Synthetic Market Data Generator

IE 421: High Frequency Trading Tech (Professor David Lariviere)

Fall 2024 • University of Illinois at Urbana-Champaign

1. The Team

- **Name:** Sai Dasari (saisd2)
    - **Email:** saisd2@illinois.edu
    - **Major:** Computer Science
    - **Role:** Backtesting Developer
    - **Relevant Experience:**
        - ML Research @ Capital One
    - **Top 3 Languages:** Python, C++, Java
- **Name:** Aryan Gosaliya (aryanag2)
    - **Email:** aryanag2@illinois.edu
    - **Major:** Computer Engineering
    - **Role:** Level 2 Data Order Book Data Developer
    - **Relevant Experience:**
        - Software Engineer Intern @ Oracle Cloud Infrastructure
        - Researcher @ Coordinated Science Laboratory
    - **Top 3 Languages:** Python, C, JavaScript
- **Name:** Arhan Goyal (arhang2)
    - **Email:** goyalarhan@gmail.com
    - **Major:** Computer Engineering
    - **Role:** Data Generator Models Developer and QA
    - **Relevant Experience:**
        - Trading platform: Created liquidity injector for CME Group's trader certification markets
        - Time series forecasting: Deep learning-based time series forecasting for client Swedavia (owner, Stockholm Airport)
    - **Top 3 Languages:** Python, Java, C
- **Name:** Rutva Pandya (rutvadp2)
    - **Email:** rutvadp2@illinois.edu

  - **Major:** Computer Engineering
  - **Role:** Backtesting Developer
  - **Relevant Experience:**
    - New to High Frequency Trading
  - **Top 3 Languages:** Python, C++, C

---

2. Introduction

Project Overview

In today's fast-paced financial markets, the ability to simulate realistic stock price movements is crucial for testing trading strategies and algorithms before they are applied to real-world trading. Our project focuses on creating a **synthetic stock market data generator** that mirrors the behavior of real financial markets. This tool can simulate stock prices under various market conditions—ranging from high-volatility jumps to stable bull or bear trends—providing a safe, controlled environment for traders, researchers, and analysts to evaluate their strategies without risking real capital.

Objectives

1. **Simulating Stock Price Data**

   Generate realistic stock price movements using advanced mathematical models (Heston, Jump Diffusion, Regime Switching, Variance Gamma) that mimic real-world market behaviors—such as trends, volatility, and sudden price jumps.

2. **Generating Order Book Data**

   Produce detailed order book data, capturing bid/ask quantities and market depth over time. This helps replicate real trading environments.

3. **User-Friendly Visualization**

   Offer a web-based graphical interface (via Streamlit) for visualizing order book changes and price trends.

4. **Enabling Backtesting of Trading Strategies**

   Integrate a backtesting framework that allows users to evaluate the performance of trading algorithms against simulated market data.

5. **Customizable Market Conditions**

   Provide multiple models with adjustable parameters, including volatility, drift, market regimes, and more, to simulate diverse scenarios (e.g., bull, bear, high volatility).

Methodology

Our project employs a **modular approach** to simulate realistic market behaviors:

- **Mathematical Models:** Implemented in Python, each model allows customization of key parameters (volatility, drift, market regimes).

- **Command-Line Arguments:** Facilitate easy configuration of simulations, such as `-model`, `-S0`, `-mu`, etc.
- **Data Export & Visualization:** Outputs data as CSV files, which are then processed for interactive visualization or backtesting.
- **Order Book Simulation:** Maintains aggregated bids and asks at multiple price levels, capturing realistic liquidity conditions.

3. Features and Limitations

Key Features

1. **Multiple Simulation Models**
    - **Heston Model:** Simulates price volatility and mean-reversion dynamics.
    - **Jump Diffusion Model:** Captures sudden price jumps from unexpected events.
    - **Regime Switching Model:** Reflects shifts between distinct market regimes (bull/bear).
    - **Variance Gamma Model:** Produces stock price paths with heavier tails (kurtosis).
2. **Order Book Data**

    Generates **detailed bid/ask levels**, bid-ask spreads, and market depth to mimic real-world trading environments.
3. **Customizable Parameters**

    Allows users to adjust **volatility**, **market trends**, **simulation duration**, and more to replicate diverse market scenarios.
4. **Interactive Visualization**

    Leverages a **Streamlit** application for a web-based interface that visualizes price movements and order book changes over time.
5. **Backtesting Support**

    Integrates with a backtesting framework so users can **evaluate the performance** of trading strategies using simulated market data.
    - **Market Maker:** Maintains bid/ask quotes for liquidity, profiting from the spread while managing inventory.
    - **Position Taker:** Takes directional positions based on momentum signals, employing stop-loss and take-profit levels.

Use Cases

- **Quick, Free Access to Market Data**
  Model any market condition and generate large volumes of synthetic data without acquisition costs.

- **Strategy Testing**
  Evaluate algorithmic strategies under risk-free, simulated conditions.
- **Educational Tools**
  Provide hands-on learning opportunities in a controlled environment for students and researchers.
- **Market Analysis**
  Simulate various market scenarios (high volatility, regime changes) to study their impact on trading behaviors.

Limitations

- **Simplified Assumptions**
  Models rely on mathematical approximations and may not fully capture real-world market complexity.
- **Parameter Sensitivity**
  Accuracy depends on the
  quality and realism of user-provided input parameters.
- **Static Simulations**
  Generated data reflects
  predetermined scenarios and does not adapt to live market events.

4. User Guide

Interacting with the Project

1. **Download the Repository**

   Clone via Git or download the ZIP from the repository:

   ```
   git clone https://gitlab.engr.illinois.edu/ie421_high_frequency_trading_fall_2024/ie421_hft_fall_2024_group_09/group_09_project.git
   ```

2. **Open the Repository**

   Open the folder in a code editor (e.g., Visual Studio Code). Access the terminal (Ctrl+Shift+`\`) to run commands.

OrderBook Module

The **OrderBook** module emulates real-world financial order books, aggregating bids and asks across price levels:

- **Add/Remove Bids/Asks**
  Reflect changes in market depth over time.
- **Retrieve Best Bid/Ask & Spread**
  Crucial for understanding immediate execution prices and liquidity.

- **Get Market Depth**
  View top levels of the order book for a quick snapshot of market state.

IntegratedDataGenerator.py

IntegratedDataGenerator.py orchestrates the simulation process:

- **Multiple Models:** Heston, Jump Diffusion, Regime Switching, Variance Gamma.
- **Order Book Setup:** Initializes an order book around a specified initial stock price.
- **Simulation Orchestration:** Runs the chosen model, updating the order book at each time step.
- **Output:** Saves simulation results (price paths, order book states) in CSV format in the `simulation_output` directory.

simulator.py

simulator.py is the main interface for executing the **synthetic market data simulations**:

- **Usage Example (Heston Model):**

```
python -m simulator.simulator --model heston --S0 100 --V0 0.04 --mu 0.05 --kappa 2.5 --theta 0.04 --sigma_v 0.3 --rho -0.5 --dt 0.003968 --T 1 --tick_size 0.01 --initial_depth 5 --max_volume 50 --price_step 0.01 --spread_limit 0.05 --depth_levels 5
```

- **Configurable Parameters:** drift (`-mu`), volatility (`-sigma` or `-sigma_v`), time steps (`-dt`), total simulation time (`-T`), and order book details (`-initial_depth`, `-price_step`, etc.).

Upon completion, the script stores all simulated data in the `simulation_output` directory, ready for visualization or backtesting.

Available Input Parameters

Below is an abbreviated summary of key parameters and their defaults:

Parameter	Type	Default	Description	Models
`--model`	`str`	*(required)*	Simulation model to use: heston, jumpdiffusion, regimeswitching, variancegamma	All models
`--S0`	`float`	100.0	Initial stock price	All models
`--V0`	`float`	0.04	Initial variance (Heston only)	heston
`--mu`	`float`	0.05	Drift or market trend	All models
`--kappa`	`float`	1.5	Mean reversion speed (Heston only)	heston
`--theta`	`float`	0.04	Long-term variance mean (Heston only)	heston

`--sigma_v`	`float`	0.3	Volatility of volatility (Heston only)	heston
`--rho`	`float`	-0.5	Correlation between price & variance (Heston only)	heston
`--sigma`	`float`	0.2	Stock price volatility (Jump Diffusion, Variance Gamma)	jumpdiffusion, variancegamma
`--lambda_jump`	`float`	0.1	Jump intensity (Jump Diffusion only)	jumpdiffusion
`--jump_mean`	`float`	0.0	Mean jump size (Jump Diffusion only)	jumpdiffusion
`--jump_std`	`float`	0.02	Jump size std (Jump Diffusion only)	jumpdiffusion
`--nu`	`float`	0.1	Variance rate (Variance Gamma only)	variancegamma
`--dt`	`float`	0.003968	Time step size in years (1/252 ~ daily steps)	All models
`--T`	`float`	1.0	Total simulation time in years	All models
`--regimes`	`json`	`'{"bull": {...}, "bear": {...}}'`	Market regimes & parameters (Regime Switching only)	regimeswitching
`--transition_matrix`	`json`	`'[[0.9, 0.1],[0.2, 0.8]]'`	Probability of switching between regimes	regimeswitching
`--initial_depth`	`int`	5	Levels on each side of the order book	All models
`--max_volume`	`float`	100.0	Maximum volume for orders	All models
`--price_step`	`float`	0.01	Price increment for bids/asks	All models
`--spread_limit`	`float`	0.05	Max distance to remove stale orders	All models
`--depth_levels`	`int`	5	Number of order book levels to simulate	All models
`--tick_size`	`float`	0.01	Tick size for price updates	All models

How to Choose a Model

Selecting the right model depends on the **market behavior** you aim to replicate:

- **Heston Model**
  Perfect for capturing **volatility** and smooth fluctuations with **mean-reversion**.
- **Jump Diffusion Model**
  Ideal for simulating markets with **unpredictable price jumps** (e.g., news shocks).

- **Regime Switching Model**
  Best for
  bull/bear markets or frequent regime transitions.
- **Variance Gamma Model**
  Suitable for
  fat tails and **asymmetrical** price distributions.

Order Book Simulation Application

The **Order Book Simulation** is an interactive **Streamlit** application designed to visualize and analyze synthetic order book data generated by the **IntegratedDataGenerator**. By uploading a simulation CSV file, users can inspect real-time order book states (including multiple bid/ask levels) at different time steps. The application leverages **Plotly** for dynamic visualizations, calculates the **bid-ask spread**, and displays crucial metrics like **current price** and **variance** (if applicable). To run the application:

```
streamlit run simulation/order_book_simulation.py
```

Open the generated link in your web browser, upload the CSV file from `simulation_output`, and use the slider to explore different points in the simulation.

Process Simulation Data with `CleanCSV.py`

The **CleanCSV.py** script is designed to streamline the preparation of raw simulation CSV files. It:

- **Removes Unnecessary Columns** (e.g., `BidAskSpread` if not needed for certain backtesting tools).
- **Converts `Time` to a Standard `DateTime`** format based on user inputs.
- **Applies a Common Simulation Period** to all files or prompts for individual periods for each CSV file.
- **Saves Processed Files** with a `process_simulation_output_` prefix in the `simulation_output` directory, ensuring easy integration into analytical workflows.

Run the script:

```
python CleanCSV.py
```

1. **Select Common Period or Individual Periods:**
   - If applying a single period to all files, you will input a start and end date/time once.
   - If specifying per file, you will provide unique dates for each CSV.
2. **Processed Output:**
   - The script saves cleaned data as `process_simulation_output_<model_name>.csv`, ensuring consistent naming and readiness for backtesting.

Backtester & Market Participants

Market Participants Simulation Project & **L2 Orderbook Backtester** are two primary components enabling robust testing of **trading strategies**:

1. **Market Participants Simulation:**
    - **Traders**:
        - **Market Maker**: Profits from spreads, maintains bid/ask quotes.
        - **Position Taker**: Momentum or directional strategies, uses stop-loss/take-profit orders.
    - **Execution**:
        - Configure participant settings in `market_participants/README.md`.
        - Run with `python tests/test_traders.py` in `market_participants_project/`.
2. **L2 Orderbook Backtester**:
    - **Data Input**: Accepts L2 CSV format with multiple bid/ask levels.
    - **Performance Metrics**: Generates results with key metrics (Sharpe ratio, max drawdown, etc.).
    - **Usage**:

      ```
      python run_backtest.py
      ```

    - **Results**: Summaries and analytics stored in the `backtester` directory, enabling deeper strategy evaluation.

---

5. Results

Models

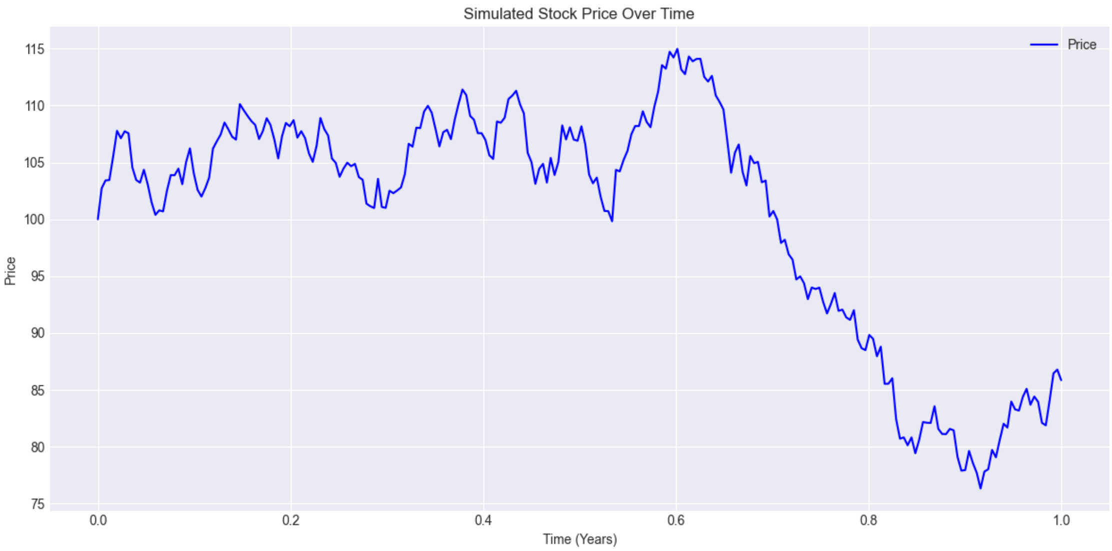
Heston Model CSV Example

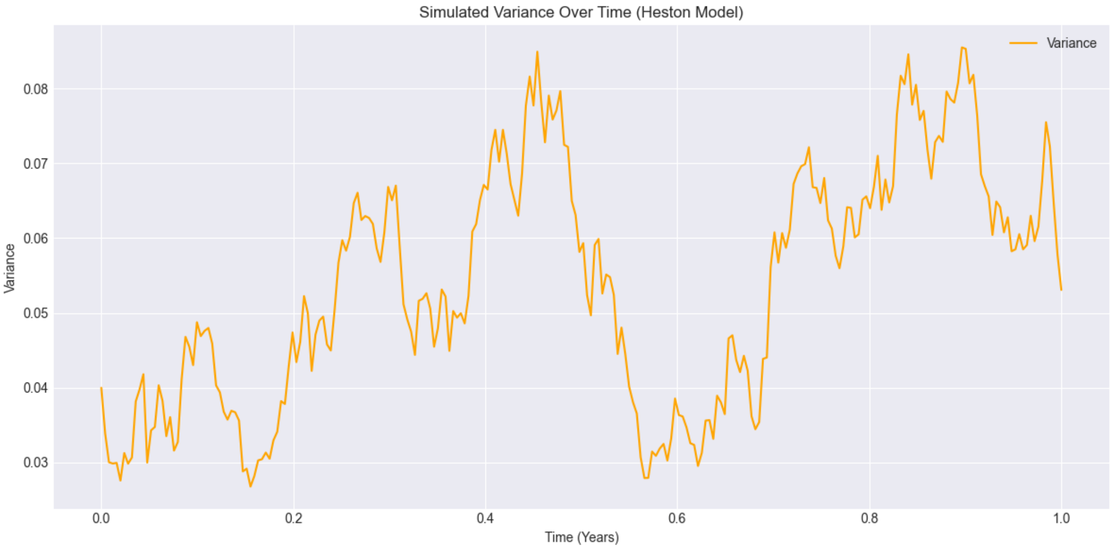
(First two rows)

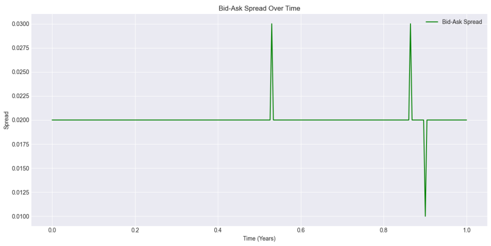
```
Time,Price,Variance,BidPrice_1,BidSize_1,BidPrice_2,BidSize_2,BidPrice_3,BidSize_3,...
0.0,100.0,0.04,99.99000000000001,35.28297304563942,99.98,15.59883912677782,99.97,23.40291999013041,...
```

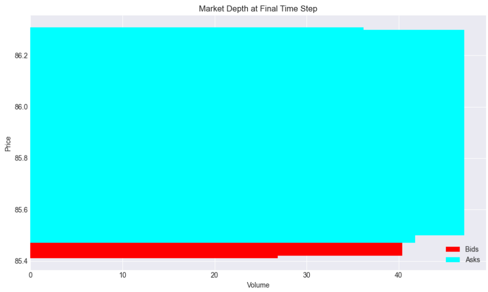
Users can visualize this data through the Streamlit app or run a backtest to assess strategy performance under Heston-like conditions.

Visualization


Simulated Stock Price Over Time
Price
115
110
105
100
95
90
85
80
75
0.0
0.2
0.4
0.6
0.8
1.0
Time (Years)
Price

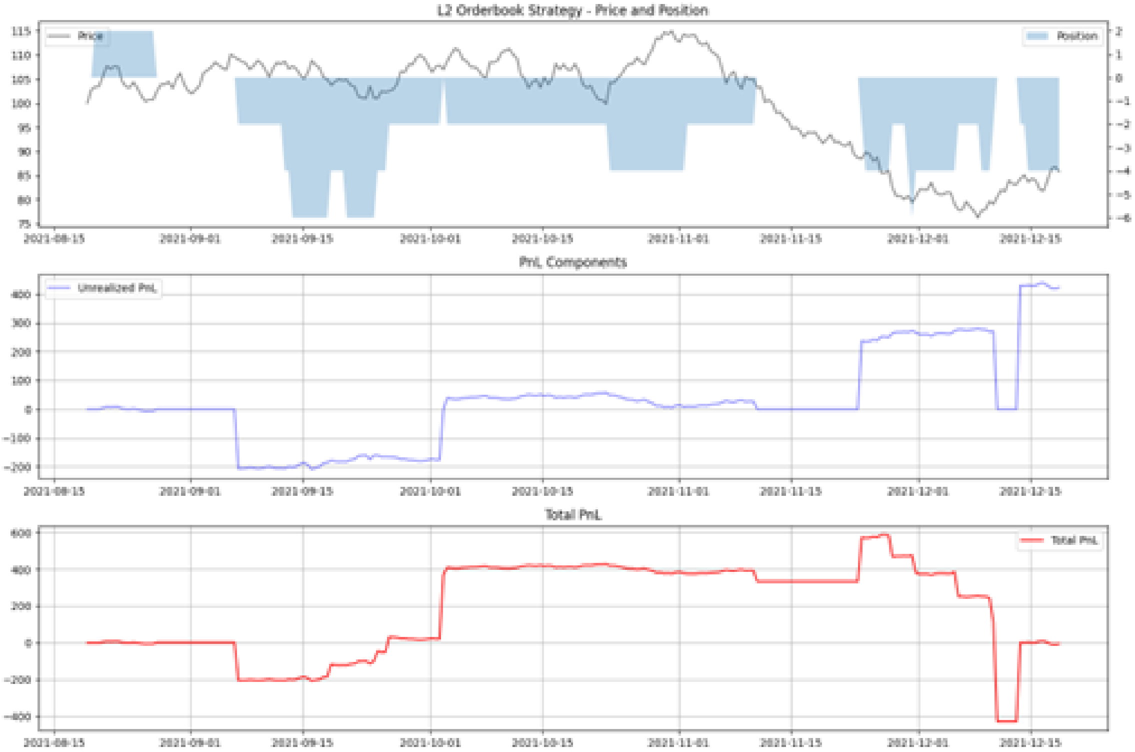

Simulated Variance Over Time (Heston Model)
Variance
0.08
0.07
0.06
0.05
0.04
0.03
0.0
0.2
0.4
0.6
0.8
1.0
Time (Years)
Variance

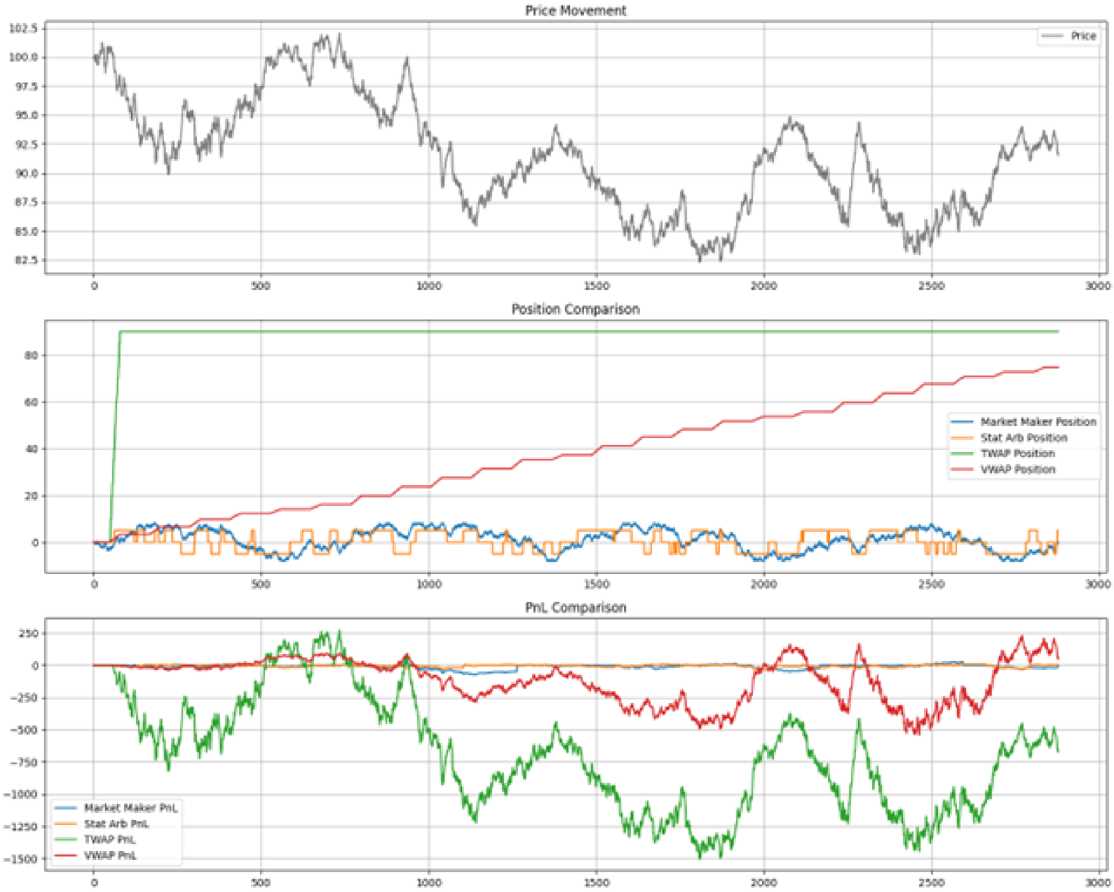

Bid-Ask Spread Over Time
Bid-Ask Spread
0.0300
0.0275
0.0250
0.0225
0.0200
0.0175
0.0150
0.0125
0.0100
0.0
0.2
0.4
0.6
0.8
1.0
Time (Years)
Spread



Backtester & Market Participants

Users observe aggregated results showing **trade count**, **final position**, **avg execution price**, etc., offering critical insights into strategy viability under various simulated market states.





6. Conclusion

Future Enhancements

1. **News Feed Plugin**
   Incorporate real-time or simulated news events influencing price fluctuations.
2. **Correlated Stocks**
   Generate data for multiple assets, modeling cross-correlations and portfolio dynamics.
3. **Enhanced Backtesting Metrics**
   Expand the framework to report advanced performance metrics (Sharpe ratio, max drawdown, etc.), improving decision-making for professional traders.

Final Thoughts

The **synthetic market data generator** successfully provides a **robust platform** for simulating a wide array of financial market scenarios. It enables **risk-free strategy testing**, fosters **educational exploration**, and assists researchers in **market behavior analysis**. With features like **customizable models**, **order book data generation**, **interactive visualization**, and **integrated backtesting**, this tool is poised to support traders, educators, and quant researchers alike. By bridging the gap between theory and practice, our solution contributes to **better-informed decision-making** in financial markets.
