DATA ANALYSIS WITH PYTHON (STOCKS CORRELATION & BACKTEST)

Dr. Shalini Goel 1, Rahul Kumar2, Hitesh Nautiyal 3, Shivang Dvivedi 4, Akshay Sharma 5
1 Associate Professor, 2 Student, 3 Student, 4 Student, 5 Student
1,2,3,4,5 – Department of Computer Science Engineering, HMR, Delhi, India

Abstract

The Stock Market is a fascinating yet uncertain world. It is full of uncertainties.
Only around 5% of traders make money in the market. According to most successful traders, having a strategy is better than randomly taking a trade. Butthe problem for manual trades arises in backtesting his/her strategies for verification. Backtesting is the general method for seeing how well a strategy or model would have done ex-post. In this minor project we have tried to backtest Particular strategy using
technology by eliminating manual hustle. We have also tried to find the correlation between two Indian Indices i.e. NIFTY & BANK NIFTY. We have formed our own setup and backtested it using python to get the desired output. We have written set of codes to automate the backtest for quick and reliable results. Our aim is to cut down the manual labour (Backtesting Execution) of the traders, so that they can focus more on the innovative ideasin the form of chart patterns.

Keywords

- Python
- Data Analytics
- Data Management
- Data Visualization

I. INTRODUCTION

The main objective of this project is to find the correlation between Nifty and
BankNifty in order to understand their nature. In this project we are also shifting the manual backtesting of strategy to automated backtesting usingpython. This project is helping in making the tiresome manual backtesting easier and faster by making it automated filtration mechanism, as we For a trader to do years of backtesting is a nightmare and hence they are
unable to focus more on developing strategy. Most of the time of traders goes into backtesting and hence strategy preparation and alteration take a backseat. So, it has become the need of the hour to shift to automated backtesting methodology.

II. RELATED WORK

[1] Successful Backtesting of Algorithmic Trading Strategies – Part I :-

Algorithmic backtesting requires knowledge of many areas, including psychology, mathematics, statistics, software development and market/exchange microstructure

What are key reasons for backtesting an algorithmic strategy?

Filtration - If you recall from the article on Strategy Identification, our goal at the initial research stage was to set up a strategy pipeline and then filter out any strategy that did not meet certain criteria. Backtesting provides us with another
can eliminate strategies that do not meet our performance needs.

Modelling - Backtesting allows us to (safely!) test new models of certain market phenomena, such as transaction costs, order routing, latency, liquidity or other *market microstructure* issues.

Optimisation - Although *strategy optimisation* is fraught with biases, backtesting allows us to increase the performance of a strategy by modifying the quantity or values of the parameters associated with that strategy and recalculating its performance.

Verification - Our strategies are often sourced externally, via our strategy pipeline. Backtesting a strategy ensures that it has not been incorrectly implemented. Although we will rarely have access to the signals generated by external strategies, we will often have access to the performance metrics such as the Sharpe Ratio and Drawdown characteristics. Thus we can compare them with our own implementation.

[2] Successful Backtesting of Algorithmic Trading Strategies – Part II

how to incorporate transaction costs, as well as certain decisions that need to be made when creating a backtest engine, such as order types and frequency of data.

Transaction Costs

One of the most **prevalent beginner mistakes** when implementing trading models is to neglect (or grossly underestimate) the effects of *transaction costs* on a strategy.

Commissions/Fees

The most direct form of transaction costs incurred by an algorithmic trading strategy are commissions and fees. All strategies require some form of access to an *exchange*, either directly or through a brokerage intermediary ("the broker"). These services incur an incremental cost with each trade, known as *commission*.

Slippage/Latency

Slippage is the difference in price achieved between the time when a trading system decides to transact and the time when a transaction is actually carried out at an exchange. Slippage is a considerable component of transaction costs and can make the difference between a very profitable strategy and one that performs poorly. Slippage is a function of the underlying asset volatility, the latency between the trading system and the exchange and the type of strategy being carried out.

Market Impact/Liquidity

Market impact is the cost incurred to traders due to the supply/demand dynamics of the exchange (and asset) through which they are trying to trade. A large order on a relatively illiquid asset is likely to *move the market* substantially as the trade will need to access a large component of the current supply. To counter this, large block trades are broken down into smaller "chunks" which are transacted periodically, as and when new liquidity arrives at the exchange. On the opposite end, for highly liquid instruments such as the S&P500 E-Mini index futures contract, low volume trades are unlikely to adjust the "current price" in any great amount.

Transaction Cost Models

In order to successfully model the above costs in a backtesting system, various degrees of complex transaction models have been introduced. They range from simple *flat* modelling through to a non-linear *quadratic* approximation. Here we will outline the advantages and disadvantages of each model:

Flat/Fixed Transaction Cost Models

Flat transaction costs are the simplest form of transaction cost modelling. They assume a fixed cost associated with each trade. Thus they best represent the concept of brokerage commissions and fees. They are not very accurate for modelling more complex behaviour such as slippage or market impact. In fact, they do not consider asset volatility or liquidity at all. Their main benefit is that they are computationally straightforward to implement. However they are likely to significantly under or over estimate transaction costs depending upon the strategy being employed. Thus they are rarely used in practice.

Linear/Piecewise Linear/Quadratic Transaction Cost Models

More advanced transaction cost models start with linear models, continue with piece-wise linear models and conclude with quadratic models. They lie on a spectrum of least to most accurate, albeit with least to greatest implementation effort. Since slippage and market impact are inherently non-linear phenomena quadratic functions are the most accurate at modelling these dynamics.

[3] Automated Trading Systems I Backtesting in strategy tester

Automated trading systems are programs that place orders on behalf of the trader. A trader sets the prerequisite condition for order placement based on technical analysis principles. The system will place orders automatically on the satisfaction of the necessary conditions. The automated trading system facilitates backtesting (strategy tester in MT4) on a demo account which gives a fair idea of the efficiency of the strategy.

Strategy Tester in MT4

The strategy tester is the PlayStation of traders where they get to try out different setups and their efficiency. The strategy tester can be accessed through View menu or by pressing Ctrl+R.
Indicators, as well as Expert Advisors, can be tested through the strategy tester in MT4.
Choose the strategy. Any default indicator or EA can be accessed through this.
Select an appropriate symbol for backtesting.
The model represents three kinds of input data viz.

Every tick , Control Points, Open Prices Only.

Custom strategies with Metatrader

The method discussed above is a simpler one but the user can use complicated trading strategies too based on his knowledge of technical analysis and MQL programming.

[4] Automated Trading Systems: The Pros and Cons

Pros:

- Minimize emotional trading
- Allows for backtesting
- Preserves the trader's discipline
- Allows multiple accounts

Cons:

- Mechanical failures can happen
- Requires the monitoring of functionality
- Can perform poorly

[5] Backtesting Systematic Trading Strategies in Python: Considerations andOpen Source Frameworks

The Components of a Backtesting Framework

Data and STS acquisition: The acquisition components consume the STS script/definition file and provide the requisite data for testing. If the framework requires any STS to be recoded before backtesting, then the framework should support canned functions for the most popular technical indicators to speed STS testing. Users determine how long of a historical period to backtest based on what the framework provides, or what they are capable of importing.

Performance testing applies the STS logic to the requested historic data window and calculates a broad range of risk & performance metrics, including max drawdown, Sharpe & Sortino ratios. Most all of the frameworks support a decent number of visualization capabilities, including equity curves and deciled-statistics.

Optimization tends to require the lion's share of computing resources in the STS process. If your STS require optimization, then focus on a framework that supports scalable distributed/parallel processing.
In the context of strategies developed using **technical indicators**, system developers attempt to find an optimal set of parameters for each indicator.

In a **portfolio context**, optimization seeks to find the optimal weighting of every asset in the portfolio, including shorted and leveraged instruments. On a periodic basis, the portfolio is rebalanced, resulting in the purchase and sale of portfolio holdings as required to align with the optimized weights.

Position sizing is an additional use of optimization, helping system developers simulate and analyze the impact of leverage and dynamic position sizing on STS and portfolio performance.

[6] Python Trading Toolbox: a gentle introduction to backtesting

We started this series by introducing some indicators based on price. Our goal is to use indicators, along with price and volume, to make investment decisions: to choose when to buy or sell a financial asset. There are different ways we can incorporate price, volume, and indicators in our investment decision process. The first, the most traditional, is to interpret their patterns in a discretional way, as followers of Technical Analysis do. Indicators can also be employed in a more quantitative approach as building blocks of a trading system that removes human discretion from the investment process. Algorithmic Trading, in particular, is an approach based on trading strategies that take positions in financial instruments on their own without human intervention. We can also use price, volume, and indicators as part of a more complex machine learning model for our investment decisions.

A process that can help us to answer this question is known as **backtesting**. With backtesting, we apply a trading or investment strategy to historical data to generate hypothetical results. We can then analyze those results to evaluate the profitability and the risk of our strategy.This process has its own pitfalls: there is no guarantee that a strategy that performed well on historical data will perform well in real trading. Real trading involves many factors that cannot be simulated or tested on historical data. Also, since financial markets keep evolving fast, the future may exhibit patterns not present in historical data. However, if a strategy cannot prove itself valid in a backtest most probably will never work in real trading. Backtesting can at least help us to weed out the strategies that do not prove themselves worthy.

[7] List of Most Extensive Backtesting Frameworks Available in Python

BlueShift by QuantInfy

Backtrader

Learn (Quantconnect)

PyAlgoTrade

Bt

Finmarketpy

Fastquant

[8] A Rookie Guide to Getting Started with Backtesting in Python!

Every Algorithmic trading rookie starts to wonder what Backtesting is and how it is implemented, yet many people ignore it based on their individual biases. But let me tell you frankly, it is a crucial step in building your algo trading robot.
To put it simply, your idea or strategy can be great in your head, but data never lies, and Backtesting is merely getting an indication as to whether your system will likely work or not.
To give you another example, think about the time when you have to decide which Mutual Fund or PMS service to invest in; you will always look at 3-Year, 5-Year Returns to arrive at a decision simply because data speaks for itself and **"Sab Mutual Funds Sahi Nahi Hai"**
Let's start Learning the **Backtesting framework** by creating and backtesting a simple strategy. We will be demonstrating a straightforward strategy to give a notion and introduce the library; the real-world strategy is much more complex. It needs various other factors to be considered, but the article is aimed at beginners.

[9] Algorithmic Trading – Backtesting a strategy in python

Create 20-day (+/- 2 standard deviations) Bollinger bands on the adjusted close price. Buy, when the price crosses the lower band from the top and hold until the price crosses the upper band from below the next time. Sell when the price crosses the upper band from below and hold until the price crosses the lower band from the top the next time.

Here are the steps to create your own back-testing code.

1. Import necessary libraries
2. Download OHLCV Data
3. Calculate daily returns
4. Create strategy-based data columns
5. Create strategy indicators
6. Create signals and positions
7. Analyze results

[10] A Powerful Reversal Trading Strategy – Back-testing in Python

Combinations of moving averages can yield powerful strategies and this is precisely the goal of the article. Sometimes we can combine different techniques such as trend-following and contrarian to form more robust strategies.

III. PROPOSED METHODOLOGY

When we're doing our research for the minor project in the data analysis
section, we found lots of stuff but the most challenging task we found was in the stock market. As in the stock market there are lots of variables present apart from the main ticker or price of the stock. We found that nowadays algorithmic trading has really started taking off.

It is extremely bizarre that a tech-savvy nation like INDIA took around 12-13 yrs
to understand the importance of algorithmic trading after SEBI (Securities and Exchange Board of India) allowed algo-trading in 2008. What we are trying to achieve in this project is a subset of algo-trading i.e. automated backtesting.

We have tried to deal with two problem statement in our project and those
are as follows:-

Is there any correlation b/w NIFTY and BANKNIFTY?

Develop a code base using python to automate specific strategy backtesting

In this particular project we have uses three step working model and they are
as follows:-

Taking a simpler strategy from an authentic and experienced traders strategies pool.

Collecting data of the instrument on which the strategy has to be deployed

Finding the number of occurrences/entries using that particular strategy on
our desired instrument for a desired data-sample

.

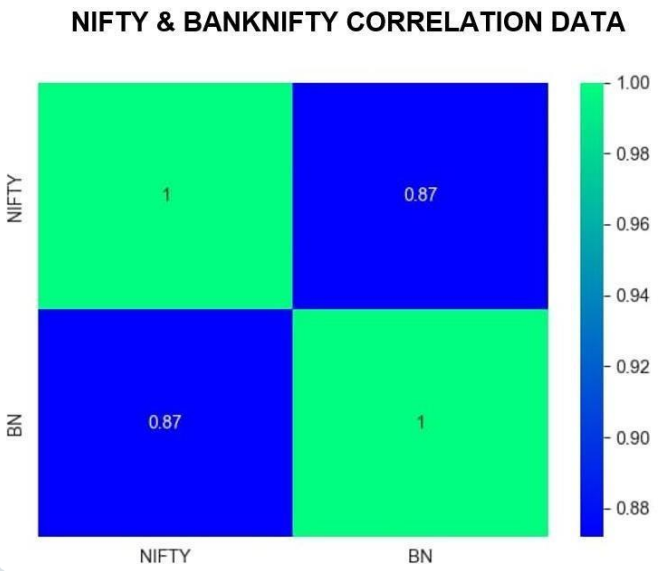
IV. IMPLEMENTATION

In this particular project we have executed our ideas in phase manner. In the first phase we found out the strategy/chart pattern on which we need to run our system for backtesting. We have used price-action based strategy since it is easier to use and has higher conversion probability. We have used different books to create a price-action based strategies. We have also taken the help of charts to observe some kind of pattern for particular movement.

In the second phase we have used our coding skills and other searching abilities to find the sample data set on which backtest needs to be run. We have taken the help of packages like tvdatafeedto collect the data. We have leveraged tvdatafeed to get the desiredmonths data of desired instruments.

In the last phase we have coded our strategy to get the backtest done. We have used the pandas to create Dataframe to run our strategy. The strategy is coded in highly efficient manner to decreasethe run-time as much as possible. While coding the strategy our primary focus was to create reliable as well as quick results in order to execute the strategy in the live market.

V. RESULTS



VI. FUTURE SCOPE

At present our code base works for particular strategy which is developed by

us. But the universe for this project is indefinite. The major step towards reformation would be to try to develop generalized codebase which can be used for n number of strategies with few requirements or input from user side.

VII. CONCLUSION

The whole project is developed using Python. The main module used in this project is Pandas for creating a Dataframe and manipulating it. We chosepython because of its rich modules and packages which assists coders in developing the code-base faster to get the desired output. Also, in the financial world Python is extremely popular.

VIII. REFERENCES

[1] https://www.quantstart.com/articles/Successful-Backtesting-of- Algorithmic-Trading-Strategies-Part-I/

[2] https://www.quantstart.com/articles/Successful-Backtesting-of- Algorithmic-Trading-Strategies-Part-II/

[3] https://wetalktrade.com/automated-trading-systems/

[4] https://www.investopedia.com/articles/trading/11/automated-trading- systems.asp

[5] https://www.quantstart.com/articles/backtesting-systematic-trading- strategies-in-python-considerations-and-open-source-frameworks/

[6] https://towardsdatascience.com/python-trading-toolbox-05- backtesting-84266edb1d59

[7] https://tradewithpython.com/list-of-most-extensive-backtesting- frameworks-available-in-python

[8] https://tradewithpython.com/a-rookie-guide-to-getting-started-with- backtesting-in-python

[9] https://blog.devgenius.io/algorithmic-trading-backtesting-a-strategy-in- python-3a136be16ece

[10] https://medium.com/the-investors-handbook/a-powerful-reversal- trading-strategy-back-testing-in-python-9ec1daa184b5

[11] Secrets of a Pivot Boss by Frank O Ochoa (For finding strategy)

[12] Trading Price Action Trends by Al Brooks (For finding strategy)

[13] Python For Data Analysis by Wes Mckinney (Coding Part)