value	0.024513	2.181142	-5.886089e+06	6.684704e+06		
option_pos_value	0.253853	0.655672	-6.711300e+05	5.873630e+05		
gross_pos_value	0.121650	2.502866	-5.520087e+06	7.148315e+06		
stock_PL	-0.071932	-0.170702	-6.388745e+05	5.163709e+05		
option_PL	0.588947	0.600919	-3.638743e+05	7.433197e+05		
net_PL	0.777775	-0.103997	-9.694342e+04	4.269232e+05		
start_cash	0.799383	-0.138466	9.169981e+05	1.443109e+06		
initial_kapital	0.797592	-0.139962	9.169759e+05	1.443100e+06		
short_fee	3.887240	18.033096	-1.268325e+00	2.428192e+02		
initial_cash	1.840538	4.178353	0.000000e+00	6.659374e+06		
interest	2.462929	7.910559	-2.087234e+01	3.149349e+02		
lever_cash	3.781740	16.880801	0.000000e+00	5.974028e+06		
leverage_fee	6.241189	46.327317	0.000000e+00	4.837435e+02		
end_kapital	0.797723	-0.139504	9.169981e+05	1.443109e+06		
net_short_fees	-0.174843	-0.515759	0.000000e+00	1.354508e+04		
net_interest_paid	-0.833472	-0.245736	2.958465e+01	3.872824e+04		
net_interest_earned	2.236608	6.514927	0.000000e+00	1.298386e+04		
net_pos_value	0.779850	-0.142445	8.685793e+05	1.419346e+06		
tot_cash	0.007592	2.500485	2.987330e+06	1.562085e+07		
daily_option_PL	-1.022962	10.837477	-5.005651e+04	6.995727e+04		
daily_stock_PL	1.725971	18.123452	-6.006872e+04	7.029373e+04		
daily_net_PL	-0.987223	13.598898	-3.578368e+04	3.236006e+04		

	Max Drawdown
gross_stock_trades	-1.000000
gross_option_trades	-0.999994
gross_trades_value	-1.000000
stock_trading_costs	-0.998443
option_trading_costs	-0.940432
net_trading_costs	-0.998443

stock_pos_value	-5.052143
option_pos_value	-107839.000000
gross_pos_value	-39687.059976
stock_PL	-352.069659
option_PL	-29.182400
net_PL	-381.252059
start_cash	-0.295621
initial_kapital	-0.295611
short_fee	-1.001101
initial_cash	-1.000000
interest	-1.083401
lever_cash	-1.000000
leverage_fee	-0.997937
end_kapital	-0.295621
net_short_fees	-0.000169
net_interest_paid	-0.000797
net_interest_earned	0.000000
net_pos_value	-0.304238
tot_cash	-0.802107
daily_option_PL	-6.703200
daily_stock_PL	-80.140810
daily_net_PL	-80.140810

Trading Strategy 1 Summary Stats (Model Free IV vs BS IV):

Gross Stock Trades show substantial trading activity with a mean significantly higher than the median, which indicates a skewed distribution with some exceptionally large stock trades. The zero minimum and the high maximum further confirm a wide range of trade values. Gross Option Trades, while lower on average than stock trades (as indicated by the mean), still show significant activity. The difference between the mean and median is less pronounced than for stock trades, but still suggests right-skewed data, pointing to fewer but larger outlier trades driving up the average. Gross Trades Value consolidates the total value from both stocks and options, revealing high trading activity overall. The mean is larger than the median, suggesting a right-skewed distribution typical for financial data where a few high-value trades increase the average trade value.

Stock_PL indicates a loss on average (negative mean), suggesting that stock trades have generally not been profitable. The negative skewness indicates that there are more losses piling up on the higher end (left side of the distribution). Option_PL, in contrast, shows a positive mean, indicating overall profitability in option trading. This suggests that while stock trading was not profitable on average, option trading appears to have offset some of those losses. A positive mean for Net_PL suggests that despite losses in stock trading, the total trading activities (including options) have been generally profitable. Cash Reserves:

Tot_Cash represents the total available cash, with a mean significantly higher than the median, again indicating a right-skewed distribution. This suggests that while the typical (median) cash reserve is lower, there are instances of significantly higher cash reserves, likely reflecting successful trades or the accumulation of profits over time. Overall, the trading strategy seems to have sufficient liquidity as well.

[8]: performance_summary(PL_trade_2)

	Mean	Median	Volatility	Sharpe Ratio	\
gross_stock_trades	1.933879e+05	0.000000e+00	609598.390241	0.317238	
gross_option_trades	5.317485e+03	0.000000e+00	19896.957856	0.267251	
gross_trades_value	1.987054e+05	0.000000e+00	618058.226745	0.321500	
stock_trading_costs	1.933879e+01	0.000000e+00	60.959839	0.317238	
option_trading_costs	5.317485e-01	0.000000e+00	1.989696	0.267251	
net_trading_costs	1.987054e+01	0.000000e+00	61.805823	0.321500	
stock_pos_value	-4.745100e+04	0.000000e+00	969208.000600	-0.048959	
option_pos_value	4.250401e+04	0.000000e+00	111077.523390	0.382652	
gross_pos_value	-4.946988e+03	0.000000e+00	969267.911742	-0.005104	
stock_PL	3.694258e+05	1.126285e+05	314278.056367	1.175474	
option_PL	-5.249225e+05	-1.378639e+05	468552.306903	-1.120307	
net_PL	-1.554966e+05	-4.706002e+04	160436.783127	-0.969208	
start_cash	8.644422e+05	9.782579e+05	155776.372500	5.549251	
initial_kapital	8.641691e+05	9.782560e+05	155802.539865	5.546566	
short_fee	3.635975e+00	0.000000e+00	17.188871	0.211531	
initial_cash	9.572315e+05	9.782579e+05	687646.847168	1.392039	
interest	3.014062e+01	3.349487e+01	30.913563	0.974997	
lever_cash	9.023479e+04	0.000000e+00	556606.755466	0.162116	
leverage_fee	9.724585e-01	0.000000e+00	5.716310	0.170120	
end_kapital	8.641983e+05	9.782569e+05	155811.821466	5.546423	
net_short_fees	3.129886e+03	3.493760e+03	1561.184468	2.004815	
net_interest_paid	2.351500e+04	2.930935e+04	10123.778714	2.322750	
net_interest_earned	6.901988e+02	3.907693e+02	422.725771	1.632734	
net_pos_value	8.620789e+05	9.782434e+05	156272.602776	5.516507	
tot_cash	9.824506e+06	9.924143e+06	979489.191186	10.030234	
daily_option_PL	-8.836072e+02	0.000000e+00	6242.512891	-0.141547	
daily_stock_PL	6.141553e+02	0.000000e+00	5674.511433	0.108231	
daily_net_PL	-2.694520e+02	0.000000e+00	3918.606039	-0.068762	

	Skewness	Excess	Kurtosis	Min	Max	\
gross_stock_trades	4.527659	25.512258	0.000000e+00	6.412519e+06		
gross_option_trades	3.985601	15.929008	0.000000e+00	1.578750e+05		
gross_trades_value	4.479469	24.919250	0.000000e+00	6.412519e+06		
stock_trading_costs	4.527659	25.512258	0.000000e+00	6.412519e+02		
option_trading_costs	3.985601	15.929008	0.000000e+00	1.578750e+01		
net_trading_costs	4.479469	24.919250	0.000000e+00	6.412519e+02		
stock_pos_value	0.107321	18.130890	-6.050401e+06	6.684704e+06		
option_pos_value	3.003672	8.805896	0.000000e+00	6.445940e+05		
gross_pos_value	1.127853	19.635929	-5.520087e+06	7.198018e+06		
stock_PL	0.328478	-1.711151	-1.211316e+03	7.971735e+05		
option_PL	-0.370850	-1.738031	-1.146922e+06	0.000000e+00		
net_PL	-0.428780	-1.664026	-4.142224e+05	5.989147e+04		
start_cash	-0.442549	-1.645331	6.094263e+05	1.084978e+06		
initial_kapital	-0.439247	-1.648369	6.094268e+05	1.085015e+06		

short_fee	7.007721	60.483658	-1.867961e+00	2.428192e+02
initial_cash	4.001612	21.396654	0.000000e+00	6.193564e+06
interest	1.109809	2.769934	-2.229330e+01	2.389453e+02
lever_cash	7.655120	63.935040	0.000000e+00	6.479074e+06
leverage_fee	7.487355	66.467590	0.000000e+00	8.246907e+01
end_kapital	-0.439263	-1.648375	6.094263e+05	1.084978e+06
net_short_fees	-0.513733	-1.226650	0.000000e+00	4.719495e+03
net_interest_paid	-0.996940	-0.352847	2.958465e+01	3.912253e+04
net_interest_earned	0.175119	-1.506167	0.000000e+00	1.262251e+03
net_pos_value	-0.415887	-1.666362	5.968101e+05	1.139550e+06
tot_cash	-1.093957	18.163451	2.468238e+06	1.514058e+07
daily_option_PL	-3.213462	43.840489	-5.467579e+04	6.995953e+04
daily_stock_PL	5.732236	78.368809	-6.006872e+04	7.030262e+04
daily_net_PL	-1.765585	45.020162	-4.063736e+04	3.233784e+04

	Max Drawdown
gross_stock_trades	-9.99998e-01
gross_option_trades	-9.999937e-01
gross_trades_value	-9.99998e-01
stock_trading_costs	-9.984430e-01
option_trading_costs	-9.404319e-01
net_trading_costs	-9.984430e-01
stock_pos_value	-1.026891e+01
option_pos_value	-9.999984e-01
gross_pos_value	-4.138909e+00
stock_PL	-1.211316e+03
option_PL	-1.146922e+06
net_PL	-4.869690e+04
start_cash	-4.383051e-01
initial_kapital	-4.383239e-01
short_fee	-1.003560e+00
initial_cash	-9.99998e-01
interest	-1.088742e+00
lever_cash	-9.99998e-01
leverage_fee	-9.880195e-01
end_kapital	-4.383051e-01
net_short_fees	-4.647275e-04
net_interest_paid	-7.548281e-04
net_interest_earned	0.000000e+00
net_pos_value	-4.762751e-01
tot_cash	-8.369786e-01
daily_option_PL	-3.528206e+04
daily_stock_PL	-2.329447e+02
daily_net_PL	-5.649311e+03

Trading Strategy 2 Summary Stats (Long Gamma):

Stock_Pos_Value shows a negative mean, indicating an overall decrease in the value of stock

positions. Option_Pos_Value, on the other hand, presents a positive mean, suggesting that, on average, option positions have increased in value or have been more profitable compared to stock positions. This is further supported by the positive skewness, indicating a distribution leaning towards gains in option values. Gross_Pos_Value combines the values of stock and option positions. The negative mean here suggests that the overall position values, when combined, have decreased, mainly dragged down by the losses in stock positions despite the gains from options.

Stock_PL and Option_PL metrics reflect the cumulative profit and loss from stocks and options, respectively. The positive mean in Stock_PL indicates overall gains in stock trading, whereas a significantly negative mean in Option_PL contradicts the positive option position value. Net_PL has a negative mean here and it indicates that, on average, the total trading activity resulted in a loss, which aligns with the negative gross position value. The infinite maximum drawdowns highlight the potential for catastrophic losses, severely questioning the viability of the strategy under adverse market conditions. This preliminary analysis reveals that this strategy has high risk.

This is again what we expect - we lock in small gains from delta hedging, but the option position usually does not profit relative to the premium paid. The losses are reflected in the data.

```
[9]: performance_summary(PL_trade_3)
```

	Mean	Median	Volatility	Sharpe Ratio	\
gross_stock_trades	5.099952e+05	3.077805e+05	5.973523e+05	0.853759	
gross_option_trades	1.193424e+04	9.756000e+03	1.269503e+04	0.940072	
gross_trades_value	5.219294e+05	3.212479e+05	6.004285e+05	0.869262	
stock_trading_costs	5.099952e+01	3.077805e+01	5.973523e+01	0.853759	
option_trading_costs	1.193424e+00	9.756000e-01	1.269503e+00	0.940072	
net_trading_costs	5.219294e+01	3.212479e+01	6.004285e+01	0.869262	
stock_pos_value	6.295601e+05	3.592170e+05	1.366080e+06	0.460851	
option_pos_value	-1.580316e+05	-1.295520e+05	1.383358e+05	-1.142376	
gross_pos_value	4.715285e+05	2.483664e+05	1.335930e+06	0.352959	
stock_PL	-4.870277e+05	-3.260604e+05	4.295566e+05	-1.133792	
option_PL	8.432022e+05	6.323162e+05	6.959722e+05	1.211546	
net_PL	3.561745e+05	2.843840e+05	3.219589e+05	1.106273	
start_cash	1.367387e+06	1.290608e+06	3.252177e+05	4.204529	
initial_kapital	1.367950e+06	1.299248e+06	3.252429e+05	4.205933	
short_fee	6.787157e+00	0.000000e+00	2.302262e+01	0.294804	
initial_cash	1.089658e+06	1.000458e+06	1.117872e+06	0.974761	
interest	3.133986e+01	3.092063e+00	5.446732e+01	0.575388	
lever_cash	1.891424e+05	0.000000e+00	4.663580e+05	0.405573	
leverage_fee	8.412343e+00	0.000000e+00	3.060027e+01	0.274911	
end_kapital	1.367973e+06	1.299249e+06	3.252398e+05	4.206043	
net_short_fees	2.823143e+03	2.724356e+03	1.941965e+03	1.453756	
net_interest_paid	1.861103e+04	2.201835e+04	8.719365e+03	2.134448	
net_interest_earned	3.989804e+03	4.429460e+03	2.046663e+03	1.949419	
net_pos_value	1.372067e+06	1.346078e+06	3.265734e+05	4.201404	
tot_cash	9.874119e+06	9.997980e+06	1.370007e+06	7.207351	
daily_option_PL	1.884751e+03	0.000000e+00	6.141750e+03	0.306875	
daily_stock_PL	-1.315462e+03	-2.600564e+01	5.998969e+03	-0.219281	

daily_net_PL	5.692893e+02	-5.238310e+00	3.307831e+03	0.172104	
	Skewness	Excess	Kurtosis	Min	Max \
gross_stock_trades	1.984503	5.247189	0.000000e+00	4.768780e+06	
gross_option_trades	1.256298	2.034144	0.000000e+00	8.294700e+04	
gross_trades_value	1.966540	5.168831	0.000000e+00	4.792298e+06	
stock_trading_costs	1.984503	5.247189	0.000000e+00	4.768780e+02	
option_trading_costs	1.256298	2.034144	0.000000e+00	8.294700e+00	
net_trading_costs	1.966540	5.168831	0.000000e+00	4.792298e+02	
stock_pos_value	0.210565	0.905196	-4.296727e+06	5.408364e+06	
option_pos_value	-0.932048	0.402343	-6.711300e+05	0.000000e+00	
gross_pos_value	-0.081621	1.165026	-4.747632e+06	4.981198e+06	
stock_PL	-1.450664	1.283994	-1.707469e+06	1.023171e+03	
option_PL	0.704176	-0.714034	-1.615640e+01	2.446407e+06	
net_PL	0.175024	-1.758702	-2.000261e+04	8.382902e+05	
start_cash	0.171141	-1.759839	9.879245e+05	1.852314e+06	
initial_kapital	0.168153	-1.761018	9.879361e+05	1.852313e+06	
short_fee	5.224135	31.838692	0.000000e+00	2.087994e+02	
initial_cash	1.498259	2.594988	0.000000e+00	6.542097e+06	
interest	3.141633	12.473001	-6.525747e+00	3.855256e+02	
lever_cash	3.084174	10.546752	0.000000e+00	3.318021e+06	
leverage_fee	5.964587	45.994124	0.000000e+00	4.045936e+02	
end_kapital	0.168168	-1.761030	9.879245e+05	1.852314e+06	
net_short_fees	1.030274	1.502672	0.000000e+00	8.809729e+03	
net_interest_paid	-0.363402	0.051323	2.958465e+01	4.067914e+04	
net_interest_earned	0.191739	0.941216	0.000000e+00	1.091922e+04	
net_pos_value	0.152497	-1.759854	9.814521e+05	1.880895e+06	
tot_cash	0.463977	1.220143	5.649799e+06	1.551103e+07	
daily_option_PL	0.426346	8.039548	-4.236907e+04	4.370388e+04	
daily_stock_PL	-0.639158	5.808489	-3.462611e+04	3.266177e+04	
daily_net_PL	-0.465936	8.052576	-2.339932e+04	1.835858e+04	

	Max Drawdown
gross_stock_trades	-1.000000
gross_option_trades	-0.999988
gross_trades_value	-1.000000
stock_trading_costs	-0.997082
option_trading_costs	-0.889483
net_trading_costs	-0.996769
stock_pos_value	-5.052143
option_pos_value	-671130.000000
gross_pos_value	-39687.059976
stock_PL	-1668.170888
option_PL	-16.156400
net_PL	-506.680632
start_cash	-0.106368
initial_kapital	-0.106373

short_fee	-0.995234
initial_cash	-1.000000
interest	-1.022907
lever_cash	-1.000000
leverage_fee	-0.997534
end_kapital	-0.106368
net_short_fees	0.000000
net_interest_paid	-0.000294
net_interest_earned	0.000000
net_pos_value	-0.137514
tot_cash	-0.632859
daily_option_PL	-3.895200
daily_stock_PL	-118.711298
daily_net_PL	-118.711298

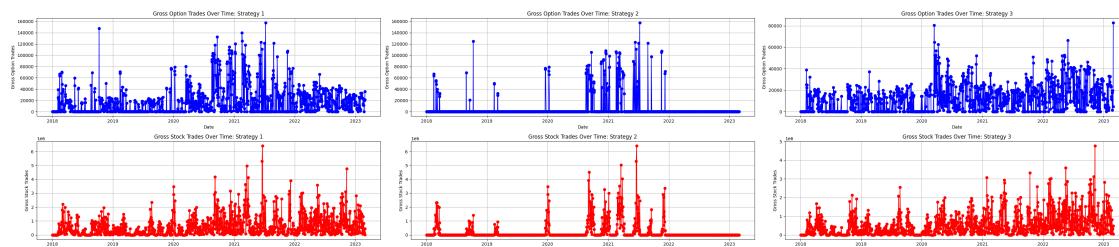
Trading Strategy 3 Summary Stats (Short Gamma):

Gross Stock Trades and Gross Option Trades show significant trading activity, with stock trades having a higher mean value, indicating more substantial volume compared to options. Both distributions show right-skewness, implying that while most of the trades are smaller, there are a few extremely large trades that increase the average trading volume. Gross Trades Value, representing the combined value of stock and option trades, also shows a high mean value with a right-skewed distribution, indicating that total trading activity encompasses a few very high-value trades alongside many smaller ones, typical in varied trading strategies.

Stock_PL indicates a substantial average loss (negative mean), suggesting that stock trading activities have generally resulted in losses. The negative skewness further points towards more significant losses than gains in the stock segment. Option_PL, on the other hand, shows a significant positive mean, indicating profitable outcomes on average from options trading. The positive skewness here suggests that while most option trades are moderately profitable, there are a few outstandingly profitable trades skewing the average upwards. Net_PL indicates that the gains from option trades have been sufficient to offset the losses from stock trades, leading to overall profitability. Yet, the max drawdown figures show significant potential for losses.

This is what we expect from being short gamma: in general, we expect our option positions to profit, while we trade some stock positions at a loss.

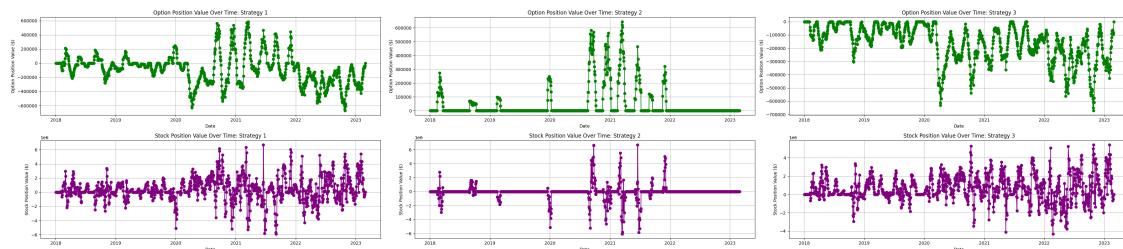
[10]: `plot_trades_over_time(PL_trade_1, PL_trade_2, PL_trade_3)`



The plots above show patterns in gross option trades and gross stock trades over time for three different trading strategies.

For Strategy 1 and 3, the volume of gross option trades and gross stock trades is quite high with a significant number of spikes, suggesting periods of high trading activity coupled with quieter times. Strategy 2, on the other hand, has significantly fewer spikes in both gross option and stock volume traded over time. For all three strategies, the volume of option trades is significantly higher and more volatile than stock trades, as indicated by the greater frequency and amplitude of peaks in the blue lines.

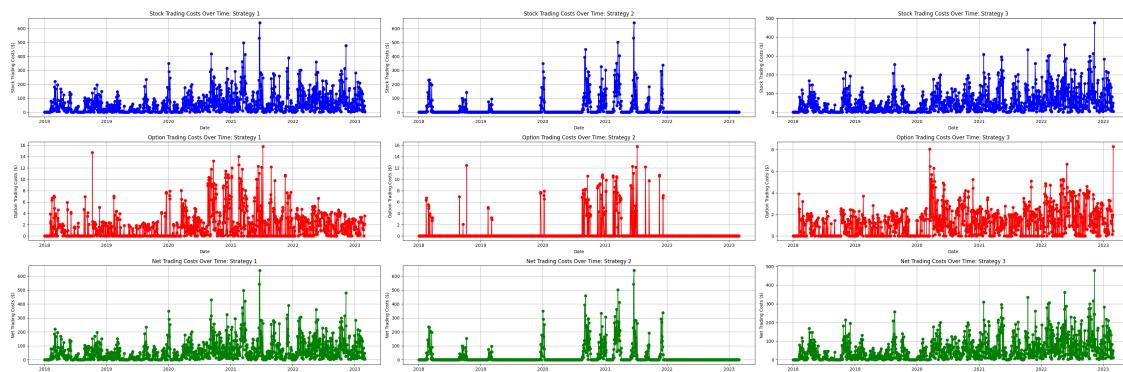
[11]: `plot_position_values_over_time(PL_trade_1, PL_trade_2, PL_trade_3)`



The plots above illustrate the option and stock position values over time for the 3 trading strategies.

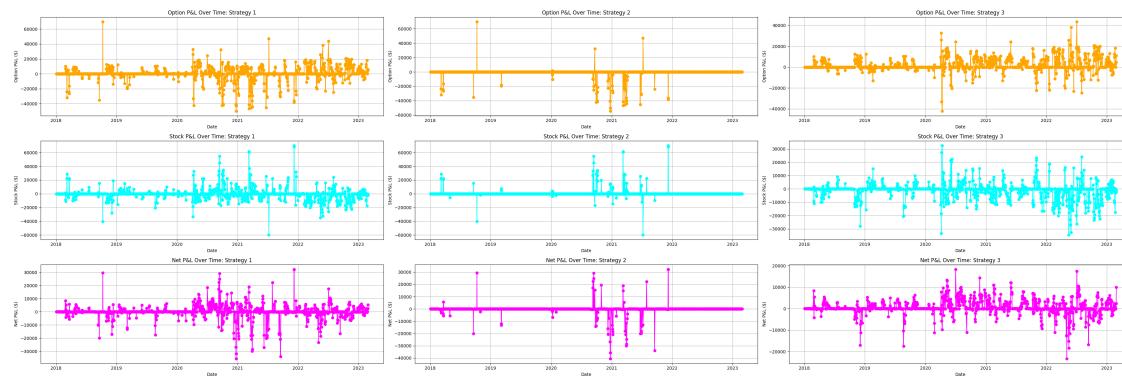
In Strategy 1, there is significant fluctuations in option position value, with some periods showing substantial positive values, especially around 2020 and 2021, suggesting profitable option positions, followed by sharp declines indicating losses which can be attributed to the covid-19 pandemic. The stock position value for Strategy 1, while also volatile, generally oscillates around a narrower range compared to the options, indicating less extreme changes in value but with notable periods of loss. Strategy 3 shows a similar pattern to Strategy 1 in terms of fluctuations. For strategy 2, the option position value always stays positive and the stock position value has minimal deviation from zero. For strategy 3, however, the option position value over time fluctuates below zero while the stock position value oscillates both positive and negative.

[12]: `plot_trading_costs_over_time(PL_trade_1, PL_trade_2, PL_trade_3)`



The plots above show a visual representation of the trading costs associated with all 3 trading strategies.

[13]: `plot_pl_over_time(PL_trade_1, PL_trade_2, PL_trade_3)`

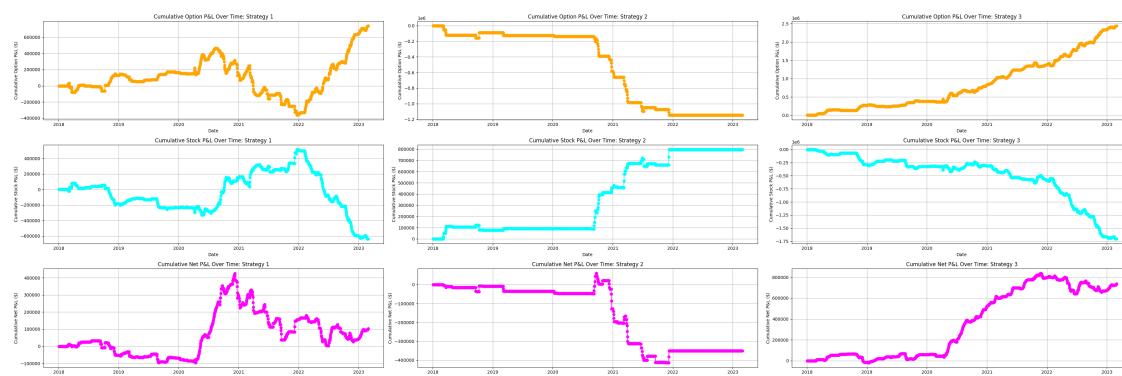


The plots above display the profit and loss (P&L) performance over time for the 3 different trading strategies. Across all strategies, option P&L tends to show greater variability and range compared to stock P&L, suggesting that options trading contributes more significantly to overall P&L fluctuations.

For Strategy 1, there's a consistent pattern of smaller losses and gains in stock trading, with larger outcomes in options trading, leading to an overall net P&L that exhibits a high-risk approach. Strategy 2 exhibits relatively less volatility particularly in the stock pnL, which appears to be slightly more stable. The net P&L for Strategy 2, while also volatile, does not seem to have very high peaks.

Overall, all three strategies demonstrate periods of both profit and loss, with options trading contributing more prominently to the volatility and magnitude of overall results.

[14]: `plot_cumulative_pl_over_time(PL_trade_1, PL_trade_2, PL_trade_3)`



The graphs above show the cumulative profit and loss (P&L) over time for the three different

trading strategies.

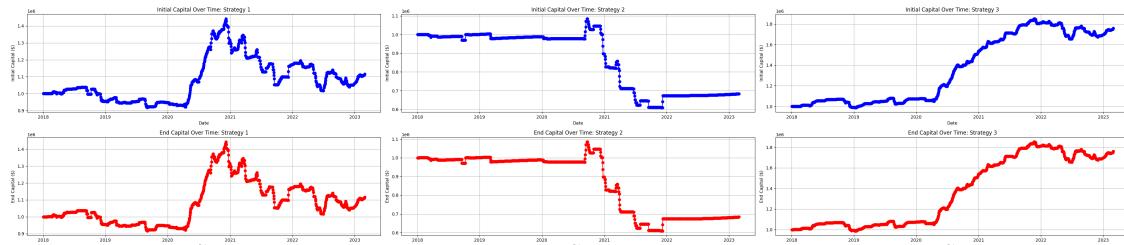
For Strategy 1, the cumulative option P&L shows an overall increasing trend (with a drop between 2020 to 2022), indicating profitable option trading activities by the end of the time period. In contrast, the cumulative stock P&L exhibits a general decline, especially after 2022, suggesting that stock trades were less successful. The net P&L shows fluctuations with a general downward trend until around mid-2020, followed by some recovery till 2021 but still underperforming.

For Strategy 2, the cumulative option P&L is mostly flat but then sharply drops in mid-2021, reflecting significant losses. The stock P&L shows a gradual increase, indicating more consistent gains from stock trading, a contrast to the option results. However, the net P&L mirrors the option performance closely, plunging in 2021, which indicates the losses in options heavily impacted the overall strategy performance.

Strategy 3 reveals continuous growth in option P&L, signifying a consistently profitable option trading strategy over time. The stock P&L, however, declines steadily, suggesting persistent losses in stock trading. The net P&L for Strategy 3 shows improvement and growth, particularly from 2020 onwards, indicating that the gains from options have been sufficient to not only recover from any stock losses but also to generate overall profit.

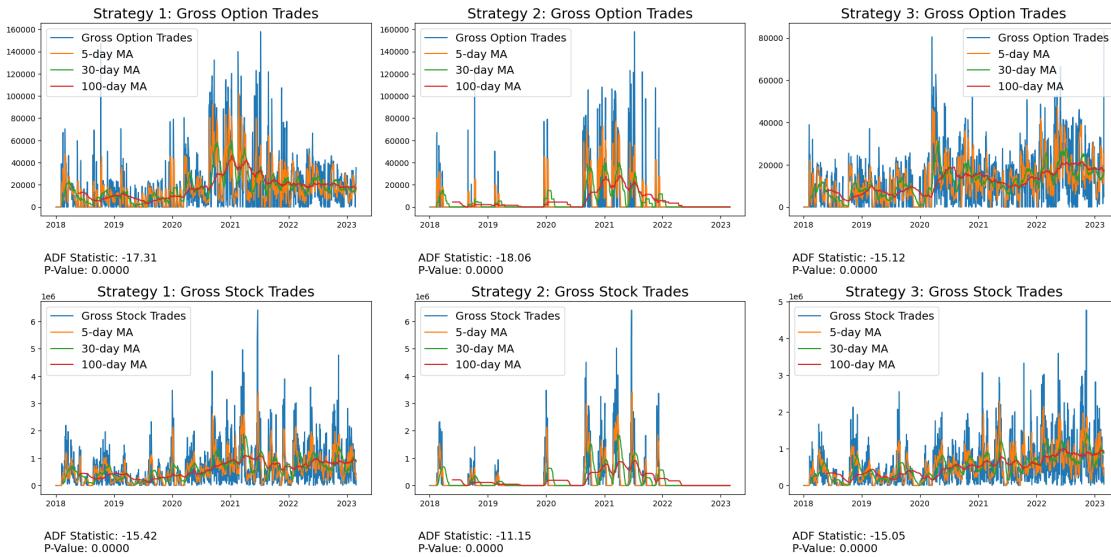
Altogether, this tends to show what one would expect: because options price IV higher than RV, being short gamma is generally a winning strategy, with exceptions occurring with large swings where the trader cannot react to the book in time.

[15]: `plot_kapital_over_time(PL_trade_1, PL_trade_2, PL_trade_3)`



In the plots above, an increase in initial capital until 2021, which sees some decrease. Strategy 2 exhibits a stable initial capital until 2020, then both initial and end capitals plummet significantly with no signs of recovery. Strategy 3 demonstrates consistent growth in both initial and end capitals from 2020 onwards, indicating a successful and effective investment approach. While we do have to treat this with some lookahead bias, it is quite consistent that options are priced with more IV than realized volatility recently, so in general our predictions are logically sound.

[16]: `plot_trades_and_test_stationarity(PL_trade_1, PL_trade_2, PL_trade_3)`



The charts show the gross option and stock trades over time for three different trading strategies, including their 5-day, 30-day, and 100-day moving averages (MAs). The moving averages smooth out short-term fluctuations and help identify longer-term trends in trading activity. The Augmented Dickey-Fuller (ADF) test results provided at the bottom of each graph further tell us about stationarity in the time series data.

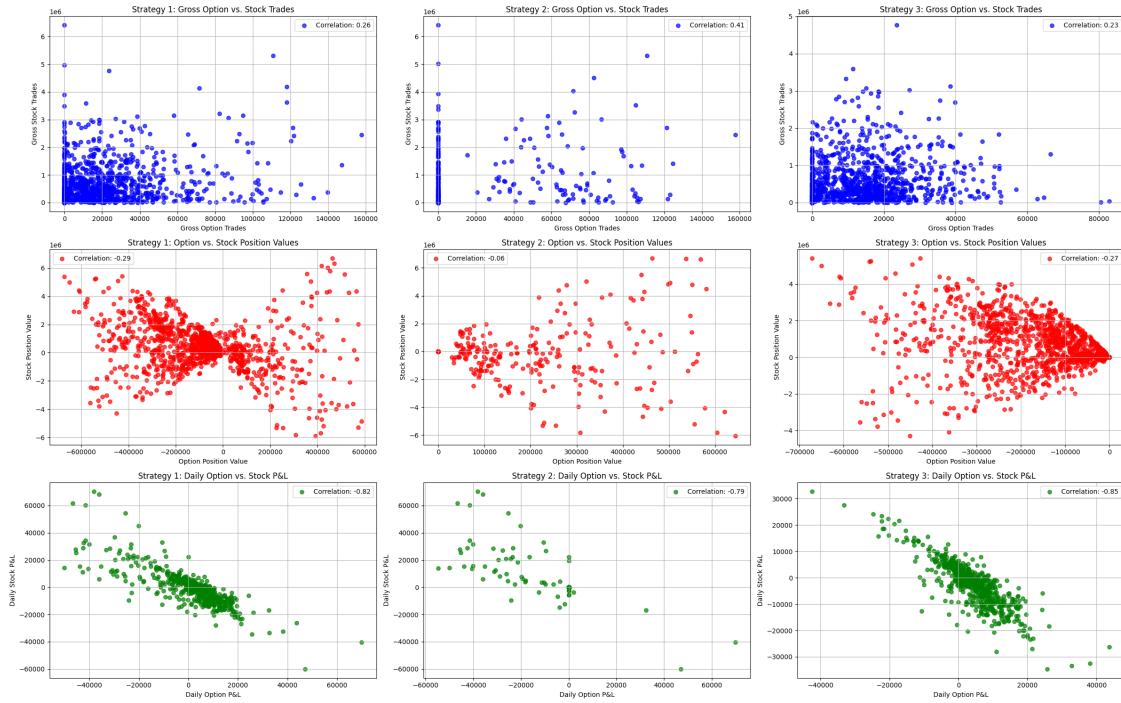
For strategy 1, both gross option trades and gross stock trades have very negative ADF statistic values (-17.31 and -15.42, respectively) with p-values of 0.0000. This suggests strong evidence against the null hypothesis.

For strategy 2, the ADF statistics for gross option trades and gross stock trades are -18.06 and -11.15 respectively. Similar to Strategy 1, these results strongly suggest that the series are stationary as the null hypothesis.

For strategy 3, again, the ADF statistics are -15.12 for gross option trades and -15.05 for gross stock trades, once again indicating that the null hypothesis can be rejected.

The consistency in the stationarity across all strategies for both option and stock trades indicates that these trading behaviors have stable dynamics, which makes them more predictable and relatively more manageable from a risk perspective.

```
[17]: plot_correlations(PL_trade_1, PL_trade_2, PL_trade_3)
```



The charts display scatter plots with correlation coefficients, comparing Gross Option Trades vs. Stock Trades, Option vs. Stock Position Values, and Daily Option P&L vs. Stock P&L.

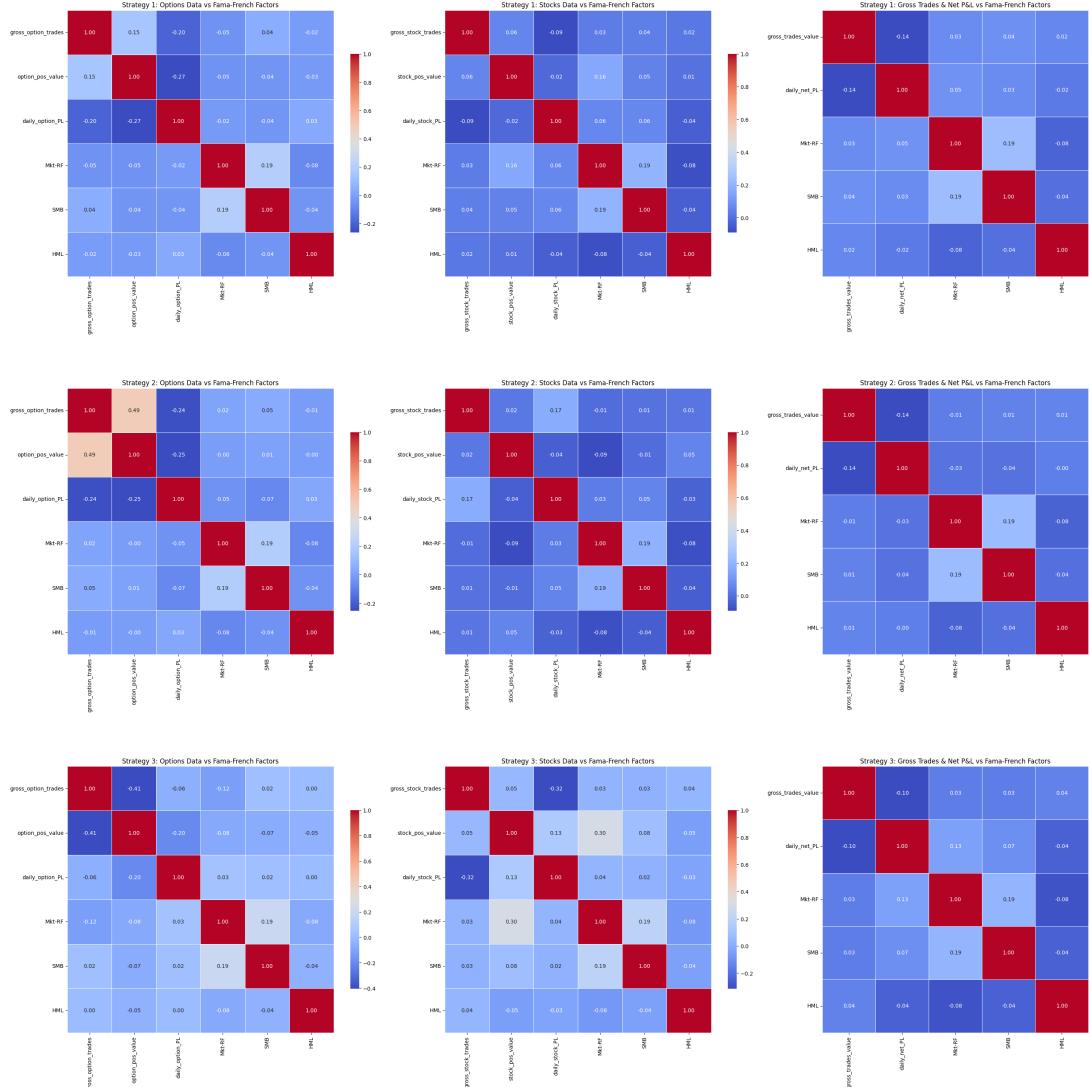
For Gross Option Trades vs. Stock Trades, Strategy 1 has a low positive correlation (0.26), indicating a slight relationship between the volume of option trades and stock trades. The data points are spread out, showing no strong linear relationship. Strategy 2 shows a moderate positive correlation (0.41), suggesting a greater relationship between the volume of options and stock trades compared to Strategy 1. Strategy 3 has a low positive correlation (0.23), similar to Strategy 1, indicating a slight relationship between the volumes.

For Option vs. Stock Position Values, all three strategies show negative correlations: -0.29 for Strategy 1, -0.06 for Strategy 2, and -0.27 for Strategy 3. These correlations suggest that, generally, as the value of option positions increases, the value of stock positions decreases, and the other way around.

For Daily Option P&L vs. Stock P&L, Strategy 1 exhibits a strong negative correlation (-0.82), indicating that on days when options P&L is high, stock P&L tends to be low, and the other way around. Strategy 2 and Strategy 3 both exhibit strong negative correlations as well (-0.79 and -0.85, respectively)

The strong negative correlations between daily option P&L and stock P&L across all strategies indicate a hedging relationship where options offset stock losses and stocks offset options losses, which is what we want - we want to lock in profits as the trading strategy progresses.

```
[18]: plot_correlation_matrices(PL_trade_1, PL_trade_2, PL_trade_3, fff_data=fff_data)
```



The plots above show correlation matrices between vgross option trades, gross stock trades, daily option P&L, and daily stock P&L and Fama-French factors (market return - Mkt, small minus big - SMB, high minus low - HML, and risk-free rate - RF) for three different strategies.

For strategy 1, gross option trades, stock trades, and daily P&Ls have relatively low to correlations with market returns (Mkt), indicating that these activities are not strongly driven by overall market movements. Similar to Strategy 1, there is similar low correlation between the trading metrics and farma french factors for strategy 2. Stock position value has a correlation of -0.09 with the market factor. For strategy 3, stock position value has a moderate positive correlation of 0.30 with the market factor while the daily net PL has a slightly positive correlation of 0.13 with the market factor and 0.07 with SMB.

In summary, these correlation matrices provide insight into how each strategy's trades and daily performance relate very weakly to broader market and factor behaviors. Given that we are trading a strategy without any particular preference for directional movement, we consider this to be a

success.

We can also relate them to F-F factors - from which we see very little impact (which is good - we should not see any impact from a simple delta-hedging strategy, even with our model-free IV implementation):

```
[19]: combined_1 = PL_trade_1.merge(fff_data, left_index=True, right_index=True, u
    ↪how='left')
combined_2 = PL_trade_2.merge(fff_data, left_index=True, right_index=True, u
    ↪how='left')
combined_3 = PL_trade_3.merge(fff_data, left_index=True, right_index=True, u
    ↪how='left')
```

```
[20]: run_regression_option_PL(combined_1, combined_2, combined_3)
```

Strategy 1: R-squared of the regression of option_PL on the Fama-French factors:
0.004904

Strategy 1 regression summary:

OLS Regression Results

```
=====
Dep. Variable:          option_PL    R-squared:       0.005
Model:                 OLS        Adj. R-squared:   0.003
Method:                Least Squares   F-statistic:     2.126
Date:      Fri, 08 Mar 2024   Prob (F-statistic): 0.0952
Time:         22:40:10        Log-Likelihood:   -17812.
No. Observations:      1298        AIC:            3.563e+04
Df Residuals:          1294        BIC:            3.565e+04
Df Model:                  3
Covariance Type:    nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
const	1.015e+05	6130.440	16.560	0.000	8.95e+04	1.14e+05
Mkt-RF	2552.7593	4478.698	0.570	0.569	-6233.546	1.13e+04
SMB	1.743e+04	8872.607	1.965	0.050	24.997	3.48e+04
HML	-6264.0446	5590.165	-1.121	0.263	-1.72e+04	4702.736

```
=====
Omnibus:                 80.328   Durbin-Watson:           0.012
Prob(Omnibus):            0.000   Jarque-Bera (JB):      97.083
Skew:                      0.599   Prob(JB):             8.29e-22
Kurtosis:                  3.600   Cond. No.              2.06
=====
```

Notes:

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

Strategy 2: R-squared of the regression of option_PL on the Fama-French factors:
0.003838

Strategy 2 regression summary:

OLS Regression Results

Dep. Variable:	option_PL	R-squared:	0.004
Model:	OLS	Adj. R-squared:	0.002
Method:	Least Squares	F-statistic:	1.662
Date:	Fri, 08 Mar 2024	Prob (F-statistic):	0.173
Time:	22:40:10	Log-Likelihood:	-18787.
No. Observations:	1298	AIC:	3.758e+04
Df Residuals:	1294	BIC:	3.760e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
const	-5.253e+05	1.3e+04	-40.403	0.000	-5.51e+05	-5e+05
Mkt-RF	4517.8311	9498.065	0.476	0.634	-1.41e+04	2.32e+04
SMB	2882.0575	1.88e+04	0.153	0.878	-3.4e+04	3.98e+04
HML	-2.507e+04	1.19e+04	-2.114	0.035	-4.83e+04	-1810.181

Omnibus:	5996.765	Durbin-Watson:	0.008
Prob(Omnibus):	0.000	Jarque-Bera (JB):	190.199
Skew:	-0.373	Prob(JB):	5.00e-42
Kurtosis:	1.279	Cond. No.	2.06

Notes:

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

Strategy 3: R-squared of the regression of option_PL on the Fama-French factors:
0.002243

Strategy 3 regression summary:

OLS Regression Results

Dep. Variable:	option_PL	R-squared:	0.002
Model:	OLS	Adj. R-squared:	-0.000
Method:	Least Squares	F-statistic:	0.9697
Date:	Fri, 08 Mar 2024	Prob (F-statistic):	0.406
Time:	22:40:10	Log-Likelihood:	-19302.

```

No. Observations: 1298 AIC: 3.861e+04
Df Residuals: 1294 BIC: 3.863e+04
Df Model: 3
Covariance Type: nonrobust
=====
            coef    std err      t    P>|t|    [0.025    0.975]
-----
const     8.436e+05  1.93e+04   43.649    0.000  8.06e+05  8.82e+05
Mkt-RF   -5450.4088 1.41e+04   -0.386    0.700 -3.31e+04  2.22e+04
SMB      1.193e+04  2.8e+04    0.427    0.670 -4.29e+04  6.68e+04
HML      2.828e+04  1.76e+04   1.605    0.109 -6295.462  6.29e+04
=====
Omnibus: 157.202 Durbin-Watson: 0.004
Prob(Omnibus): 0.000 Jarque-Bera (JB): 134.842
Skew: 0.707 Prob(JB): 5.24e-30
Kurtosis: 2.296 Cond. No. 2.06
=====
```

Notes:

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

The three regression results that use the Fama-French factors to predict option_PL show very small R² values, with the best performing regression being strategy 1 with an R² value of 0.005. No factors individually had a significant p-value.

[]: run_regression_stock_PL(combined_1, combined_2, combined_3)

Very similar to the regression results from option_PL, the three regression results that use the Fama-French factors to predict stock_PL show very small R² values, with the best performing regression again being strategy 1 with an R² value of 0.005. No factors individually had a significant p-value.

[]: run_regression_net_PL(combined_1, combined_2, combined_3)

Once again, very similar to the regression results from option_PL and stock_PL, the three regression results that use the Fama-French factors to predict net_PL show very small R² values, with the best performing regression again being strategy 1 with an R² value of 0.008. Once again, no factors individually had a significant p-value in any of the 3 regressions (although the HML factor was close for strategy 1 and strategy 3).

[]: plot_rolling_volatility(combined_1, combined_2, combined_3, window_size=30)

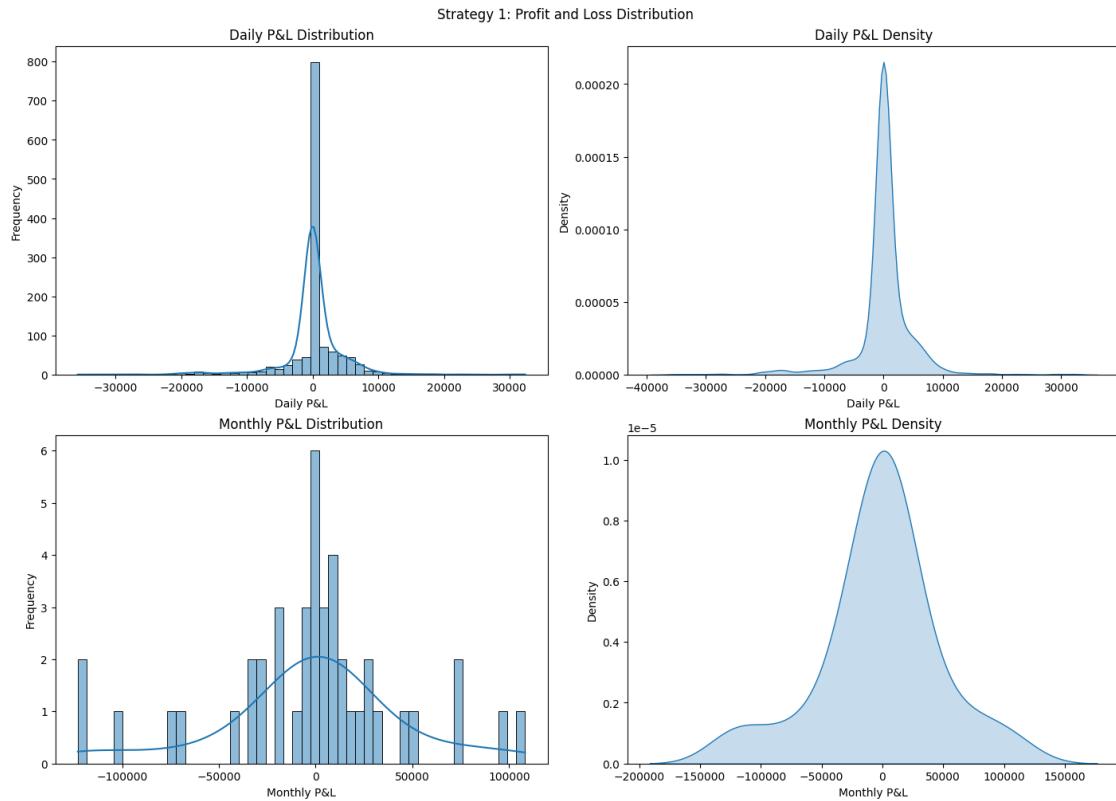
The graphs illustrate the 30-day rolling volatility of returns for the 3 trading strategies.

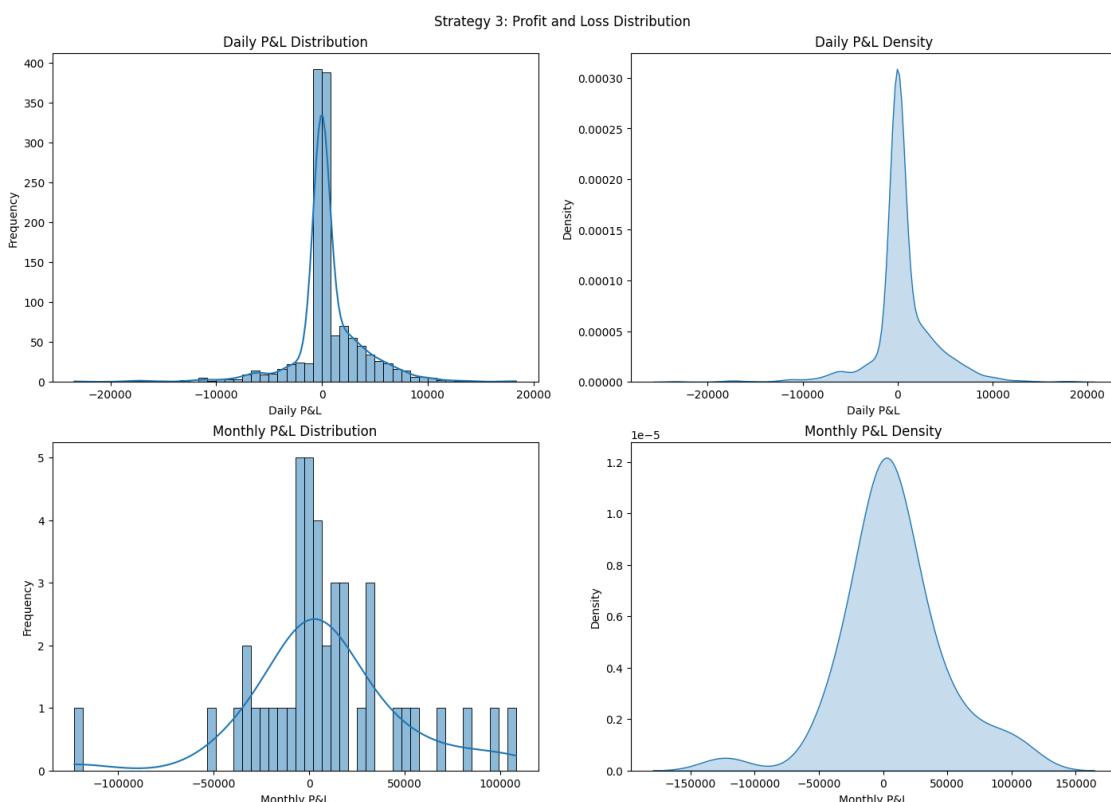
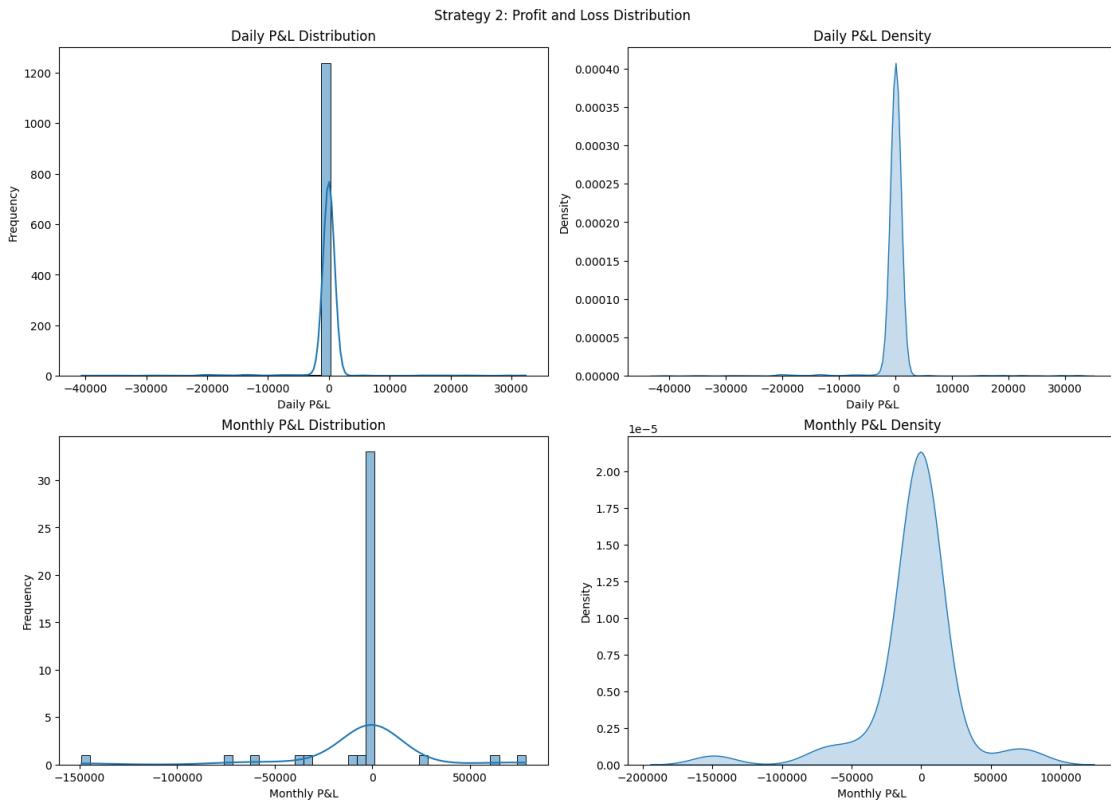
The volatility profile for Strategy 1 shows significant peaks, particularly around early 2019, 2020, and 2022. The volatility profile for strategy 2, on the other hand, is slightly different with major

peaks in 2018 and a huge spike in 2020, staying constant otherwise. Strategy 3 also has a very different volatility profile, having elevated spiked from 2018 to 2019, followed by flat rolling vol for the rest of the trading period.

This would lead us to believe that much of the volatility in the strategy stems from long gamma positions, again reinforcing that in general, a short gamma position is more likely to profit.

```
[24]: plot_pnl_distribution(combined_1, combined_2, combined_3)
```





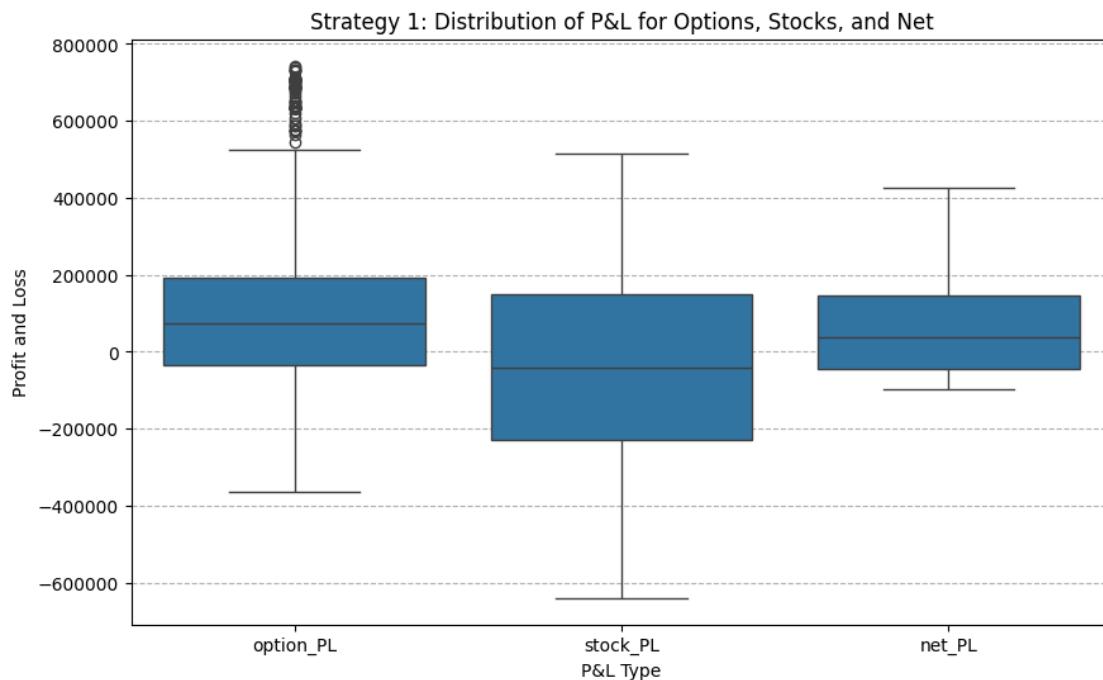
The profit and loss (P&L) distribution for Strategy 1 indicates that on a daily basis, returns are typically close to zero but with a slight skew towards losses, as shown by the sharp peak and longer left tail in the daily distribution plots. In contrast, the monthly P&L exhibits greater variability and a slight positive skew, indicating that while day-to-day results are often negligible or slightly negative, the aggregate monthly outcomes display a wider range of results with potential for profitability. This suggests that Strategy 1 experiences regular minor losses or gains daily but has the capacity for broader, more significant outcomes on a monthly basis, possibly due to cumulative effects of the trading strategy over a longer period.

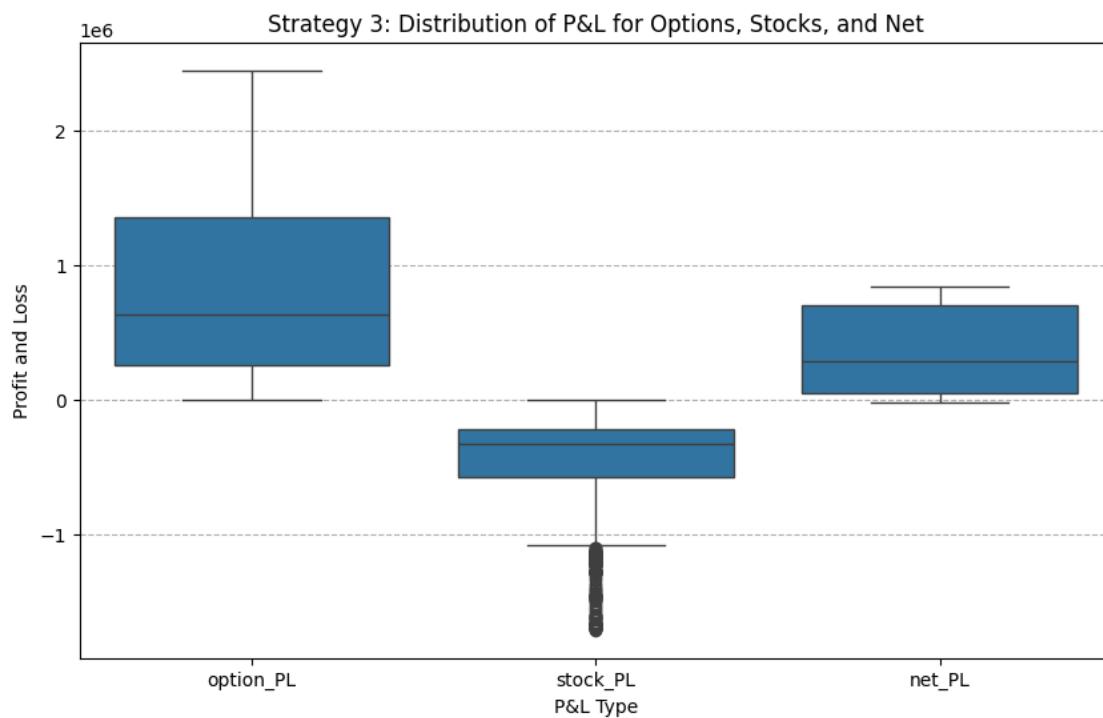
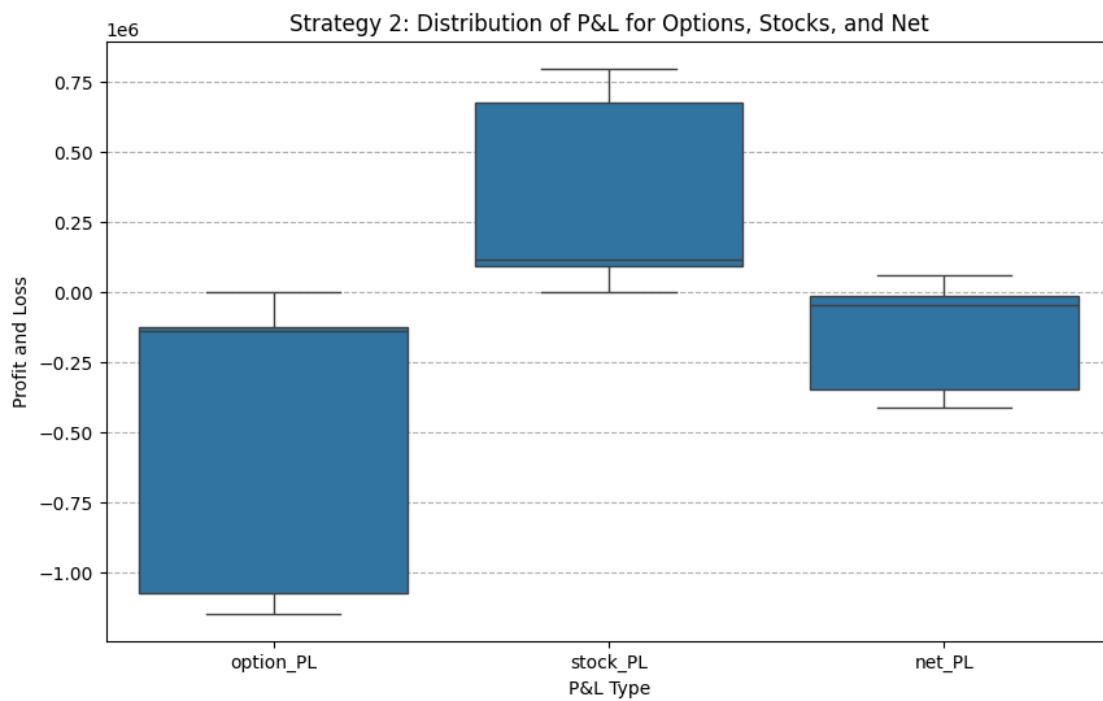
For Strategy 2, the daily P&L distribution and density plot show a very narrow, sharp peak around zero, indicating extremely consistent results on a day-to-day basis with minimal deviation from zero, suggesting very low daily volatility. In contrast, the monthly P&L distribution exhibits a wider range of outcomes with a significant peak at zero but with a notable spread, as seen in the histogram and density plot, suggesting that while daily returns are consistently close to neutral, the cumulative effect over the month can result in a wider range of profit and loss outcomes.

For Strategy 3, the daily P&L distribution again has a sharp peak around zero, indicating that most days result in minimal profit or loss, but with a visible spread indicating occurrences of both significant gains and losses. The density plot supports this with a narrow peak. The monthly P&L distribution and density show a more dispersed range than the daily figures, with a central peak slightly skewed towards the right, leaning towards profitability.

This mostly supports what we observed: short gamma positions generally have resulted in small profits. This is also shown below:

```
[25]: plot_pl_boxplots(combined_1, combined_2, combined_3)
```





The box plots above show the min, max, median, 1st, and 3rd quartiles of the option, stock, and net PL of all 3 trading strategies. These results were already analyzed with the performance summary above. Nevertheless, they show a visual representation of these statistical parameters. A majority of long option positions are unlikely to profit, which our analysis holds.

It is interesting to note that while the short-only portfolio was more profitable, the IV-comparison (model-free) strategy eliminated many severe outliers; however, this is a bit less interesting when one realizes that the outlying stock losses are generally correlated to strong option profits.

```
[26]: calculate_profit_factor_analysis(combined_1, combined_2, combined_3)
```

Strategy 1: Option Profit Factor: 1.315174804570438

Strategy 1: Stock Profit Factor: 0.7723735545069135

Strategy 2: Option Profit Factor: 0.11664217051956995

Strategy 2: Stock Profit Factor: 4.939689657031665

Strategy 3: Option Profit Factor: 3.6354558279313203

Strategy 3: Stock Profit Factor: 0.4134146298162746

The results show the profit factor for options and stocks across the 3 trading strategies. The profit factor is calculated as the total gains divided by the total losses. A profit factor greater than 1 indicates that the strategy is profitable (since total gains exceed total losses), while a profit factor less than 1 indicates a losing strategy.

Strategy 1: Option Profit Factor: 1.31517 - indicates that options trading is profitable, as gains exceed losses. Stock Profit Factor: 0.77237 - indicates that stock trading is not profitable, as losses exceed gains.

A mix of long/short positions is generally more stable, as we expect.

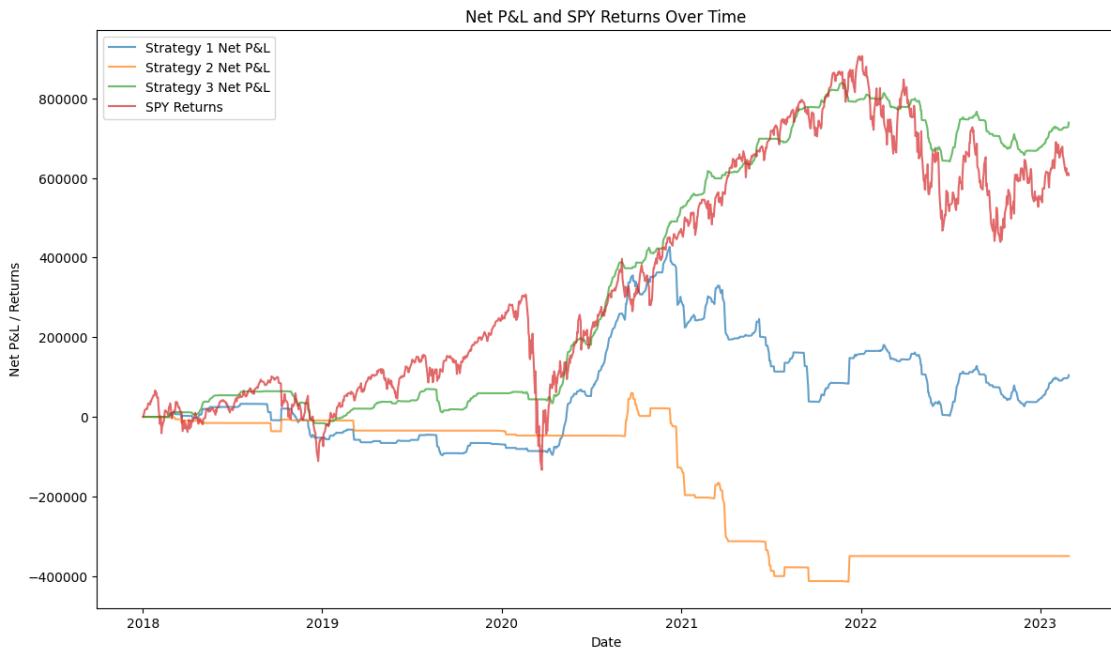
Strategy 2: Option Profit Factor: 0.11664 - indicates that options trading is highly unprofitable, with losses far exceeding gains. Stock Profit Factor: 4.93969 - indicates a highly profitable stock trading approach, with gains significantly outweighing losses.

Long option positions are likely to not be profitable, clearly shown here, where most of the recovered capital is from hedging.

Strategy 3: Option Profit Factor: 3.63546 - indicates that options trading is very profitable. Stock Profit Factor: 0.41341 - indicates that stock trading is not profitable, with losses outpacing gains.

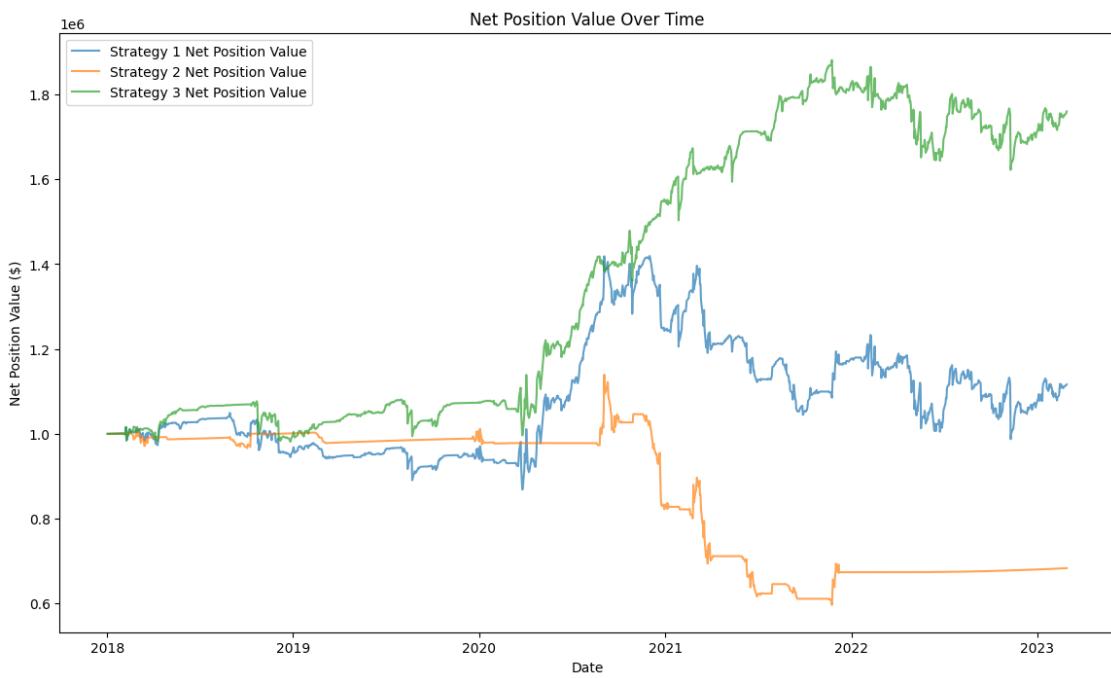
And a short gamma position is profitable as we expect.

```
[27]: plot_cum_net_pl_and_spy_returns(combined_1, combined_2, combined_3, spydata)
```



The plot above compares the returns for all 3 trading strategies along with excess returns on SPY on a \$1,000,000 initial capital.

[28]: `plot_net_pos_val(combined_1, combined_2, combined_3)`



The plot above displays the net position value over time for the 3 different trading strategies. Strategy 1 shows stability until early 2020, after which there's a rapid increase, peaking around mid-2021 / 2022 before it enters a period of volatility and slight decline, yet it remains significantly above its starting value. Strategy 2 maintains a relatively flat line with slight variations, indicating stability but minimal growth, followed by a steep decline towards the end of 2020, bottoming out and remaining flat. Strategy 3 experiences gradual growth until it peaks in early 2022, experiencing volatility towards the end.

We can also briefly visit the max drawdown metrics, which extends what we see above. We can see visually where these metrics occur:

```
[29]: long_short = pd.read_csv('simdata/PL_trade_1.csv')[['date','net_pos_value']].
    ↪sort_values(by='date').reset_index(drop=True)
long_only = pd.read_csv('simdata/PL_trade_2.csv')[['date','net_pos_value']].
    ↪sort_values(by='date').reset_index(drop=True)
short_only = pd.read_csv('simdata/PL_trade_3.csv')[['date','net_pos_value']].
    ↪sort_values(by='date').reset_index(drop=True)

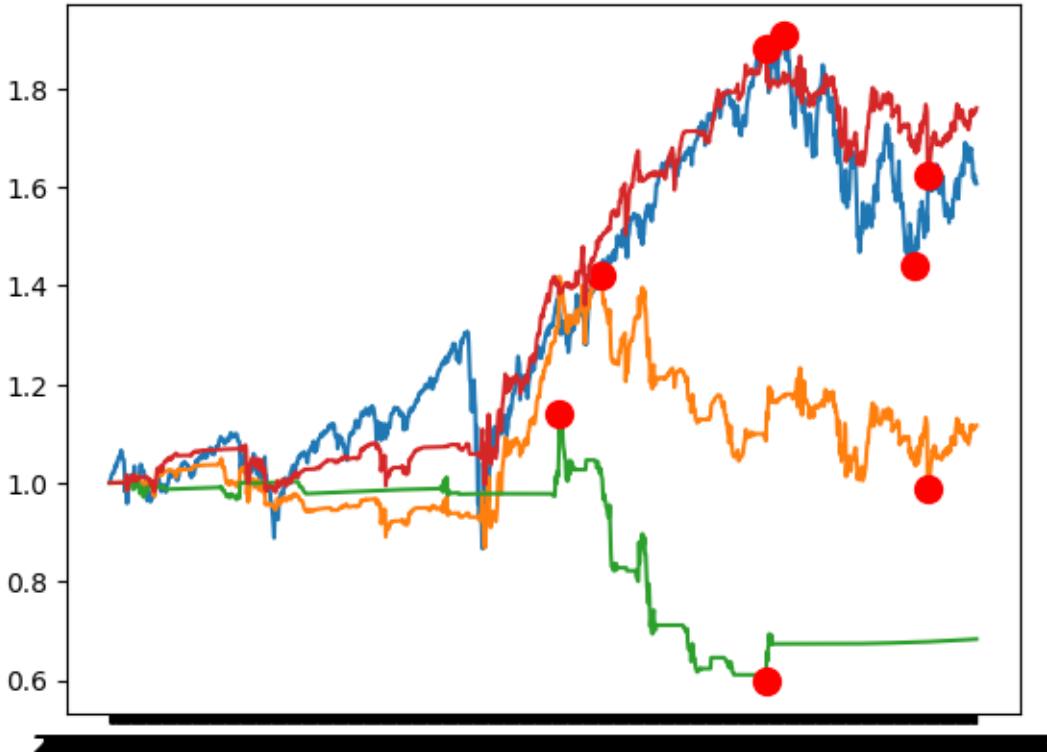
long_short['long_short_return'] = long_short['net_pos_value'] / long_short.
    ↪loc[0, 'net_pos_value']
long_only['long_only_return'] = long_only['net_pos_value'] / long_only.loc[0, 'net_pos_value']
short_only['short_only_return'] = short_only['net_pos_value'] / short_only.
    ↪loc[0, 'net_pos_value']
```

```
[30]: start_date = '2018-01-01'
end_date = '2023-02-28'
spy_data = pd.read_csv('spy_tickerdata.csv')[['date','adj_close']].
    ↪sort_values(by='date').reset_index(drop=True)
spy_data = spy_data.loc[(spy_data['date'] >= start_date) & (spy_data['date'] <= end_date)].copy().reset_index(drop=True)
spy_data['spy_return'] = spy_data['adj_close'] / spy_data.loc[0, 'adj_close']

ret_data = spy_data[['date', 'spy_return']].copy()
ret_data['long_short_return'] = long_short['long_short_return']
ret_data['long_only_return'] = long_only['long_only_return']
ret_data['short_only_return'] = short_only['short_only_return']
ret_data = ret_data.set_index('date')
```

```
[31]: for col in ret_data.columns:
    max_dd(ret_data[col])
```

```
0.24497270037076455
0.3042381586224645
0.4762755640185654
0.1375138413928244
```



Also interesting is that if we look at some summary stats, we note that our model-free IV-decided strategy, strategy 1, has a very low R-squared with the market, which is a good sign for separating it from the directional movement of the market. The Sharpe, however, is poor.

Strategy 2 (long gamma only) clearly does not perform well.

Strategy 3, in contrast, seems to be the best performer among the three, with a significant positive annualized return of approximately 11.6%. It is important to note that the annualized volatility is more than the return, however. It is interesting to note that despite being a direction-agnostic strategy, the beta and information ratio is quite high. Alpha is a bit better than the other two implementation.

Strategy 3 appears to be the most successful in terms of both absolute and risk-adjusted return.

```
[34]: combined_metrics_df = combined_performance_metrics(spydata, combined_1,
                                                       combined_2, combined_3, rfr=0.01)
combined_metrics_df
```

	Annualized Return	Annualized Volatility	Sharpe Ratio	Beta	\
Strategy 1	0.021625	0.186944	0.062186	0.221865	
Strategy 2	-0.071188	0.138066	-0.588035	-0.285525	
Strategy 3	0.115873	0.138542	0.764194	0.547836	

Information Ratio	Alpha	R-squared	Tracking Error
-------------------	-------	-----------	----------------

Strategy 1	0.034872	0.004964	0.086054	0.142349
Strategy 2	-0.421468	-0.029837	0.386690	0.070793
Strategy 3	0.386764	0.056731	0.350932	0.146681

1.12 10. Additional Concluding Thoughts

We've made most of our observations in the simulation and analysis sections, but we can give a few more thoughts:

- * As we expect, our simulation tends to support long option positions being generally unprofitable on their own
- * Because of options pricing IV higher than RV (due to high tail risk), we saw short option positions perform as expected
- * Our risk management decision to close a week before expiry really minimized the tail risk associated with short gamma, even through volatile times such as March 2020 and 2022 bear markets, keeping the maximum drawdown to 14%
- * Transaction costs are less impactful on the strategy due to daily rebalance outweighing costs, in general
- * Using model-free IV to decide both long and short positions reduced the overall profitability, but it can be argued that it also removed some outlying performers and underperformers. However, setting a threshold to trade only short gamma positions (and not take positions at all if the threshold is not met, rather than also having model-free IV thresholds on the other side) seems to still outperform.

1.13 Appendix

1.13.1 Simulate and Save

Full Simulation and saving to CSV of all possible position opens.

```
[ ]: simulations = create_simulations(options, data, dropna_greeks=True)

simulations_long = {date: calculate_realized_PL(df.copy(), long_op=True) for
    ↪date, df in simulations.items()}
simulations_short = {date: calculate_realized_PL(df.copy(), long_op=False) for
    ↪date, df in simulations.items()}

[ ]: os.makedirs('simdata', exist_ok=True)

for date, df_long in simulations_long.items():
    csv_path_long = f'simdata/simulation_long_{date}.csv'
    df_long.to_csv(csv_path_long, index=False)

for date, df_short in simulations_short.items():
    csv_path_short = f'simdata/simulation_short_{date}.csv'
    df_short.to_csv(csv_path_short, index=False)
```