Expert Talk

Build more robust trading strategies leveraging LSEG Tick History – Query for backtesting

Backtesting is a key part of developing systematic trading strategies. Having access to high-frequency tick data to backtest is useful not only for intraday trading strategies but also for more accurately simulating daily trading strategies using different execution algorithms. However, historically it has been difficult to use tick data, given the time and cost it takes to maintain internal tick databases. LSEG Tick History – Query is a solution which provides a fully-maintained database full of thousands of tickers with market-depth data, avoiding the need to maintain your own tick database.

Why backtest a trading strategy?

Let's say we create a trading strategy. Obviously, we can't tell with certainty what its future performance will be in a live environment. However, we can use historical data to gauge how it would have performed in the past. While past performance is not a guarantee of future performance, backtesting is an important part of developing a trading strategy.

If a trading strategy systematically loses money historically, it's an important data point to consider when deciding whether or not to run a strategy live – unless we have solid reasons to believe that market conditions are likely to change. Backtesting can also enable us to understand multiple factors, like the trading costs which a strategy can absorb before becoming unprofitable, and a strategy's estimated capacity.

LSEG DATA & ANALYTICS

What are the key components of an accurate backtest?

We need to ensure that any data used in our backtest has been properly timestamped and cleaned. If data has inaccurate timestamps, look-ahead bias can creep into our results (i.e., using future data) which can inflate our historical returns. We also need to make sure that our trading cost assumptions are accurate in our backtest. If these are incorrect, it can also result in overstating our historical returns. A backtest is only useful as part of developing a trading strategy if it accurately represents historical returns.

Using high-frequency data to backtest daily trading strategies can be beneficial

The complexity of a backtest can vary depending on which types of instruments we're trading and on the frequency of our trading rule. Clearly, for backtesting high-frequency trading strategies, having access to high-frequency market data is a necessity. If we're backtesting a daily trading rule, then on the surface backtesting might require less market data.

However, having high-frequency data is beneficial even for daily trading strategies. This dataset needs to be properly timestamped and clean. In particular, having high-frequency market data can enable us to backtest generating our trading signal at many different times of day, not just at market close, which is typically what we have if we use daily data exclusively.

Having access to more granular high-frequency tick data is often necessary for backtesting

With high-frequency tick data, we can simulate the execution of trades using different execution algorithms, whatever the frequency of our original trading signal. Rather than having our backtest execute every trade at the time of the signal in market orders at the closing price, we can opt to backtest how the strategy would perform if this execution was spread out using execution algorithms or at different times of the day. This can be particularly important for larger funds. The notional sizes they execute can have very large market impact. Always executing everything at market in large clips can have a negative impact on the returns of their trading strategies.

We'll often want access not only to top of order book data, with the best bid/ask quote, but also lower levels of the order book and the volumes associated with each bid/offer posted. Level 2 and especially level 3 data gives us the ability to accurately gauge market depth, given they allow us to access many more levels of the order book, not purely top of book data. Only examining the top of the order book will give us only a partial view of current market liquidity, whereas market depth will give us a much fuller picture.

These more granular datasets will enable us to compute execution algorithms such as volume-weighted average price (VWAP) or construct our own estimates for a risk transfer price, to more accurately gauge how much a market order would cost us at the time. We can also calculate metrics such as market imbalance, where we are monitoring the skew in the order book, using more granular market data, which can be an input into a trading algorithm.

Why it can be time-consuming to maintain your own tick database

In order to backtest a trading strategy with high-frequency tick data, we need access to a database with vast amounts of tick data. It can be a time-consuming proposition to keep a tick database up to date, whether we are using streaming data to populate it on a regular basis or even downloading data to update it periodically. We also need to manage the hardware for the tick database, which needs to be scaled over time as more data comes online, and consider increasing user demand for compute power. If we are using level 2 and in particular level 3 data, the demands on compute and storage size will increase significantly.

Is there a way we can focus on backtesting with tick data while avoiding the time and expense of maintaining a tick database internally?

How LSEG Tick History – Query makes it easier to work with tick data for backtesting

With LSEG Tick History – Query, we no longer have to maintain our own tick database internally. Instead, we can use LSEG Tick History which is a prepopulated data with over 25 years of tick data with thousands of tickers, which is continuously updated, at a rate of 2 TB per week. It also includes much more granular level 2 and level 3 data.

We can access LSEG Tick History via BigQuery, using the very familiar SQL language to perform queries to do backtesting. BigQuery is accessible through a web interface or via APIs, including in Python, so it can integrate with our own backtesting framework.

Furthermore, we no longer need to manage our own internal hardware and scaling. Instead, BigQuery automatically scales as the compute is required. Rather than taking the data to our compute (i.e., downloading and managing it locally), we instead take our compute to the data, saving time and money.

Saeed Amen

Saeed Amen is the founder of Cuemacro. Over the past 15 years, Saeed Amen has developed systematic trading strategies at major investment banks including Lehman Brothers and Nomura. He is the author of *Trading Thalesians: What the ancient world can teach us about trading today* (Palgrave Macmillan) and the co-author of *The Book of Alternative Data* (Wiley). Through Cuemacro, he now consults and publishes research for clients in the area of systematic trading. He has developed many Python libraries, including finmarketpy and tcapy for transaction cost analysis. His clients have included major quant funds and data companies. He has presented his work at many conferences and institutions, including the ECB, IMF, Bank of England and Federal Reserve Board. He is also a visiting lecturer at Queen Mary University of London and a co-founder of the Thalesians.