HeadlandsTech

AUGUST 3, 2017 BY MAX DAMA, QUANTITATIVE RESEARCHER

Quantitative Trading Summary

This summary is an attempt to shed some light on modern quantitative trading since there is limited information available for people who are not already in the industry. Hopefully this is useful for students and candidates coming from outside the industry who are looking to understand what it's like working for a quantitative trading firm. Job titles like "researcher" or "developer" don't give a clear picture of the day-to-day roles and responsibilities at a trading firm. Most quantitative trading firms have converged on roughly the same basic organizational framework so this is a reasonably accurate description of the roles at any established quantitative trading firm.

The product of a quantitative trading company is an automated software program that buys and sells electronically traded securities to make a profit. This software program is supported by many systems designed to maintain and optimize it. Most companies are roughly divided into 3 main groups: strategy research, core development, and operations. Employees generally start and stay in one of these groups throughout a career. This guide focuses on strategy research and core development roles.

Primary job requirements:

- Strategy research ('research'): Programming, statistics, trading intuition, and the ability to understand market data
- Core development ('dev'): Low-level software engineering, networking, and system architecture

The software components of a quantitative trading system are built by one of these two teams. The majority of the components are built in-house at most

HeadlandsTech

can be a separate process, although we'll discuss some variants later.

Programs for live production trading:

1. Market data parser: Dev. Normalizes each exchange's protocol (including different versions over time) into the same internal format.
2. Trading strategy: Research/Dev. Receives normalized data, decides whether to buy or sell.
3. Order gateway: Dev. Converts from internal order format to each exchange's order entry protocol (different than the market data protocol).

Programs to support live production trading:

1. Monitoring GUI: Dev. GUIs used to be important for click traders but are now mainly used to monitor that the trading system is performing appropriately. They are still occasionally used to manually adjust a few parameters, such as overall risk tolerance.
2. Drop copy: Dev. Secondary order confirmation to make sure you have the trading positions you think you do.
3. Market data capture: Dev. Record the market data in parallel to what's going into the strategy to verify later that the strategy behaved as intended and to run statistical tests on historical data (live capture is more reliable than purchasing from a vendor so most major firms avoid buying data from a vendor).
4. Startup scripts: Dev/Operations. Launch all these different software programs in the right order and at the right time of day each time they need to be restarted (typically daily or weekly), and alert or recover from startup problems.

Programs to optimize and analyze the trading strategy:

the best.

2. Production reconciliation: Research. Metrics to confirm that the state in the trading strategy's internal algorithm matched calculations using captured market data.
3. Back testing simulator: Research. Shows estimated trading strategy profit or loss on historical data.
4. Graphing: Dev/Research. Display profit or loss, volume, price and other statistics over time.

In a 'typical' established quantitative trading company the department breakdown would be:

- Research
- Dev
- Back office and operations
    - operations/monitoring
    - telco/networking/hardware
    - accounting/HR
    - management/business development
    - legal/compliance

Since we won't focus on them later, here's a brief description of the latter groups:

- Operations/monitoring: Monitor strategies and risk intraday and overnight to ensure there are no problems (like Knight Capital's +$400m loss).
- Telco/networking/hardware: Purchase and rack servers, configure switch firmware, operating system settings, and network interface cards or FPGAs, connect co-located datacenters (possibly in different countries), etc.
- Accounting/HR: Like any business there is tax, accounting, and human resources work

countries, negotiating fees, licensing telecom networks, and keeping ahead of new updates.
- Legal/compliance: Trading is one of the most regulated industries. There are US and international regulators (SEC, CFTC, FCA, etc), huge and diverse rulesets such as MIFID, industry regulating agencies like FINRA, and exchange-level self-regulatory regimes (CME, NYSE, etc) that each have their own rules. Ensuring and documenting compliance with each set of rules takes a lot of work.

Some of the key differentiating factors between quantitative trading companies are:

1. How they divide up research teams – internal collaboration vs competition between siloed research/trading teams.
2. Which exchanges and products they focus on.
3. What type of trading strategy is used and how it's optimized.

Although we are unable to explain how each specific company divides up their research/trading teams, the overall structure and organization of employees and software at most major quantitative trading firms follows a similar general pattern to what was described above.

Trading strategies

Now that you have a high-level understanding of what a typical quantitative trading company does and the different roles that exist, let's go into more detail about trading strategies.

The industry has generally settled on three main types of strategies that are sustainable because they provide real economic value to the market:

are still competitive in arbitrage have one of 3 advantages:

- Scale: To determine that some complex option or futures spread products are mispriced relative to a set of others, nontrivial calculations must be performed, including the fee per leg, and then the hedged position has to be held and margined until expiry. Being able to manage this and have low fees requires scale.
- Speed: Speed either comes from having faster telco or being able to hedge. For example, triangular arbitrage on FX products traded in London, NY, and Japan and are a major impetus for the Go West and Hibernia microwave telecom projects. Arbitrageurs rely on the speed of their order gateway connections so they can hedge on related markets if they are overfilled.
- Queue position: Being able to enter one leg of an arbitrage by passively buying on the bid or selling on the offer reduces costs by not having to cross the spread on that leg, so being able to achieve good queue position can give an edge in arb trades.

- Market Taking: Placing a marketable buy or sell order to profit from a predicted price change. The economic value market takers are being paid for is either:
  - properly pricing the relative value of related securities
  - trading and thereby contributing to price discovery in products after observed changes in supply and demand

  Like a real estate negotiation that can change a deal's value minute-by-minute when negotiators come to the table face-to-face and discover each other's positions, even though the fundamentals of the deal or real estate market certainly don't fluctuate by the minute, market takers are the high-stakes-mediators of the trading world. Market taking requires predictive signals and relatively low-latency because you pay to cross the spread. A common low-latency market taking strategy would be to attempt to buy the remaining liquidity at a price after a large buy trade. Some firms have

- Market Making: Posting passive non-marketable buy and sell orders with the goal to profit from the spread. The economic value market makers are being paid for is connecting buyers and sellers who don't arrive at the market at the same time. Market makers are compensated for the risk that there may be more buyers than sellers or vice versa for an extended time, such as during times of market stress.

Basic trading system design

A quantitative trading system's input is market data and its output is orders. In between is the strategy algorithm.

Input

The input to a trading system is tick-by-tick market data. The input is handled in an event loop. The events are the packets sent by the exchange that are read off the network and normalized by the market data parser. Each packet gives information about the current supply and demand for a security and the current price. A packet can tell you one of three things:

- A limit order was added to the book. Primary fields: {price, side, order id, quantity}
- A limit order was canceled. Primary fields: {order id}
- A trade occurred. Primary fields: {price, aggressor side, quantity}

For example, a few packets look like this (*for a more detailed, real example see section 4 appendix 1 of this spec*):

AddOrder { end_of_packet: 1; seq_number: 103901; symbol_id: 81376629; receive_time: 13:03:46.304606537089; source: 1; side: S; qty: 1; order_id: 210048618; price: 99.25; }

price: 99.00; side: S; }

Trade { end_of_packet: 1; seq_number: 103902; symbol_id: 81376629; receive_time: 13:03:46.304606537835; source: 1; aggressor_side: B; qty: 1; order_id: 210048321; price: 99.00; match_id: 20154940; }

If the trading system adds up all the AddOrder packets and subtracts CancelOrder and Trade packets, it can see what the order book currently looks like. The order book shows the aggregate visible supply and demand currently available at each price. The order book is an industry-standard normalization layer.

When you add up all the orders, the order book could look like this:

Sell 10 for $99.25

Sell 5 for $99.00 (best offer)

Buy 10 for $98.75 (best bid)

Buy 10 for $98.50

This is the main view of the market data input used by the strategy algorithm.

Strategy algorithm

To put into practice what we discussed above, let's outline a market taking strategy utilizing what is often referred to as market micro-structure signals that may have made money back before quantitative trading became very competitive. Some companies have each member of their intern classes program a strategy like this as a teaching project during a summer. This strategy calculates some signals using the order book as input, and buys or sells when the aggregate signals are strong enough.

A signal is an algorithm that takes market data as the input and outputs a theoretical price for a security. Market micro-structure signals generally rely on price, size, and trade data coming directly from data feeds. Please reference the order book state provided previously as we walk through the following signal examples.

- A basic signal, likely used in some form by most firms, is 'book pressure'. In this case book pressure is simply (99.00*10 + 98.75*5)/(10+5) = 98.9167. Because there is more demand on the bid, the theoretical price is closer to the offer than the bid. Another way of understanding why this is a valid predictor it is that if buy and sell trades randomly arrive in the market on the bid and offer, then there's a 2/3 chance of them filling the entire offer before the entire bid, because it's 2 times bigger, so the expected future price is slightly closer to the offer than the bid.
- A second basic signal that many quantitative trading firms use is 'trade impulse'. A common form is to plug trade quantity into something like the book pressure formula, but with the average bid and offer quantity in the denominator instead of the current quantity (let's say the average is 15). So if there is a sell trade for 9 on this book, the trade impulse would be -0.25*9/15 = -0.15. This example signal would only be valid for the span of 1 packet. Another way of understanding why this is a valid predictor is that sometimes buy and sell trade quantity is autocorrelated over very short intervals, because there are often multiple orders in flight sent in reaction to the same trigger by different people (this is easily measured), so if you see one sell trade, then typically the next order will also be a sell.
- A third common basic signal is 'related trade'. Basically, you could just take the same signal as (2), but translate it over from a different security that is highly correlated, by multiplying it by the correlation between them.

The book pressure and trade impulse signal are enough to create a market taking strategy. After the sell trade for 9, the remaining quantity on the book is:

Sell 5 for $99.00 (best offer)

Buy (10-9 = 1) for $98.75 (best bid)

Buy 10 for $98.50

But our theoretical price is = book pressure + trade impulse = (99.00*1 + 98.75*5)/(1+5) + -0.25*9/15 = 98.79167 – 0.15 = $98.64167! Since our theoretical price is below the best bid, we will send an order to sell the last remaining quantity of 1 at $98.75, for a theoretical profit of $0.10833.

That is a high-level overview of a simple quantitative strategy, and provides a basic understanding of the flow from the input (market data) to the output (orders).

Digression: Trade signal on an FPGA

If you ran the market taking strategy from the previous section live in a real trading system, you would likely find that your orders rarely get filled. You want to trade when your theoretical price implies there's a profitable opportunity, but other trading systems are faster than yours so their orders reach the market first and there's nothing left for you.

State of the art latency, as of 2017, can be achieved by putting the trading logic on an FPGA. A basic trading system architecture with an FPGA is to have the FPGA connected directly to the exchange and also to the old trading system. The old trading system is now only responsible for calculating hypothetical scenarios. Instead of sending the order, it notifies the FPGA what hypothetical condition needs to be met to send the order. Using the same case as before, it could hypothetically evaluate the signal for a range of trade quantities:

- Sell trade, quantity = 1…

- Sell trade, quantity = 4…
- Sell trade, quantity = 5…
- Sell trade, quantity = 6: (99.00*4 + 98.75*5)/(4+5) + -0.25*6/15 = 98.7611
- Sell trade, quantity = 7: (99.00*3 + 98.75*5)/(3+5) + -0.25*7/15 = 98.7271
- Sell trade, quantity = 8…

With any sell trade of quantity 7 or more, the theoretical price would cross below the threshold of the best bid (98.75), indicating a profitable opportunity to trade, so we'd want to send an order to sell the remaining bid. With a trade quantity of 6 or less we wouldn't want to do anything.

The FPGA is pre-programmed to know the byte layout of the exchange's trade message, so all it has to do now is wait for the market data, and then check a few bits and send the order. This doesn't require advanced Verilog. For example, the message from the exchange could look like the following struct:

```
struct Trade {

uint64_t time;

uint32_t security_id;

uint8_t side;

uint64_t price;

uint32_t quantity;

} __attribute__((packed));
```

Because of the relative ease of this setup, it has become a very competitive trade – some trading firms can make these types of trade decisions in less than one microsecond. Also, because the FPGA connects directly to the exchange, an data internally with a switch, the switch might introduce too much latency to be competitive. Many companies will now pay for multiple connections which raises their costs significantly.

Digression: 'Minimum viable trading system'

As I mentioned above, the simple 3-signal trading strategy could have made money several years ago. Even a few years ago, the 'minimum viable trading system' that could cover trading fees was simple enough that an individual could build a successful one. Here's a good article by someone who created their own trading system in 2009, and could be another starting point to understand the basics of automated trading if all of this has gone over your head- http://jspauld.com/post/35126549635/how-i-made-500k-with-machine-learning-and-hft.

This guide only covers, at a high level, trading and work being done by professionals in established quantitative trading firms, so things like co-location, direct connection to the exchange without going through an API, using a high-performance language like C++ for production (never Python, R, Matlab, etc), Linux configuration (processor affinity, NUMA, etc), clock synchronization, etc are taken for granted. These are large and interesting topics which are now well understood inside and outside the industry.

Other strategies besides market micro-structure signals

Market micro-structure signal based strategies, as described above before the two digressions, are just one type of strategy. Here are some other example trading strategy algorithm components used by many major quantitative trading companies:

- Model based

    - Triangular arbitrage for FX or ADRs
    - Implied chains for futures spreads, butterflies, and packs
    - Weighted constituent price calculation for ETFs
- Rule based
    - Only buy or sell during a certain time range
    - Don't trade against an iceberg order
    - Cancel a resting order if the queue position is worse than 50%
    - Don't trade if the last 10 trades lost money

Supporting research infrastructure

Now that you have a brief high-level overview of the production trading system, let's dive deeper into research. The job of a researcher is to optimize the settings of the trading system and to ensure it is behaving properly. Working for an established company, this whole software system will likely already be in place, and your job would be to make it better.

With that in mind, here are some more details about 4 other main software components I listed above that are programmed and used by the research team to optimize and analyze the trading strategy:

1. Parameter optimization: Most major quantitative trading firms have a combination of signals, model-based pricing, and rule-based logic. Each of these also have parameters. Parameters enable you to tailor a generic strategy to make more money on a specific product, or adapt it over time. For one product, you might want to weight a certain signal higher than another, or you might want to down-weight it as the signal decays. You quickly run into the curse of dimensionality as parameter permutations multiply. **One of the main jobs for a researcher is to figure out the optimal settings for everything, or to figure out automated ways of optimizing them.** Some approaches include:
    - Manual selection based on intuition

HeadlandsTech

- Backtesting different settings and picking the best

2. Production reconciliation: Sophisticated strategies have many internal components that need to be continually verified in live production trading. Measuring these, monitoring them, and alerting on discrepancies is how researchers make sure things are working as they expected. If the algorithm performs differently in production than it did on historical data, then it may lose money when it was supposed to be profitable.
3. Backtesting simulator: Plenty of information is available publicly about backtesting, such as the tools available from Quantopian or TradeStation. Simulating a low latency strategy using tick data is challenging. The volume of data to simulate a single day reaches into the 100s of GBs so storing and replaying data requires carefully designed systems.
4. Graphing: The trading strategy is a mathematical formula in a computer, so debugging it and adding new features can be difficult. Utilizing a Python or JavaScript plotting library to publish custom data and statistics can be helpful. Additionally, it is essential to understand positions and profits or losses during and after the trading day. Graphical representations of different types of data sets makes many tasks easier.

Conclusion

Most people who are new to the industry think that researchers primarily work on new signal development, and developers primarily optimize latency. Hopefully now it's obvious that the system has so many components that those two jobs are just a few parts of a much wider set of roles and responsibilities. The most important skills for success are actually very close attention to detail, hard work, and trading intuition. On top of that it should be clear that having strong programming skills is essential. All of these systems are tailor-made in-house and have to be constantly tweaked and improved by the users themselves – you.

HeadlandsTech

Note:

The information above is a collection of some helpful information to shed some light on what a quantitative trading firm does and what you could be doing if you worked at one. The information, although intended to be helpful to you, should not be relied on and is not represented to be accurate or current. Please note this is by no means an exhaustive description of what goes on at a quantitative trading firm. Nor should this be taken as covering industry best practices or everything you need to know to start trading quantitatively. This is simply a very high overview of information I think those considering joining a quantitative trading firm may find useful as they navigate the interview process.

Appendix: Latency and the timing of events

Similar to the breakdowns by Grace Hopper (https://youtu.be/ZR0ujwlvbkQ?t=45m08s) and Peter Norvig (http://norvig.com/21-days.html#answers), here's a table of approximately how long things take:

- Time to receive market data and send an order via an FPGA: ~300 nanoseconds
- Time to receive market data and send an order via a 'slow' software trading system: ~30 microseconds
- Minimum time between two packets from the exchange: ~10-1000 microseconds
- Microwave between BATS and INET stock exchanges: ~100 microseconds
- Fiber between BATS and INET stock exchanges: ~150 microseconds
- Time for an exchange to match an order and send a response: ~100 microseconds – ~5 milliseconds
- Microwave between NY and Chicago: ~4 milliseconds
- Fiber between NY and Chicago: ~7 milliseconds
- Fiber between NY and European exchanges: ~35 milliseconds

HeadlandsTech

Exchanges almost all use different technology, some which dates back 10+ years. Different technology decisions and antiquated infrastructure have resulted in trading idiosyncrasies. There are many publicly available discussions of the effects of these idiosyncrasies. Here are a few interesting items:

https://www.slideshare.net/EurexGroup/deutsche-brses-t7-insights-into-trading-system-dynamics

https://www.bloomberg.com/news/articles/2017-03-17/currency-traders-race-to-reform-last-look-after-bank-scandals

https://markets.cboe.com/us/equities/market_statistics/order_type_usage/

http://www.cmegroup.com/notices/disciplinary/2016/11/NOTICE-OF-EMERGENCY-ACTION/NYMEX-16-0600-ELDORADO-TRADING-GROUP-LLC.html

https://brillianteyes.wordpress.com/2010/08/28/espeed-trading-procotol/

http://cdn.batstrading.com/resources/membership/CBOE_FUTURES_EXCHANGE_PLATFORM_CHANGE_MATRIX.pdf

https://quantitativebrokers.com/whitepapers

www.wsj.com/articles/SB10001424127887323798104578455032466082920

https://www.wsj.com/articles/SB10001424127887324766604578456783718395580

UNCATEGORIZED