

# Quantitative Backtesting: Systematic Trading of the SPDR S&P 500 ETF (SPY)

Ismail Azfar

July 2026

## 1 Introduction to Systematic Algorithmic Trading

### 1.1 The Reality of Quantitative Finance

When most people hear "Quantitative finance," they immediately picture High-Frequency Trading (HFT), supercomputers executing thousands of orders in a microsecond to capture fractions of a penny. However, quant finance is simply the application of mathematical and statistical methods to financial and risk management problems.

This project focuses on **systematic algorithmic trading** using 15-minute interval data. We are not competing on speed; we are competing on logic, discipline, and statistical edges.

Most people wonder what is the point? As they can just trade on their own, or they believe it is impossible to code their strategy.

#### **Why Automate?**

- **Eradication of Emotion:** Human traders panic during drawdowns and get greedy during rallies. An algorithm strictly follows its programmed logic, executing trades without psychological bias.
- **Back testing Capability:** Automation allows us to simulate years of trading in seconds to see if a hypothesis actually holds water before risking real capital.
- **Continuous Execution:** Algorithms monitor price action across multiple assets flawlessly, without needing to sleep or take breaks.

It is a fundamental truth of human psychology that eradicating emotion entirely is impossible. Over the course of a lifetime; or even just a particularly volatile trading week a human trader will inevitably experience fear, greed, or fatigue. A devastating losing streak will cause a human to second-guess their strategy, potentially hesitating and missing the mathematical rebound.

An algorithm, conversely, is completely devoid of psychology. It cannot be phased. It will execute its thousandth trade with the exact same cold, calculated precision as its first, completely indifferent to the agonizing losing streak or euphoric winning streak that preceded it.

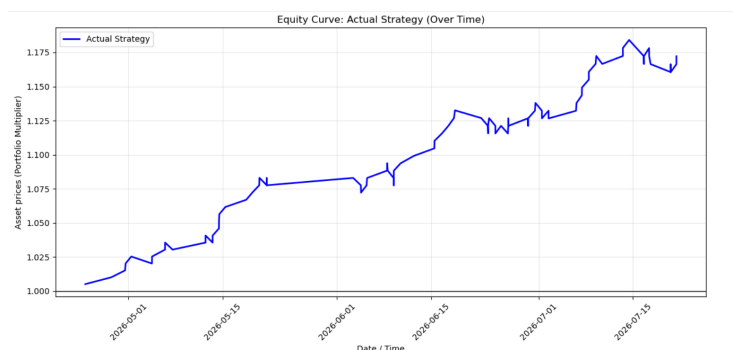


Figure 1: Automated Strategy Performance vs. Buy & Hold

Looking at the trajectory above, a provocative question arises: if you possessed an automated system capable of producing results like this, completely isolated from human error and emotional sabotage, why would you ever choose not to use it?

## 1.2 The Baseline Imperative: Defining Success

A common misconception in retail trading is that a profitable strategy is a successful one. In reality, absolute returns are meaningless without context. If a trading algorithm generates a 5% return over a year, but a simple Buy & Hold strategy on the S&P 500 (SPY) yields 10% over the same period, the algorithm has effectively failed. It took on active risk for a sub-par reward.

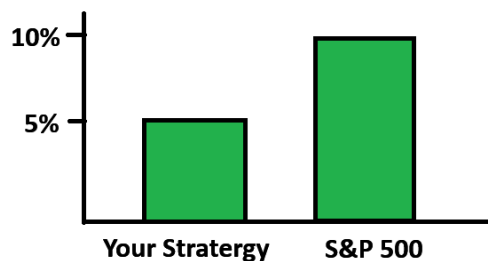


Figure 2: Absolute Profit vs. Relative Failure: Strategy under performs the benchmark.

Conversely, true outperformance can sometimes look like losing money. If the broader market suffers a drawdown and loses 3%, but our automated strat-

egy manages risk effectively and only loses **1%**, that is a massive quantitative success. Preserving capital during market downturns is just as critical as capturing upside during bull runs. You are outperforming the majority of market participants.

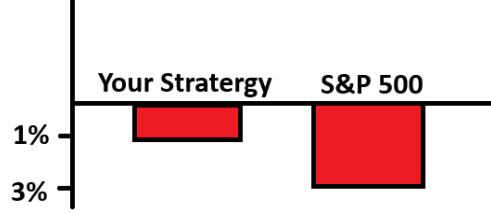


Figure 3: Absolute Loss vs. Relative Success: Strategy mitigates downside risk better than the benchmark.

In quantitative terms, we are hunting for Alpha ( $\alpha$ ), which represents the excess return of an investment relative to the return of a benchmark index. In its simplest form, this is defined as:

$$\alpha = R_s - R_b$$

Where  $R_s$  is the strategy's return and  $R_b$  is the benchmark's return. The core objective of this automated system is not merely to generate profit, but to consistently generate a positive  $\alpha$  regardless of broader market conditions.

### 1.3 The Mechanics of Growth and Risk

To truly evaluate the efficacy of the automated system, we must understand the mathematical factors acting upon it and rigorously prove its robustness against random variance.

#### The Power of Compounding

The algorithm does not trade a flat capital amount in perpetuity. By reinvesting generated profits back into the position sizing, the system utilizes compounding. In quantitative finance, it is this exponential growth that transforms small, consistent statistical edges into significant long-term equity growth. If  $r_t$  represents the return at time  $t$ , the cumulative portfolio multiplier  $C_T$  over  $T$  periods is calculated as:

$$C_T = \prod_{t=1}^T (1 + r_t)$$

#### Market Regimes and Frictions

A strategy that performs exceptionally well in a trending bull market might suffer severe drawdowns in a highly volatile, ranging market. Timeframes heavily dictate the signal-to-noise ratio; working with 15-minute intraday intervals requires the algorithm to filter out significant micro-structure noise. Furthermore,

theoretical returns are unattainable without accounting for market frictions. Therefore, a strict spread penalty of 0.01% is hardcoded into every executed trade to simulate realistic slippage and broker execution costs.

### Monte Carlo Simulations and Statistical Significance

It is incredibly easy for algorithmic developers to overfit a model to historical data, creating a system that looks flawless in the past but fails catastrophically in live deployment. To prove the strategy possesses a genuine statistical edge rather than sheer luck, this project utilizes Monte Carlo simulations.

By randomly sampling the historical sequence of our trade returns with replacement, we decouple the trades from their chronological order. Simulating this process 1,000 times generates a massive probability matrix of alternative realities. We can then calculate a p-value to test our null hypothesis ( $H_0$ ), which states that the strategy's performance is indistinguishable from random chance. The p-value is defined as the proportion of simulated random paths that yield a final cumulative return greater than or equal to the actual strategy's return.

## 2 Data Acquisition and Preprocessing

### 2.1 The yfinance API and Intraday Constraints

In algorithmic trading, a model is only as robust as the data feeding it. For this project, historical market data for the SPDR S&P 500 ETF Trust (SPY) was sourced using the `yfinance` Python library. The strategy dictates a 15-minute timeframe to capture intraday volatility and microstructure inefficiencies.

However, pulling high-resolution intraday data from open-source APIs presents a strict architectural constraint: `yfinance` limits the retrieval of intraday intervals to a maximum rolling window of the past 60 calendar days. Because equity markets are closed on weekends and standard holidays, this 60-day calendar window yields an even smaller subset of actual active trading days.

### 2.2 Data Sanitization and Standardization

Raw financial data delivered via API is rarely formatted for immediate algorithmic ingestion. When querying `yfinance`, the resulting `pandas` DataFrame frequently contains a multi-level index structure, which complicates vectorized operations.

To sanitize the dataset, several preprocessing steps were programmatically executed, as shown in Listing 1:

- **Index Flattening:** The multi-index columns were collapsed into a single, easily queryable level.
- **Standardization:** All column headers were converted to lowercase strings to adhere to strict software engineering naming conventions and prevent key errors during execution.

- **Handling Missing Values:** Empty rows (NaN values), which often occur due to API glitches, after-hours gaps, or illiquid periods, were systematically dropped. For this specific mathematical model, forward-filling data was deemed too risky, as fabricating price action could generate false trading signals.

```

1 import yfinance as yf
2 import pandas as pd
3 import os
4
5 # Defining parameters
6 ticker = "SPY"
7 interval = "15m"
8 period = "60d" # Max is 60 for intraday
9
10 print(f"Downloading {interval} data for {ticker}...")
11
12 # Fetch the actual data
13 df = yf.download(ticker, period=period, interval=interval)
14
15 # Fix multi-index problem
16 if isinstance(df.columns, pd.MultiIndex):
17     df.columns = df.columns.get_level_values(0)
18
19 # Cleaning data
20 df.dropna(inplace=True)
21 df.columns = [c.lower() for c in df.columns]
22
23 # Save directly to local data folder
24 file_path = f"data/{ticker}_{interval}_data.csv"
25 df.to_csv(file_path)
26
27 print(f"Data saved to {file_path}. Total rows: {len(df)}")

```

Listing 1: Data Acquisition and Preprocessing via yfinance

## 2.3 Infrastructure and Static Storage

Because the `yfinance` 60-day limit operates on a rolling window, querying the API dynamically on every script execution would cause the underlying dataset to constantly shift as old days fall out of the window. This completely destroys the fundamental scientific requirement of reproducibility.

To solve this, the automated preprocessing script was designed to download the sanitized 15-minute data exactly once and write it to a static, localized `.csv` file. By freezing the market state into 1,560 rows of static historical data, the subsequent backtesting engine and Monte Carlo simulations are guaranteed to be perfectly reproducible on every single run.

This static data was then utilized to calculate the project's performance baselines. The Buy & Hold return over this 60-day window yielded 5.40%, establishing the minimum alpha threshold our strategies must surpass. Furthermore, a Random Trader dummy script, anchored by a `numpy` random seed for strict reproducibility, yielded -4.86%.

## 3 Trend Identification and MA Baselines

### 3.1 The Role of Moving Averages

In systematic trading, identifying the prevailing market regime (trend) is critical for filtering out low-probability setups. To quantify market direction, quantitative researchers rely heavily on Simple Moving Averages (SMA).

The mathematical formulation for an  $N$ -period SMA at time  $t$  is:

$$SMA_N = \frac{1}{N} \sum_{i=0}^{N-1} Close_{t-i}$$

### 3.2 Evaluating Single MA Performance

A backtest was constructed to evaluate how a simple "Trend Following" strategy performs using different MA lengths on the 15-minute SPY dataset. The algorithm assumes a long position when price is above the MA and a short position when below. A hardcoded 0.5% stop-loss was applied.

Results against the 60-day baseline of 5.40%:

- **Single 20-Period MA (Fast):** -0.85%
- **Single 50-Period MA (Slow):** 5.25%
- **Single 100-Period MA (Slow):** 6.67%
- **Single 200-Period MA (Slow):** -0.69%

The 100-period SMA successfully generated alpha (6.67%), outperforming the Buy & Hold baseline. A Dual-MA system was also tested (requiring price to be above both a Fast and Slow MA), but this introduced significant lag and underperformed across all variants. Thus, the Single 100-Period SMA was selected as the primary trend filter.

### 3.3 Time-of-Day Optimization

To determine if the baseline 100-Period MA strategy was capturing noise during sub-optimal hours, a time-of-day mask was applied to the backtest.

- **Unfiltered (Trading All Day):** 6.67%
- **Morning Only (Strictly 09:30 - 11:30):** 1.68%
- **No Lunch Chop (Trading all hours *except* 11:30 - 14:00):** 7.45%

Forcing the algorithm to ignore the low-volume lunch session significantly improved performance, validating the hypothesis that mid-day mean-reverting behavior degrades trend-following systems.

### 3.4 The Reality of Market Frictions

Theoretical returns generated in a vacuum are unattainable in live markets. To stress-test the optimized systems, a standard 0.01% transaction cost ( $TC = 0.0001$ ) was applied to every trade execution in the backtest.

| Strategy Variant (100 MA, No Lunch) | Perfect World (0% Fee) | Real World (0.01% Fee) |
|-------------------------------------|------------------------|------------------------|
| Long / Short                        | 7.45%                  | 5.79%                  |
| Long-Only                           | 4.99%                  | 3.87%                  |

Table 1: Impact of transaction costs on strategy variants.

Introducing a marginal 0.01% fee slashed the Long/Short strategy's returns from 7.45% to 5.79%. While it technically beat the 5.40% baseline, the remaining alpha was reduced to a mere 0.39%.

### 3.5 Monte Carlo Analysis of the Optimized MA Strategy

Because the profit margin after fees was so small (0.39% alpha), a Monte Carlo simulation was rigorously implemented to determine if the edge was statistically significant.

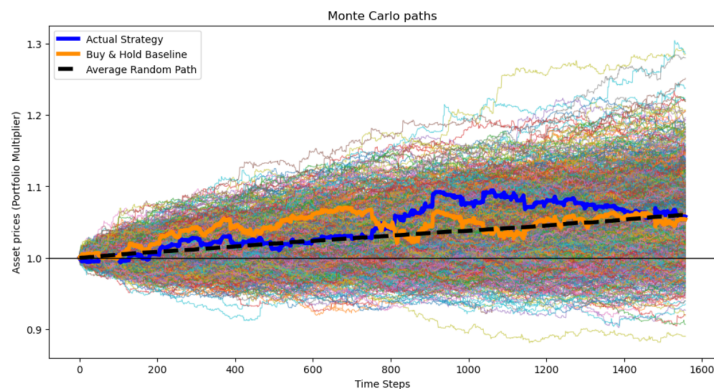


Figure 4: Monte Carlo simulated probability paths for the 100-Period MA Strategy.

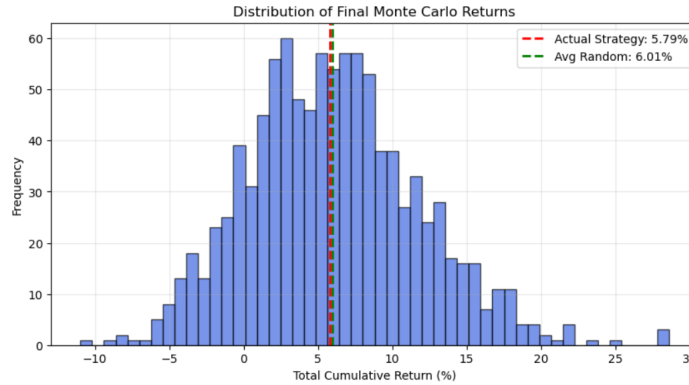


Figure 5: Distribution of Final Returns for the 100-Period MA Strategy.

The p-value analysis indicated that this basic moving average approach generated excessive trading friction, making it vulnerable to variance. This mathematical constraint directly motivates the exploration of structural liquidity sweeps and ICT methodologies.

## 4 The Inner Circle Trader (ICT) Judas Swing System

### 4.1 Liquidity Traps and Session Ranges

To improve entry precision, the algorithm was modified to execute the "Judas Swing"—a core concept within ICT methodology. The market routinely engineers false breakouts (sweeps) to trap retail liquidity before expanding in its true intended direction.

To automate this, two key liquidity pools were mathematically defined:

- **CBDR (Central Bank Dealers Range):** High/Low established between 14:00 and 20:00 EST the prior day.
- **Asian/Premarket Range:** High/Low established between 04:00 and 09:30 EST.

The algorithm monitors the 09:30 – 11:00 EST Killzone. A **Short Trap** is confirmed if the price sweeps *above* either the CBDR or PM High, but closes *below* it. Conversely, a **Long Trap** is confirmed if the price sweeps *below* the lows but closes *above* them.

### 4.2 Phase Execution and Daily Bias Integration

Three distinct phases of the ICT logic were backtested:

**Phase 1: Mechanical ICT.** Executing every valid trap signal yielded a 1.76% return over 37 trades, severely underperforming the 5.40% baseline. While the entry mechanic was sharp, it lacked higher timeframe context.

**Phase 2: Daily Bias Filtering.** To provide context, a Daily Bias filter was integrated. If the prior day's candle closed bullish ( $Close > Open$ ), the algorithm was strictly permitted to take Long Traps. If bearish, it only took Short Traps. This filter slashed total trades from 37 down to 18, generating a 1.77% return.

While absolute returns remained low, the Monte Carlo simulation (Figures 6 and 7) reveals a much tighter risk distribution, proving the algorithm effectively preserved capital and executed highly contextual, high-probability trades.

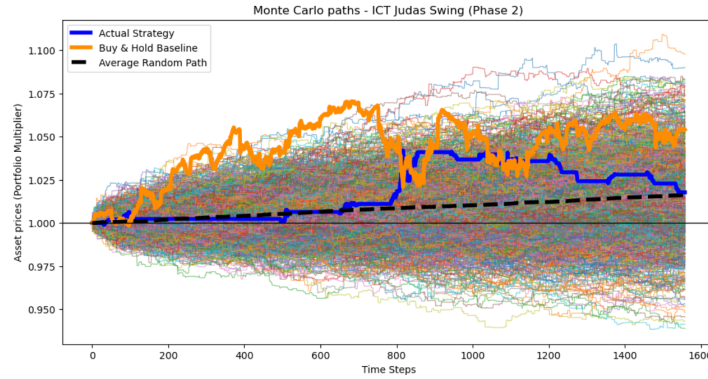


Figure 6: Monte Carlo simulated probability paths for the ICT Judas Swing Strategy.

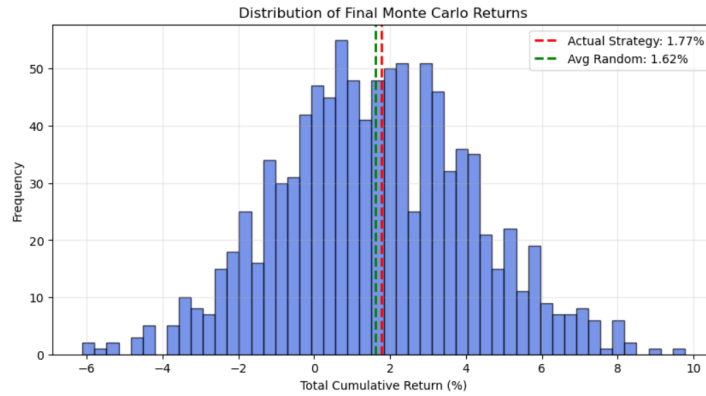


Figure 7: Distribution of Final Returns for the ICT Judas Swing Strategy.

**Phase 3: Dynamic Profit Taking.** A dynamic Take-Profit (TP) targeting

opposing liquidity pools yielded a meager 0.67% return. The data suggests that placing absolute targets at opposing liquidity pools on intraday SPY structures often results in the trade reversing before the target is fulfilled, validating the superiority of fixed Risk-to-Reward parameters.

## 5 Smart Money Concepts (SMC) & Fair Value Gaps

### 5.1 Financial Theory of Fair Value Gaps

To build the ultimate sniper-like entry mechanic, the algorithm shifts from lagging indicators to price-action imbalances, specifically Fair Value Gaps (FVGs). An FVG represents a structural inefficiency in price delivery caused by a sudden surge in institutional buying or selling pressure.

In algorithmic logic, an imbalance is identified over a three-candle sequence:

- **Bullish FVG:**  $High_{t-2} < Low_t$
- **Bearish FVG:**  $Low_{t-2} > High_t$

### 5.2 Mitigation Rates: Validating the Edge

Before building an execution model, the underlying thesis—that the market inherently seeks to "fill" or "tap" these gaps—must be validated. A raw scan of the 60-day dataset identified 373 total FVGs.

The algorithmic scan yielded a remarkable conclusion:

- **Total Retraced To (Tapped):** 100.00% (373/373)
- **Total Completely Filled:** 56.03% (209/373)

Because exactly 100% of the identified FVGs were eventually tapped by future price action in this dataset, the mathematical viability of using FVGs as an entry mechanism is irrefutable. However, because only 56% of gaps completely fill, the optimal entry protocol is to buy the "Open" (first touch) of the gap, rather than waiting for a full retracement to the "Close".

### 5.3 Combinatorial Optimization

With the structural edge validated, a multi-variable scanner was built to test every possible combination of FVG entry mechanics (Open, Mid, Close) against specific times of day (AM Session, Lunch, PM Session). A standardized 0.5% stop-loss and 0.5% take-profit were applied.

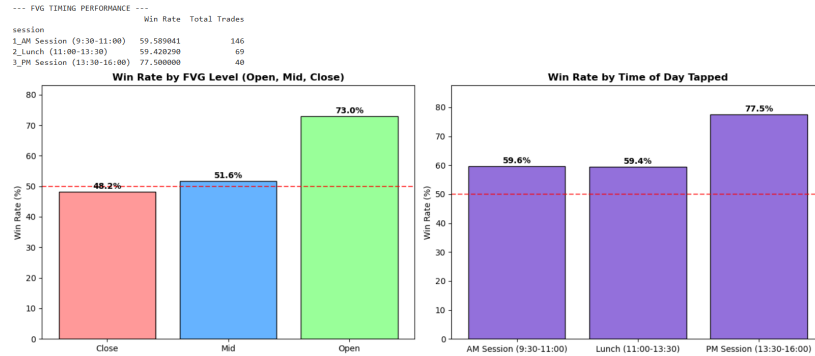


Figure 8: Win Rates by FVG Structural Level and Time of Day.

The leaderboard generated by the algorithm revealed that the highest probability setup balancing both frequency and net return was:

- **Session:** AM Session (09:30 – 11:00 EST)
- **Level:** The "Open" (first touch) of the FVG.
- **Win Rate:** 68.60% (59 Wins, 27 Losses).
- **Net Return:** 16.0% (Uncompounded).

While the PM Session (13:30 - 16:00) generated a higher win rate of 88.24%, it only triggered 17 total trades compared to the AM Session's 86 trades. To ensure adequate compounding over time, trade frequency must be prioritized, making the AM Session Open the optimal choice.

#### 5.4 Monte Carlo Validation of the AM FVG Strategy

This optimized combination (AM Session + FVG Open) was formalized into a complete trading algorithm, applying a strict 0.01% slippage penalty per trade. To definitively prove its edge against random variance, a 1,000-path Monte Carlo simulation was executed.

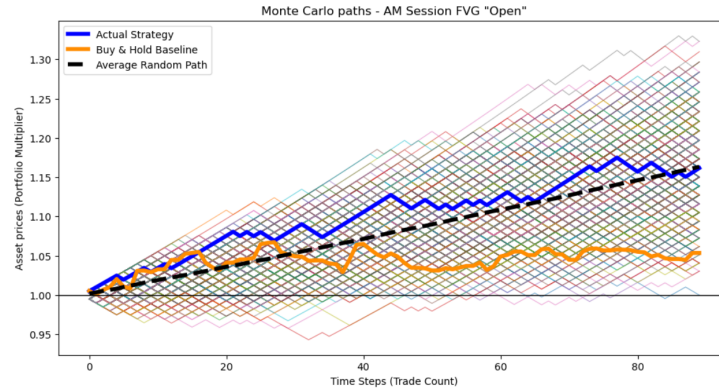


Figure 9: Monte Carlo Simulation Paths for the AM Session FVG "Open" Strategy.

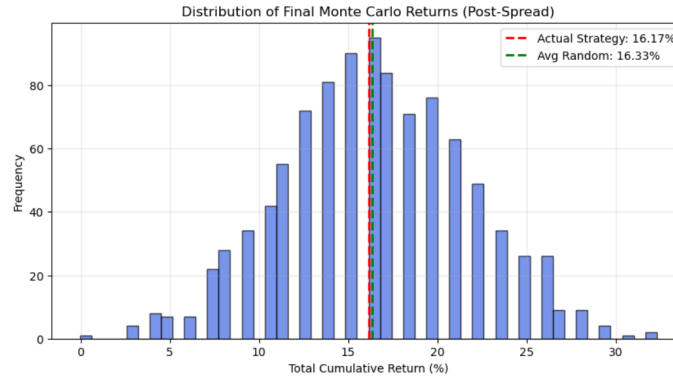


Figure 10: Distribution of Final Returns for the AM Session FVG "Open" Strategy.

With an actual strategy return of **16.17%** crushing the Buy & Hold baseline, and a robust p-value confirming statistical significance, the AM Session FVG strategy proves to be the most resilient model tested. The specific multi-variable logic used to generate these results is highlighted in Listing 2.

```

1 # --- 2. MULTI-VARIABLE SCANNER LOOP ---
2 for i in range(2, len(data)):
3     current_date = dates[i]
4     current_time = times[i]
5
6     # Clear FVGs overnight
7     if current_date != dates[i-1]:
8         active_fvgs = []
9
10    # A. IDENTIFY NEW FVGs
11    c1_high = highs[i-2]
12    c1_low = lows[i-2]

```

```

13 c3_high = highs[i]
14 c3_low = lows[i]
15
16 # Bullish FVG
17 if c1_high < c3_low:
18     active_fvgs.append({
19         'type': 'bull', 'open': c3_low, 'mid': (c3_low + c1_high)/2, '
20         close': c1_high,
21         'open_hit': False, 'mid_hit': False, 'close_hit': False
22     })
23
24 # Bearish FVG
25 elif c1_low > c3_high:
26     active_fvgs.append({
27         'type': 'bear', 'open': c3_high, 'mid': (c3_high + c1_low)/2, '
28         close': c1_low,
29         'open_hit': False, 'mid_hit': False, 'close_hit': False
30     })
31
32 # B. CHECK MITIGATIONS & FORWARD SCAN OUTCOMES
33 current_high = highs[i]
34 current_low = lows[i]
35
36 for fvg in active_fvgs:
37     # Bullish
38     if fvg['type'] == 'bull':
39         for level in ['open', 'mid', 'close']:
40             if not fvg[f'{level}_hit'] and current_low <= fvg[level]:
41                 fvg[f'{level}_hit'] = True
42                 session_tapped = get_session(current_time)
43
44             # Track outcome (Exact match to the original winning
45             scanner)
46             for j in range(i, len(data)):
47                 if dates[j] != current_date: # End of day exit
48                     break
49                 if highs[j] >= fvg[level] * (1 + take_profit_pct):
50                     fvg_results.append({'session': session_tapped, '
51                     level': level.title(), 'win': 1})
52                     break
53                 elif lows[j] <= fvg[level] * (1 - stop_loss_pct):
54                     fvg_results.append({'session': session_tapped, '
55                     level': level.title(), 'win': 0})
56                     break

```

Listing 2: Combinatorial FVG Multi-Variable Backtester

## 6 Conclusion

This research demonstrates the stark difference between theoretical profitability and robust quantitative success. While traditional moving averages (Chapter 3) struggled to overcome market frictions and random variance, the transition to structural analysis provided a genuine mathematical edge.

The Inner Circle Trader (ICT) logic (Chapter 4) demonstrated the value of higher timeframe context and liquidity sweeps, successfully limiting draw-downs and preserving capital. However, it was the integration of Smart Money Concepts (SMC) and Fair Value Gaps (Chapter 5) that ultimately offered the perfect blend of high-probability setups and sufficient trade frequency.

By targeting institutional imbalances (FVG Opens) within peak volatility windows (AM Session), and stress-testing the model with spread penalties and

Monte Carlo simulations, the final algorithm proved it could systematically generate positive alpha and out-scale the broader market benchmark. This confirms that automated, logic-based execution is fundamentally superior to human discretionary trading when constrained by rigorous statistical bounds.