Algorithmic Trading

Dr. Javier Sabio González
Head of Advanced Analytics & Algorithmic Trading, BBVA C&IB

Topics

1. Maket Macrostructure
2. Market Microstructure
3. **Algorithmic Trading Fundamentals**
4. Algorithmic Execution
5. Algorithmic Market-Making
6. Algorithmic Investment
7. The Future of Algorithmic Trading

3. Algorithmic Trading Fundamentals

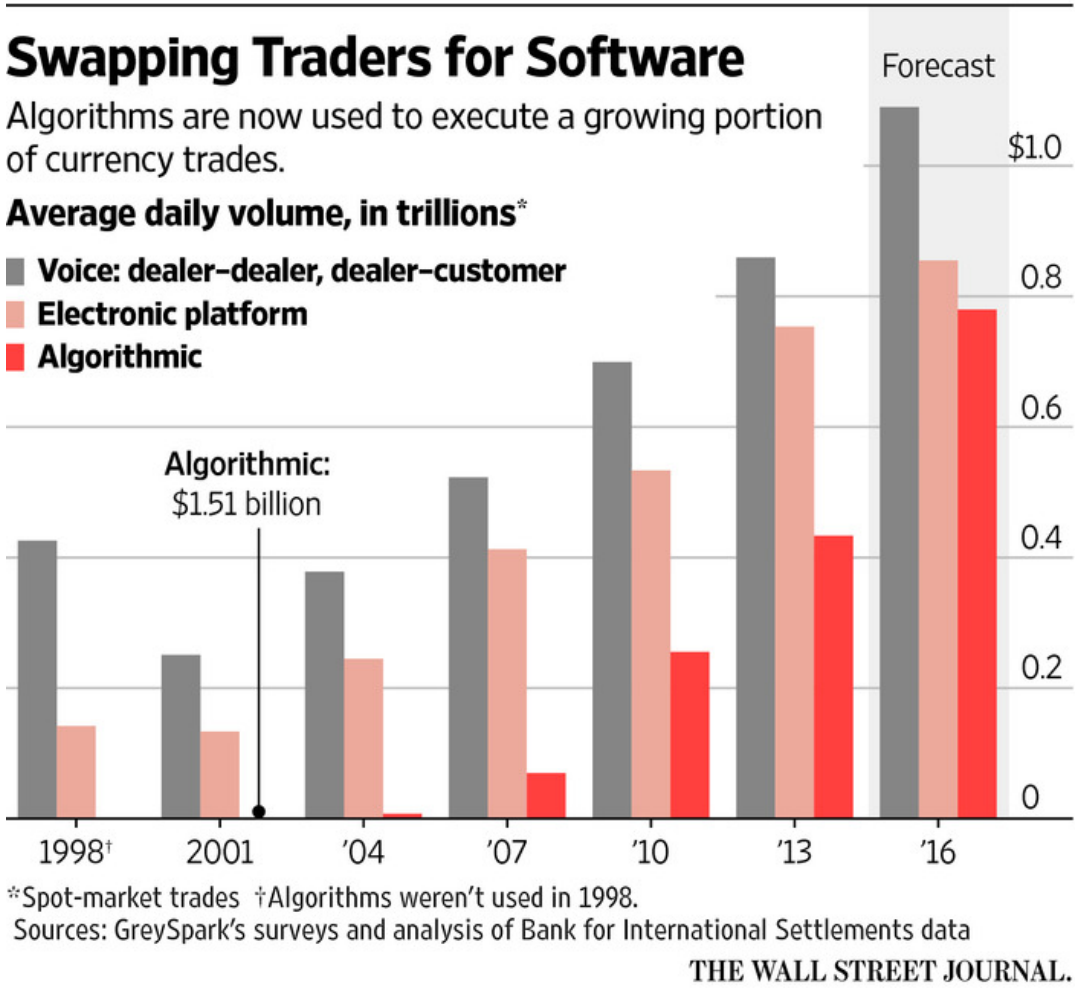
I. What is a trading algorithm?

- "**Trading technology in which order and trade decisions are made electronically and autonomously"** (BIS, 2011)
- "Trading in financial instruments where a **computer algorithm automatically determines individual parameters of orders** such as whether to initiate the order, the timing, price or quantity of the order or how to manage the order after its submission, **with limited or no human intervention**" (Article 4, Definition 39, Directive 2014/65/UE aka MiFID 2)
- **Remarks**:
  - In this course, we consider algorithmic trading and automated trading as synonyms. In some references the term algorithmic trading is used for algorithmic execution, a type of automated/algorithmic trading.
  - Algorithmic trading is a subset of "systematic trading" (trading strategies defined in a methodical way) and of "quantitative trading" (trading strategies defined using mathematical analysis and computations)
  - MiFID 2 definition is very wide, covering both very complex trading algorithms but also simple trading rules like auto-negotiation rules and dynamic stop-losses. In this course we will focus on the first definition, which concerns more "complex" strategies.

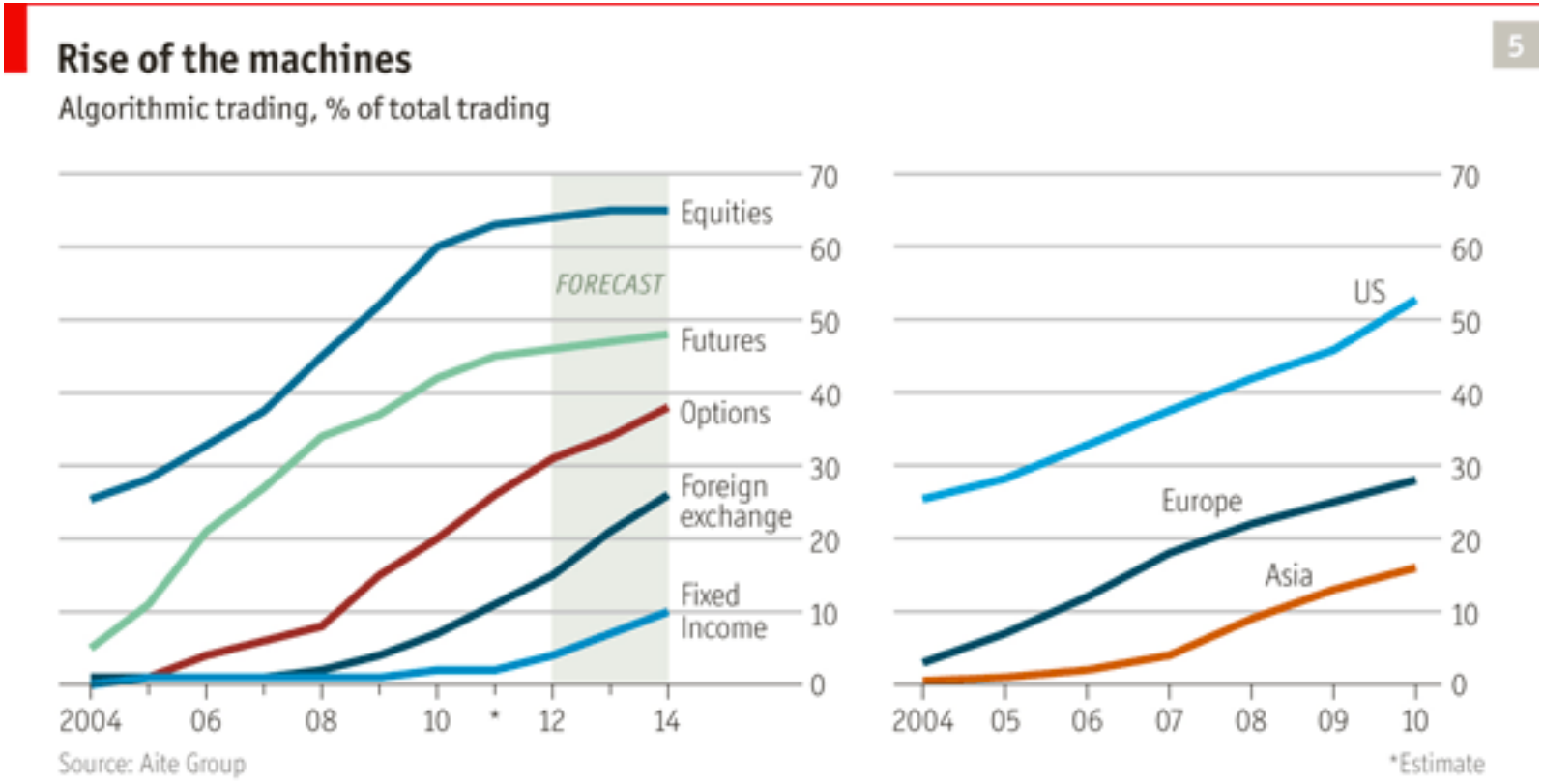
High-frequency Trading (HFT)

- According to BIS, 2011: "**Subset of Automated Trading in which orders are submitted and trades executed at high speed, usually measured in microseconds, and a very tight intraday inventory position is maintained.**
  - HFT strategies seek to gain advantage from the ability to process information on market conditions quickly and react instantaneously
  - The strategies tend to result in a large number of small trades that are held for short periods and also tend to generate a high volume or message traffic
  - To gain an edge, HFT companies tend to locate their servers physically closed to an electronic platform's matching engine (co-location) and hence minimise latency" (BIS, 2011)
- According to **MiFID 2** (Article 4, Definition 40 of the Directive): "Algorithmic trading technique characterised by:
  - *infrastructure intended to minimise network and other types of latencies, including at least one of the following facilities for algorithmic order entry: co-location, proximity hosting or high-speed direct electronic access*
  - *system-determination of order initiation, generation, routing or execution without human intervention for individual trades or orders*
  - *high message intraday rates which constitute orders, quotes or cancellations"*

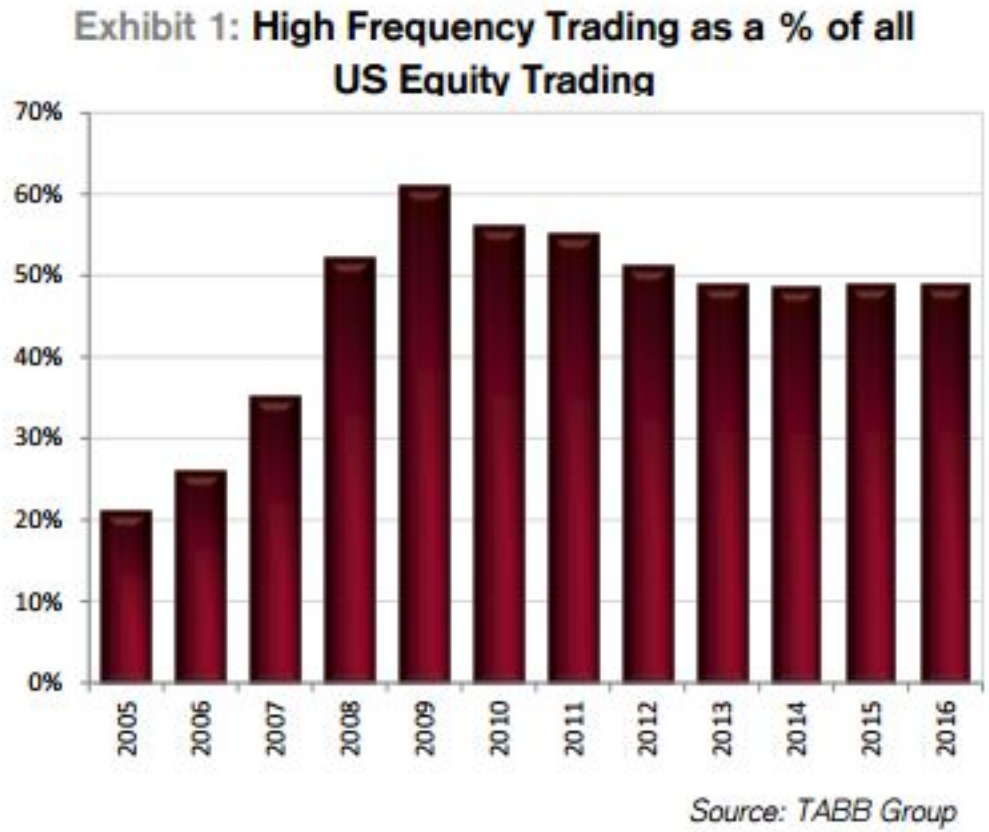
Algorithmic Trading growth



Algorithmic Trading across financial markets



High-frequency Trading growth



Algorithmic Trading chronology

- Algorithmic trading evolution is quite connected to the evolution of electronic trading
- **70's-80's:**
  - Computerisation of the order flow in financial markets and popularisation of quantitate analysis and trading with the Black Scholes option pricing model.
  - First computer algorithms to optimally split block trades to reduce market impact
- **October 19th 1987**: the stock market crash is attributed to "program trading" and "portfolio insurance": computer strategies that create synthetic put options by trading equity futures and using the Black Scholes formula.
- **80's-90's:**
  - fully electronic execution in some US Exchanges and development of first electronic communication networks (ECNs).
  - In the U.S., decimalisation (reduction of the minimum tick size) might have encourage algorithmic execution
  - Popularisation of "pairs trading" algorithms (a kind of relative value / mean reversion strategy) in Wall Street, probably originating in Morgan Stanley. David Shaw and James Simons are among the pioneers of these techniques in Investment Banks, though later they would set up their hedge funds.
- **1997-98**: Bertsimas and Lo publish "Optimal Control of Execution Costs", and Almgren and Chriss "Optimal liquidation", setting a mathematical framework to build algorithmic execution strategies

Algorithmic Trading chronology

- **2001**: IBM publishes “Agent-Human Interactions in the Continuous Double Auction”, where researches found that two algorithmic strategies (IBM's own MGD, and Hewlett-Packard's ZIP) could consistently out-perform human trader
- **2000’s:** development of low latency hosting and HFT algorithms, hough HFT would account for less than 10% of total equity trading in the USA yet.
- **2005**: HFT share in the USA market rises to 35%
- **2005-2007**: Regulations like Reg NMS and MiFID seek to increase competition across trading venues, with many of new ones opening. Fragmentation of market liquidity incentives the use of execution strategies (in particular Smart Order Routing), as well as relative value and arbitrage strategies, trying to automate the exploitation of arbitrage opportunities.
- **2008**: Avellaneda and Stoikov publish “High-frequency trading in a limit order book”, setting a mathematical framework to build market making strategies
- **May 6th 2010:** a computer driven sale from HFT algorithms produces a Flash Crash, with the Dow Jones plummeting 1000 points in a single day, and nearly 1 trillion $ were wiped out of the market value
- **2011**: Fixnetix develops first microchip able to execute trades in nanoseconds
- **2012**: HFT becomes responsible of about 60% of all US trades, although share would later decline to around 50%

Algorithmic Trading chronology

- **September 2012**: First algorithms using news based signals thanks to the launch of Dataminr
- **April 2013:**
  - Bloomberg terminal includes live tweets into its data service
  - A false tweet about white-house bombing causes a flash crash and the re-boom, with Dow plunging 1% in 3 minutes
  - A server in Washington DC is capable to transmit data to New Jersey at the speed of light using microwave transmission technology
- **September 2013**: Italy becomes first country to impose tax on HFT
- **October 15th 2014**: In a period of 12 minutes the yield of 10-year Treasury notes plunges dramatically and then surges. The movement is attributed to HFT
- **October 7th 2016**: HFT is blame of a quick plunge and the recovery of the pound in a few minutes
- **2017**: J.P. Morgan claims to release an Equities Execution Algorithm based on Deep Reinforcement Learning, the same Machine Learning technology behind the success of Alpha Go.
- **January 2017**: Solarflare reduces trading latency to 20 nanoseconds
- **March 2017**: HFT makes up 80% of Bitcoin trading
- **3th January 2018**: MiFID 2 European Regulation introduces a comprehensive set of rules to regulate the algorithmic and high-frequency trading activities

Reasons to use Algorithmic Trading

- **Speed**: computer algorithms can send orders and react to changing market conditions in much lower time than human traders
- **Information processing**: computer algorithms can check and process a large number of data sources at the same time
- **Scalability**: computer algorithms allow to scale up the number of trades, instruments and markets without requiring extra human head-counts.
- **Systematic trading**: computer algorithms use a systematic approach to trading (although this is not necessary restricted to them). This helps minimise human emotions and helps evaluating the performance of the strategy with historical data and comparing between alternative strategies.
- **Quantitative trading**: computer algorithms usually use a quantitative approach to trading, helping to consistently optimise the trading target, for instance reducing transaction costs in execution or maximising the profit & loss with a minimum inventory in market making.
- **Minimise human errors**: computer algorithms help reducing human mistakes (e.g. fat fingers), although they introduce their own new set of risks (see later)

II. Types of trading algoritms

Following again BIS, 2011, we have three categories of algorithmic strategies:

• **Trade execution:** "Algorithms (that) are concerned with minimising the price impact of a transaction" (buy or sell a block of instruments)

- They "generally split a large trade into smaller trades, which they execute over time and across venues."
- "Such algorithms are used by broker-dealers as well as end investors (e.g. asset managers) for establishing or exiting a position at low cost"

• **Market-making:** "Algorithms (that) generate indicative or live screen quotes or may be used to reply to requests for quote (so called auto-quoting)"

- Broker-dealers and HFT houses have "algorithms in particular (that) do business by standing ready to trade at a publicly quoted bid and ask price. The main purpose is to profit from the bid-ask spread, while ensuring tight risk control over inventory positions and minimising the risk of transacting with an informed counter-party"
- HFT houses use them as a profit strategy in some markets. Broker-dealers usually automate with them the market-making of transactions of small volume, so their traders can concentrate on those trades with larger volumes and potential profits.

• **Intraday investment:** "Strategies that attempt to exploit systematic short-term patterns in asset prices or arbitrages".

- For example, HFT algorithms identify short-lived trading opportunities from price and order flow information and act on such signals quickly.
- HFT may also implement strategies identifying arbitrage opportunities in markets with fragmented liquidity

Algorithmic execution. A primer.

Aim: *execute (buy or sell) a (large) volume in a financial instrument, typically in markets based on a limit order book trading mechanism*

Low cost executions are **key** for most of the players in the financial markets:

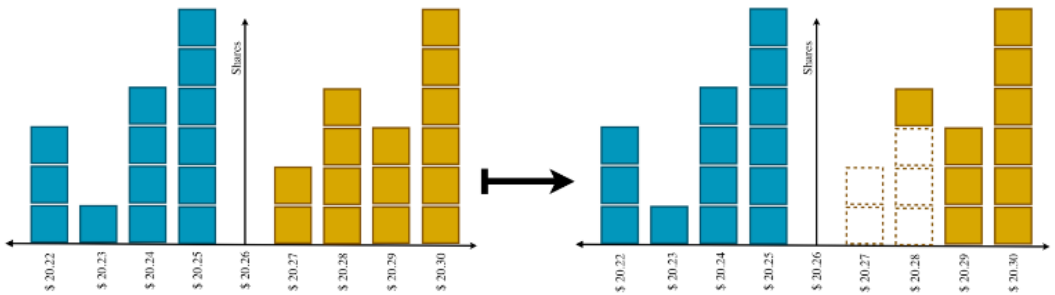
- If execution costs are high, trading strategies (algorithmic or not) that are seemingly profitable might really not be it. This affects **banks** that are hedging their market risk, and investors like **mutual and hedge funds** that need to get returns from their portfolios
- **Banks** usually perform execution of block trades on behalf of clients, that rely on their expertise and low fees access to markets. In a competitive market, these banks need to reduce execution costs to attract clients.

Algorithmic execution. A primer.

- For **small orders**, there is a **trade-off** between using market orders and therefore crossing the spread, or using limit orders to get a better price but risking not executing the order
- For **large orders**, there might be not enough liquidity in the best bid/offer even to execute with a market order without having to take liquidity at worse prices.
  - **Execution algorithms** are used in this case to split the order in smaller parts that are executed gradually in time, with limit or market orders. The most simple being equally distributing the order over a time window, or “Time Weighted Average Price” (TWAP algo)
  - They face a different **trade-off** between executing more quickly causing a “market impact” or executing more slowly and having “market price risk”
    - *Temporary market impact* is caused when worse prices than best bid/offer are obtained due to large market orders
    - *Permanent market impact* is caused when the market gets a signal that a large order is being executed, and agents reposition their limit orders to get better prices

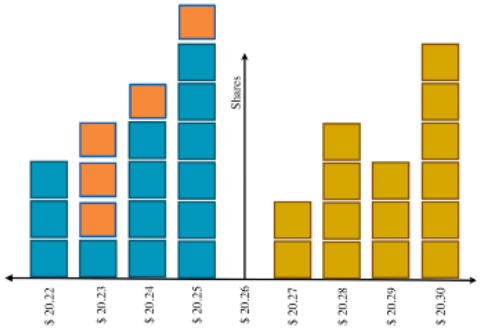
Algorithmic execution. A primer.

> Market Order



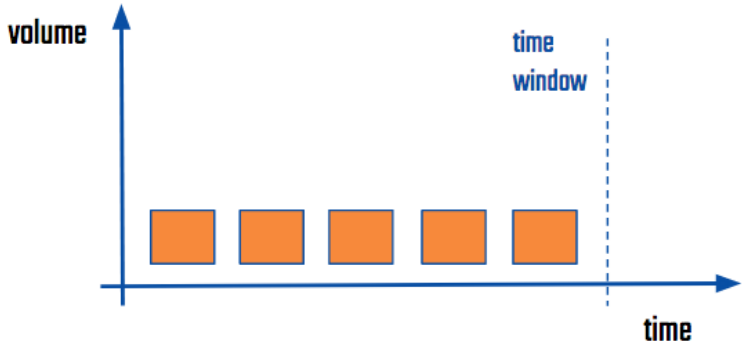
Price almost guaranteed, but market impact (bad price)

> Limit Orders



Price guaranteed, execution is not

> TWAP



Price not guaranteed, more price risk but less market impact, execution guaranteed

Algorithmic market making. A primer.

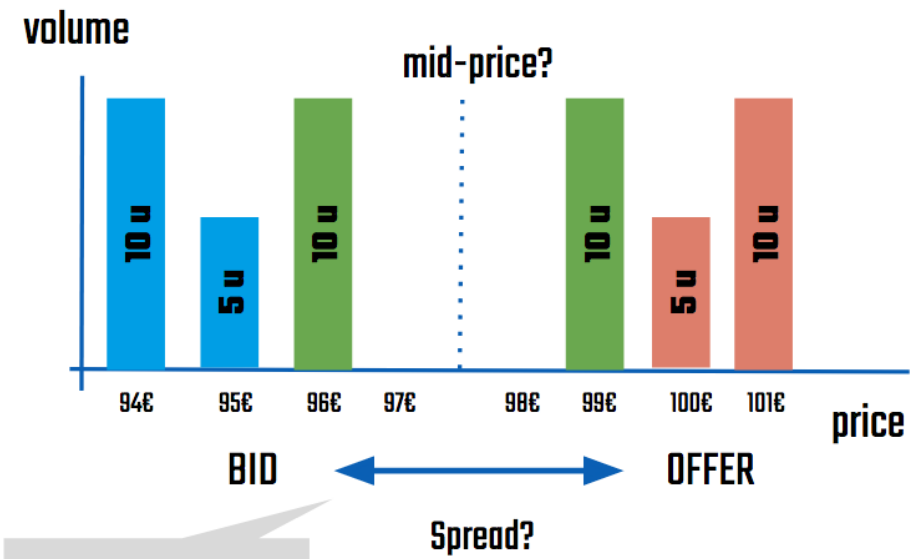
Aim: *provide "liquidity" to a financial instrument, or immediacy for trading it at a reasonable price.* The dealer or market maker stands ready to buy/sell financial instruments when the investors need it, the compensation for this service being the bid-offer spread (buy cheaper than sell)

- The service is similar to that provided by dealers in non-financial markets. For example a car dealer buys and sell used cars whenever the seller/buyer needs it, but it will buy them cheaper and sell them more expensive than the "fair price" (price that one could get bypassing the dealer)
- The bid-offer pays for the service of immediacy. It also compensates for the risks of the activity:
  - **Inventory risk**: in order to be able to buy and sell financial instruments at any moment, dealers must maintain a minimum inventory. This inventory, if not hedged, carries risk if prices move so the inventory loses value.
  - **Information asymmetry**: by standing ready to buy and sell from/to investors, the market maker has the risk that investors have more information and are trading ahead of the news. For instance, the market could be selling because of information that the market maker doesn't have, therefore inadvertently accumulating a risky position.

Algorithmic market making. A primer.

- **Algorithmic market making** is the automation of parts or the totality of this provision of liquidity. A fully automated market making algorithm:
  - Selects automatically the buy and sell quotes with respect to a mid or fair price. This requires calculating this fair price, the bid-offer spread and the skew (asymmetry with respect to the fair price)
  - Adjusts these quotes automatically based on market conditions, the client with which is trading (if known) and the inventory accumulated
  - Auto-hedges the inventory with correlated instruments to reduce the risk of holding it
- Algorithmic market making is used by **banks** (normally for trades of small volume), n**ew liquidity providers and HFT houses**
- There are strategies for **quote-based, order-based** and **hybrid** markets

Algorithmic market making. A primer.



Assets	Liabilities
Swap	Swap
Swap	Bund Future
Swap	Bund Future

Quoting:
mid-price + spread

Hedging inventory risk

Execute hedges in Trading Venues

Algorithmic intraday investment. A primer.

Aim: *generate returns by exploiting short-term patterns or arbitrage opportunities in the market.* Typically used by intraday investors and hedge funds, although banks might exploit them as well in their activity as dealers, for instance to optimise their inventory and hedges.

- **Momentum strategies:** strategies that try to detect and benefit from trends in the market, typically in prices.
  - Momentum directional strategies try to detect trends in the market anticipating the effect of news and events.
- **Mean reversion strategies:** strategies that try to detect deviations in market conditions that will likely revert to average conditions.
  - Relative value mean reversion strategies focus on identifying discrepancies in prices that share similar economic or financial characteristics, or just because the historical correlations between the price series. The idea is that these discrepancies will tend to disappear in the future.
- **Arbitrage strategies:** strategies that try to detect misprices of financial instruments, for instance between derivatives and their underlyings, between prices of the same instrument in different markets, etc

IV. Developing trading algorithms

- Define the **target** of the algorithmic strategy: is it order execution in a single stock exchange? Is it returns generation in the bond futures markets? etc
- Define a way to evaluate the performance of your algorithmic strategy and compare the performance of different algorithms, typically by using some measurable and unambiguous **benchmarks**
- Understand which kind of **market data**, historical and real-time, you will have access to, and the **latencies** both of market data and order placement. The type of strategy suitable for a given target depends heavily on this.
- **Design and prototype** the basic strategy to complete the target. Different ways:
    - Mathematical optimisation tools like Dynamic Programming. For instance the Almgren - Chriss framework for optimal execution
    - Data-driven machine learning algorithms like (Deep) Reinforcement Learning
    - Human trading expertise codified in rules
    - Heuristics
- Depending on the algorithm, **training** with historical or synthetic data may be required: to calibrate some parameters or indicators, or for the whole strategy if is fully data-driven

IV. Developing trading algorithms

- **Test** the performance of the algorithm in historical or synthetic data. Be aware not to use the same data as for calibrating and training the algorithm
- Implement the **productive version** of the algorithm (if necessary)
- Run the productive version in **test environments** (internal and from the markets)
- **Deploy** the algorithm gradually (small volumes, subset of instruments/markets, …) and **monitor** for signals that the performance is not aligned with test results.
- Keep **monitoring** the algorithm (real-time and post-trade): financial market conditions change suddenly and what it used to work could stop working.

Typical benchmarks for algorithmic strategies

Execution strategies

- **Close**: the mid-price of the market when the strategy finishes
- **Market Order Close**: the cost of a market order when the strategy finishes, using the notional of the strategy
- **Open-High-Low-Close average**: mean of mid prices at open, high, low and close in the period the strategy executes
- **Time Weighted Average Price (TWAP)**: arithmetic average of market prices during the window of execution
- **Volume Weighted Average Price (VWAP)**: average of market trade prices weighted by their volume during the window of execution
- **Decision Open**: the mid-price of the market when the strategy was launched
- **Market Order Decision Open**: the cost of a market order when the strategy was launched, using the notional of the strategy
- **Arrival Open**: the mid-price of the market when the first order of the strategy reached the market
- **Market Order Arrival Open**: the cost of a market order when the first order of the strategy reached the market, using the notional of the strategy

Typical benchmarks for algorithmic strategies

Market making, momentum, mean-reversion and arbitrage strategies

- **Profit & Loss**: running and final profits or losses of the strategy, starting with an initial inventory or position, and running without further injections of money
- **Sharpe Ratio**: average return earned in excess of a risk-free rate proxy per unit of total volatility risk. It helps identifying strategies that have extra returns because they incurred in higher risk.

$$S_a = \frac{E[R_a - R_b]}{\sigma_a}$$

- **Beta:** indicates whether an strategy is more or less volatile than the market as a whole. Helps to benchmark an strategy against the general performance of the market.
- **Maximum Drawdown**: maximum loss from a peak to a through of an strategy, before a new peak is achieved. Indicator of downside risk of an strategy.
- **Sortino Ratio**: variation of the Sharpe ratio where instead of overall volatility of an asset, it uses the standard deviation of negative returns, differentiating between overall volatility and harmful volatility.

$$S = \frac{R - T}{DR} \qquad DR = \sqrt{\int_{-\infty}^{T} (T - r)^2 f(r)\, dr}$$

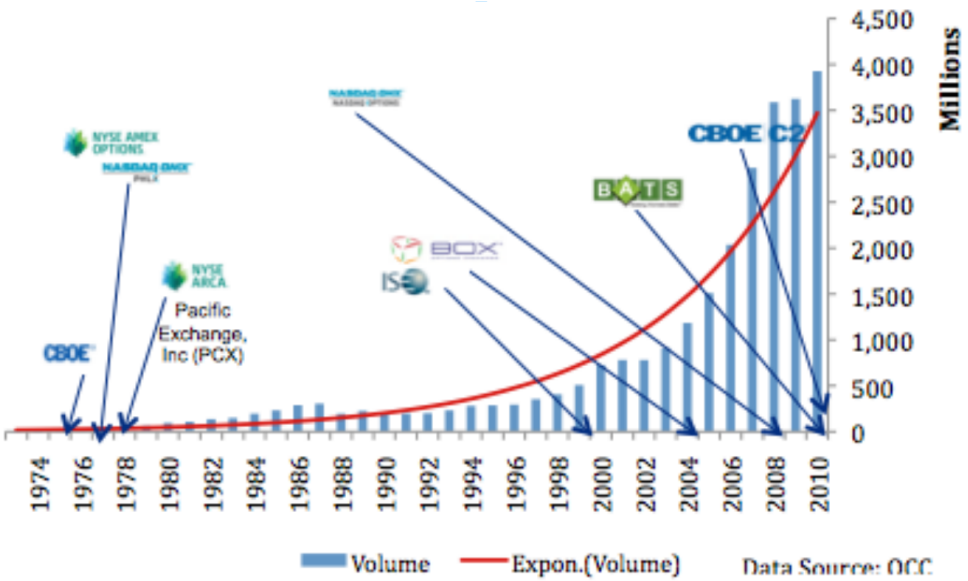
- **Omega Ratio:** relative measure of the likelihood of achieving a certain return. It is the ratio of the cumulative probability of returns of an strategy above a minimum return defined by the investor, to the cumulative probability of returns below it.

$$\Omega(r) = \frac{\int_r^{\infty} (1 - F(x))\, dx}{\int_{-\infty}^{r} F(x) dx}$$

V. Algorithmic Trading infrastructure

A good algorithmic trading infrastructure should have the following **requirements:**

- **High speed / low latency:** delays in transmitting data and orders from the infrastructure to the market. How low is the latency necessary depends on the type of algorithm and the market and instruments.
- **Capacity:** electronic and algorithmic trading are increasing dramatically the number of messages, due to smaller sizes of orders and more frequent updates. The infrastructure needs to be able to cope with a large number of messages.
- **Resiliency:** the infrastructure must have ways to monitor the performance of the algorithm, and mechanisms to handle failures in the infrastructure itself and the algorithms, for instance allowing for a quick and orderly disconnection from the market if necessary

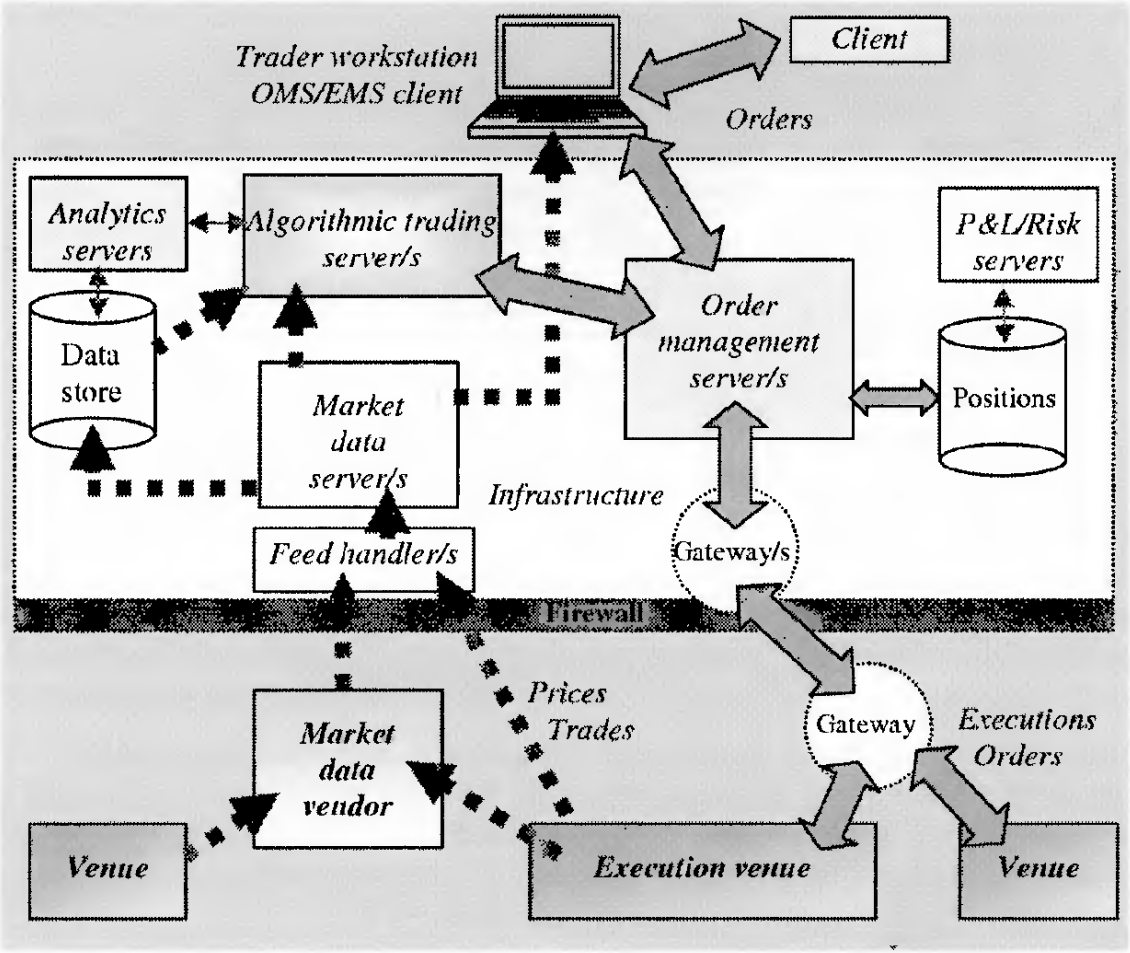


Typical infrastructure for specialised companies

- **Algorithmic trading server**: a platform to run the real-time algorithms
  - It could or not be the same one where prototyping is done
  - Ideally it is the same where testing and estimation of performance is done
- **Market data server**: platform that provides historical data to test and calibrate the algorithms, and real-time data to feed the trading algorithm so it can react to market conditions. Data can come directly from the market, or via a market data vendor
- **Order management server** (OMS): platform that handles the orders produced by the algorithm, sending them to the markets via gateways, monitoring their status, booking the trades in the systems, etc
- [Optional] **Execution management server** (EMS): specialised platform to handle the execution of orders in the markets, typically offering a suite of execution algorithms. Alternatively, execution logic could also reside in the algorithmic trading server
- **Trading workstation**: main interface for the trader, aggregating client orders from electronic platforms, allowing to launch trading algorithms, monitor the algorithms and orders via the OMS, do pre-trade and post-trade analysis, showing market conditions, etc
- **Analytics/backtesting server**: uses historical or synthetic data to calibrate the indicators of the algorithms and perform backtesting and stress testing. It takes historical data from a Data Store
- **P&L and risk server:** use the information of the positions taken by the algorithms at any time to calculate profit & loss and risk metrics.

Sketch of typical infrastructure

Backtesting



Source: Algorithmic Trading & DMA, Barry Johnson

Algorithmic Trading Server

- Also called in some contexts “strategy container”
- It runs the algorithmic trading strategies, producing the “child” orders that are sent to the actual trading venues and reacting to market data changes
- There are different approaches to build this component:
  - Use a **third party server** that includes a suite of third party algorithms
  - Use a third party server that provides **tools to write proprietary strategies**, with different options from support of traditional programming languages to complex graphic interfaces
  - Use an **internal server** and **in-house proprietary tools**
- Typical programming languages to write strategies are J**ava, C#, C++** for higher performance code, and Python, Matlab for lower performance code and prototyping
- Normally the servers use complex event processing logic. There is a trend now in using “**reactive programming**” principles to implement algorithmic strategies
- Large brokers, dealers and hedge funds build their own algorithmic trading servers, and have dedicated teams to design the algorithmic strategies, since third-party algorithms cannot usually achieve the differentiation and customisation that in-house ones have
- Smaller houses usually buy third-party platforms
- Even if the algos are developed in-house, the execution layer is sometimes outsourced to third-party solutions, thanks to the proliferation of “Execution Management Systems” (EMS), that include connectivity to markets & brokers, and standard execution algorithms.

Market Data Server (MDS)

- A Market Data Server is a central point to handle historical and real-time data
- **Historical data:**
  - Typically stored in a “data store”, “data lake”, “data repository”.
  - This data is used for calibration and backtesting of the strategies
  - Traditionally, for algorithmic trading, KDB+ has been the standard choice by the industry, due to its in-memory performance and specialisation in handling time series. The cons are high license costs and the need for very specialised developers
  - Nowadays, Big Data distributed database tools like Hadoop are catching up, specially since the development of Spark, which speed ups calculations due to its in-memory capabilities
  - A historical database can be built in-house, from the real-time data feeds, although it requires a very resilient system to avoid having gaps or errors in the data stores.
  - Alternatively, one can buy historical data from vendors like Thomson Reuters and Bloomberg
- **Real-time data:**
  - It can be obtained from market data vendors (Reuters, Bloomberg) or directly from the trading venues
  - Market data vendors have as advantages: 1) single point to get the data, they provide with all the connectivities to exchanges 2) standardised inputs, 3) resilience
  - However, connecting directly to the trading venues has the advantage of lower latency
- For professional MDS, it is recommended to have at least a couple of alternative data sources

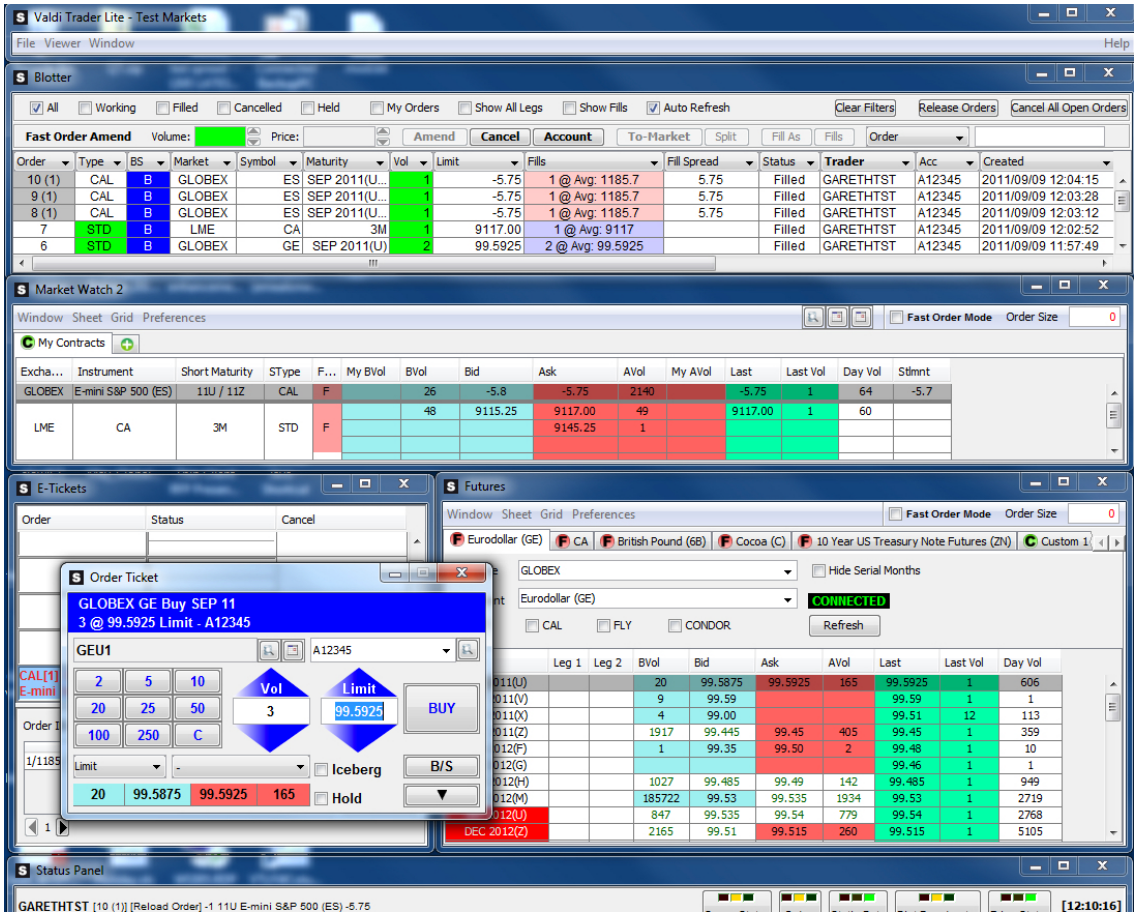
Order Management System (OMS)

- OMS provide the basic functionality to handle all the orders sent to the trading venues: order entry and order routing (encoding + transmission)
- They usually pack extra functionality like pre-trade and post-trade analytics, real-time benchmarks, risk management, and connection to back-office systems for booking and reconciliation
- Some examples:
  - Bloomberg: TOMS (Trade Order Management Solutions) for fixed income, SSEOMS (Sell Side Execution and Order Management Solutions) for equities, AIM (Asset and Investment Management) for the buy-side
  - Fidessa OMS: equities, equity options, derivatives, US Treasuries, …
  - FlexTrade ColorPalette: equities
  - FIS (previously SunGard) Valdi: multi-asset

http://www.tradersmagazine.com/gallery/sellside-order-management-systems-an-overview-113407-1.html

Order Management System (OMS)

Order entry: it can be manual or via algorithms. The OMS typically has a client to enter manual orders or launch algorithms, or it is integrated with the trading workstation.



FIS Valdi OMS client

Order Management System (OMS)

Order routing: composed of two parts:

- Encoding: a protocol to transmit the information about an order
  - In trading pits, there was a hand code to communicate orders
  - In electronic trading, the standard nowadays is to use the FIX ("Financial Information eXchange") protocol, although some trading venues also offer their own proprietary APIs
- Transmission: using FIX engines to connect clients to dealers and brokers, dealers to other dealers and exchanges, etc

Example FIX messages

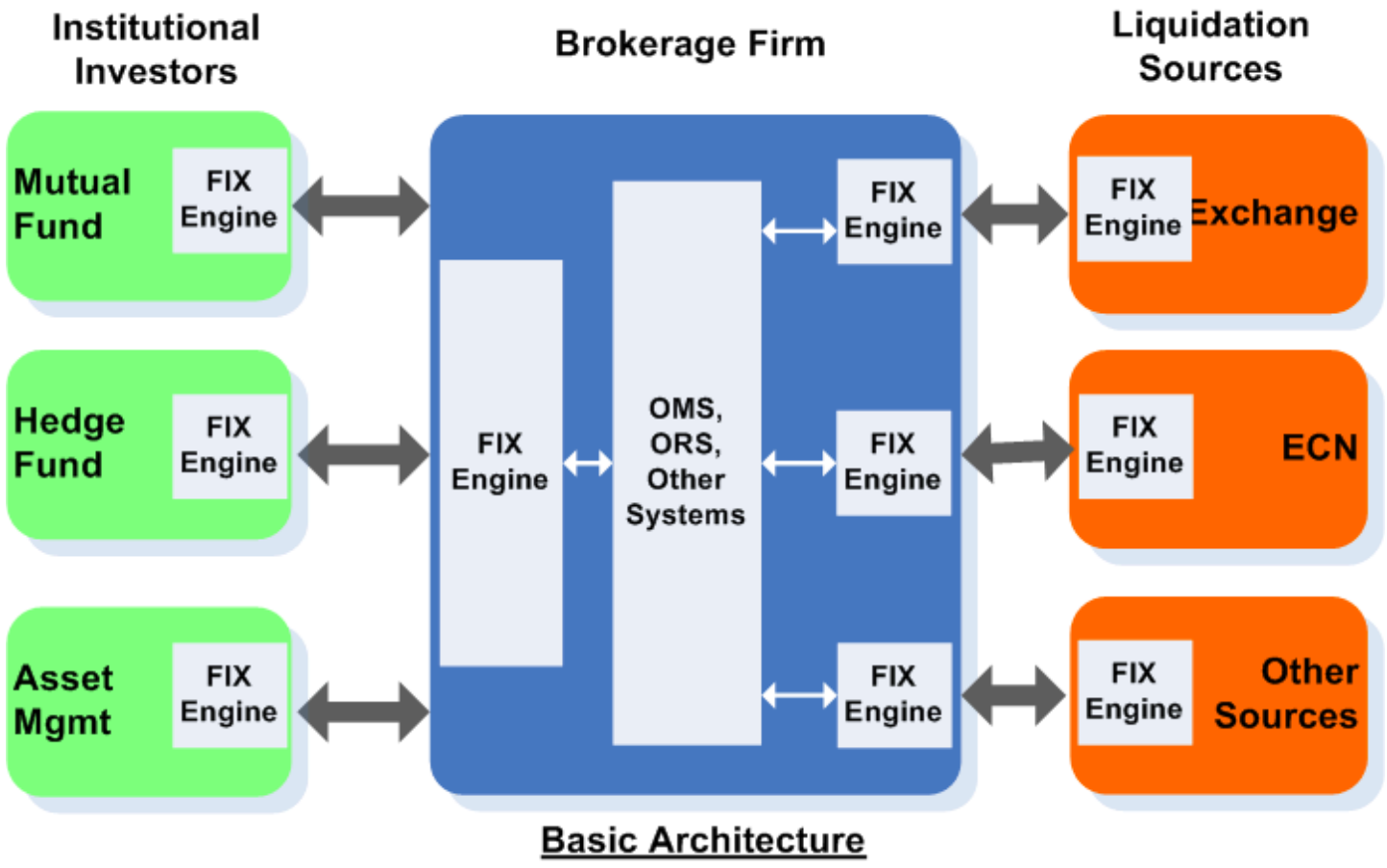
A FIX 4.2 message that would be seen in a FIX engine log file for an Order for 10000 shares of IBM.

8=FIX4.2|9=0132|35=D|57=Simulator|34=2|49=SLGM0|56=SB10|52=20100315-13:45:28|55=IBM|40=2|38=1000|21=2|11=order 0|60=20100315-17:45:20|54=1|44=110.5|10=097

The message fields are delimited using the ASCII 01 <start of header> character.

Tag 8 FIX version	Tag 55 Symbol
Tag 49 SenderCompID	Tag 54 Side
Tag 35 MessageType	Tag 38 OrderQty

Order Management System (OMS)



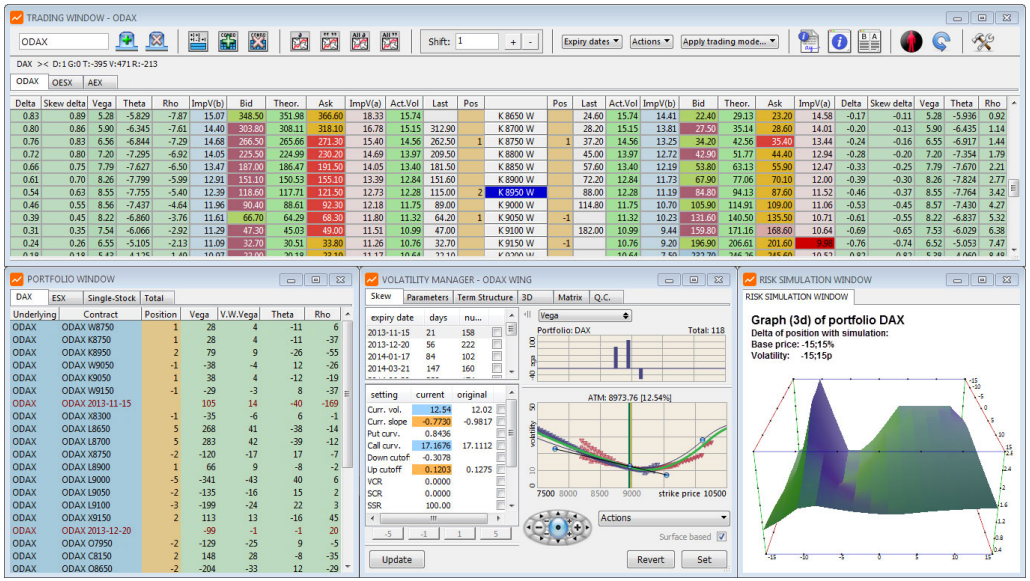
Basic Architecture

Execution Management System (EMS)

- EMS provide functionality to execute orders and launch algorithms
- There share a lot of functionality with OMS, and nowadays they are converging in products called OEMSs.
- They are more focused on the execution capabilities: interface for order and algorithm entry, connectivity with multiple exchanges, brokers and platforms, pre-trade and post-trade analysis, real-time monitoring, etc.
- Algorithms might come as part of the EMS, or the EMS is connected to brokers/dealers that offer their own suite, which can be launched from the EMS
- EMS also typically support Direct Market Access capabilities
- OMS are typically more integrated with the firm systems (“inward looking”) whereas EMS offer light interfaces and connectivity (“outward looking”).
- Whereas a firm would use only a OMS, many EMS can potentially be used.
- Some examples:
    - Bloomberg: TSOX for fixed income, EMSX for equities, FXGO for forex
    - TradingScreen: TradeSmart OEMS
    - Portware Enterprise: multi-asset

Market Making Trading Platforms

- Trading software specialised in the market-making activity of broker-dealers and new liquidity providers. It typically offers:
  - Management and aggregation of Request for Quote and Request for Stream coming from different client-to-dealer platforms
  - Pricing and quoting engines
  - Auto-quoting and auto-hedging capabilities
  - Integration with OMS and EMS
- Examples:
  - ORC Trader for listed derivatives (now integrated within Itiviti)
  - Itiviti's T-bricks
  - ION Trading
  - Numerix One View Distribution



Analytics / backtesting server

- It handles historical data from the Market Data Server
- It provides with **off-line analytics** for the algorithmic trading server, typically for:
  - **Calibration** of indicators and **parameters** of the trading algorithms
  - Evaluating the **performance** of the trading algorithms
- The analytics server can use either:
  - **Historical** data from the Market Data Server
  - **Synthetic** data generated with a model (which can be also calibrated with historical data).
- **Historical vs Synthetic data:**
  - Historical data corresponds to real market scenarios, whereas synthetic could generate very unlikely scenarios, biasing the algorithms or the performance evaluation
  - However historical scenarios might not repeat themselves, but synthetic could potentially be tuned better to reflect expected future conditions
  - Historical data is limited to what the database contains (and of course to history), whereas infinite synthetic data can be potentially be generated
  - Historical testing of algorithms cannot include the reactivity of the market to our own algorithms, whereas on-line synthetic data generation can
- Analytics / backtesting servers are either in-house developments or come bundled with algorithmic trading servers

P&L and risk server

- They track the real-time performance of the trading positions, algorithmic or not, providing with profit & loss metrics, risk/exposure indicators, etc
- They are usually the same systems used for P&L and risk for all the trading operations of the company, although some vendors offer specialised algorithmic trading capabilities
- They can be developed in-house or bought from third-parties
- Examples of third-party vendors:
    - Calypso Risk
    - Murex MX.3

Infrastructure for small companies and private investors

- Small companies and private investors normally buy **third-party solutions** that bundle algorithmic trading servers, analytics/backtesting platforms, connectivity to brokers and exchanges, etc
- Examples:
  - Software companies: Deltix, QuantHouse, AlgoTrader, Qbitia, MetaTrader
  - Brokers: TradeStation, Interactive Brokers (IB)
  - QuantConnect: https://www.quantconnect.com, offers a web-based algo trading platform with free backtesting capabilities and real-time trading via associated brokers.
- There is a popular alternative to write trading algorithms: **"crowdsourced hedge funds":** they offer a free platform to code and backtest trading algorithms in popular programming languages like Python. The best algorithms in a backtesting or paper trading (real-time faked trading) might receive prices, or get included into the hedge fund portfolio with shared profits
- Examples:
  - Numerai: https://numer.ai
  - Quantiacs: https://quantiacs.com/
- Nowadays, a basic algo trading platform can be built using open-source libraries, cloud infrastructure like AWS and Google Cloud, and trading via APIs offered by broker accounts
  - Examples of open-source libraries: PyAlgoTrade, Zipline (backtesting), Backtrader (backtesting), QSTrader (backtesting), Alpha lens (alpha factors research), Pyfolio (portfolio research)
  - Examples of cloud infrastructure: Google Cloud, Microsoft Azure, AWS
  - Examples of broker APIs: Interactive Brokers. For cryptocurrencies, direct access to the Exchanges APIs is available in some cases

Infrastructure for small companies and private investors

QuantConnect interface

VI. Testing trading algorithms

Typical workflow should follow these lines:

1. Historical backtesting of the algorithm
2. Adverse scenario testing: historical and simulated if possible
3. Test in real-time with test environments: local and from the exchanges if available
4. Deploy in production in a limited scale: small volume, subset of instruments/markets, …

Historical backtesting

- Get data from Market Data Server / Vendor / Plugin. **Granularity** of data depends on the kind of algorithm (tick data, OHLC, EOD, …)
- **Clean data** before using it for backtesting. Some of the things to look at:
  - **Outliers**: very extreme data points that can come either from data corruption or because they were actually rare days. From the latter, typically predictable rare days are taken out (eg day after Brexit, day after Trump election…) but not unpredictable ones (eg Flash Crashes)
  - **NaNs**: not a number, or equivalents in other languages, typically from missing data
  - **OTC Trades**: in Limit Order Book data, exchanges usually add OTC trades later to the historical data series, but this is not liquidity available in real time
  - **Survivorship bias**: corporations are closed and data is no longer available. If an algorithm is backtested with only companies that have survived until today, the backtesting is not covering the default/bankrupt scenario
  - **Corporate actions** like dividends and splits, that require to adjust the price series
  - For listed derivatives, **roll-overs** to the most liquid series
- Historical backtesting must take into account the **spread** / market order cost and the **slippage:** impact from latencies of data and orders and fees from the exchange

Historical backtesting

Disclaimers from historical backtesting:

- Historical data cannot react to the orders sent by our algorithm, and therefore it does not take into account **market impact**
  - Instantaneous: market order from our algo consume liquidity that is not available anymore, and limit orders provide more liquidity than the historical, changing the impact of market orders
  - Permanent: other agents might react when seeing our orders (because of size, frequency, or other patterns) and adjust their orders
- Interpretation of results is prone to several bias:
  - Survivorship bias: as discussed before
  - Look ahead bias: due to a mistake, the backtesting makes the algorithm use information from the future in its decisions, e.g. when using the same dataset to calibrate the algorithm and to backtest it
  - Data snooping bias / overfitting: the optimal set of parameters that maximises the performance of the algorithm in the historical data might be exploiting a spurious correlation in this dataset
- Markets usually change **regimes** (where stationary statistical properties are different). Backtesting over several regimes might give an average performance that is not representative of performance within a particular regime

Recommendations:

- Use a different training set to calibrate the strategy and to evaluate its performance (training / test)
- Try techniques for regime detection to isolate data from different market regimes
- Take backtesting results as optimistic bounds: if the algo does not work in backtesting, it is unlikely that it will work in reality, but the other way round is not necessarily true

Adverse scenario testing

It has the following aims:

- Test the performance of the algorithm in typical **market stress** situations, e.g. high volatility regimes, sudden crashes, low liquidity shocks, etc
- Find the **worst case scenarios** that affect the performance of the algorithm

How can it be done?

- Using **historical** data:
  - Select days likely to be outliers from a statistical point of view, using indicators like volatility or volume
  - Select relevant historical days: Brexit, Trump, Flash Crash, …
- Using **synthetic** data:
  - Calibrate a model to historical data and stress its parameters to generate adverse scenarios

Disclaimers:

- Historical stressed scenarios are outliers, and therefore might not be representative of future stressed conditions
- Scenarios generated with synthetic data might not be representative of realistic market conditions: the generative model must be carefully selected and calibrated

VII. Effects and risks of the Algorithmic Trading activity

- **Increase of liquidity**: market making strategies from HFT algorithms are based on placing two-sides (bid/ask) limit orders, therefore increasing the liquidity in the markets where they operate, with more volume available and smaller bid/ask spreads.
  - There is a difference, though, with the traditional market making activity of broker-dealers: HFT houses don't have contractual or reputational obligations to provide liquidity, and they tend to remove quotes in stressed markets, contributing to quick falls in price ("flash crashes") since liquidity suddenly disappears

- **Reduction in volatility:** markets with more liquidity typically have less volatility, although this only applies to non-stressed markets as discussed above. HFT activity can potentially increase extreme events, as discussed below, so the net effect in market volatility is not clear

- **Reduction of volumes per trade / increase in the number of trades**: associated to execution algorithms, that typically divide large orders in small parts over time to reduce market impact

- **Reduction of arbitrages across markets and instruments:** relative value and arbitrage algorithms try to benefit from deviations between prices of the same instrument in different markets, or derivative vs underlying prices. This activity has the consequence of reducing these deviations.

- **Amplification of market instabilities:**
  - In a single market, when many algorithms follow the same signals ("**herding**"), or when market making HFT algorithms provide liquidity that is suddenly removed when there is a stress signal, and effect that could be behind many **"flash crashes"**
  - Across markets, since many algorithms (multi-asset execution, arbitrage…) operate simultaneously in multiple instruments and markets, producing quickly **contagions**

- **Market disruption due to misbehaving algos:**
  - **Involuntarily**: due to errors in the implementation of the algorithms or in the trading systems (**operational risk)**
  - **On purpose:** algorithms that try to get profits disrupting the markets in a way that is considered unfair to other financial agents (**market abuse)**

Source: The Economist

Example: pound flash crash 2016

7th of October 2016: overnight the FX rate USDGBP suddenly plunges 6% and quickly recovers

Investigations point out to an **important role of algorithmic trading**:

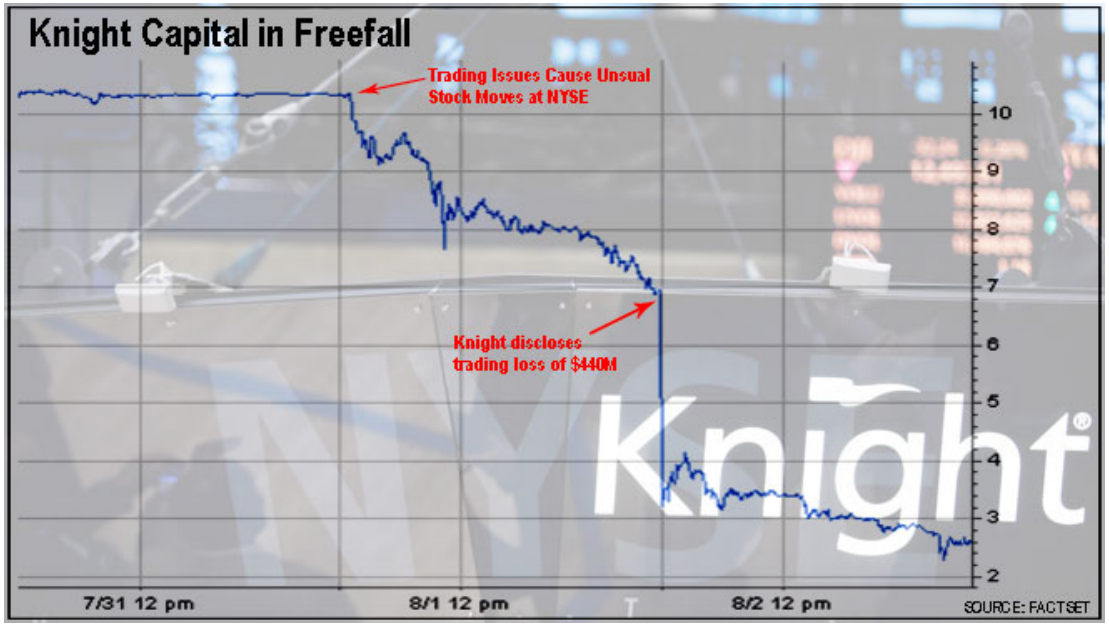
- **Social media reading algos**, reacting to a tweet from Francois Hollande concerning Brexit, might have triggered the sell-off
- **HFT market making strategies** removing quotes and stop loses might have amplified the drop

Not all the factors, though, might have been algorithmic: lack of liquidity during Asian market hours and human traders panicking are also to blame

Source: Bloomberg

Example: Knight Capital

- 1 August 2012, Knight Capital Group loses $440 million in 45 minutes
- Knight Capital was an American financial firm specialised in **market making and electronic execution** for clients
- The loses were caused by a massive number of child orders sent by an algorithm after a **bug in a production** release overnight reached its automated routing system for equity orders.
- **Millions of orders were sent** to several stocks of NYSE. A huge position in these stocks was accumulated. Moreover, loses amplified due to reverse market making quotes (buying high and selling low)
- When the glitch was stopped, Knight had still a large position in stocks, that had to unbundle at a discount in order to clean its balance sheet (Goldman Sachs was at the other side of this trade)
- Due to loses of around 30% of the company's value, its shares dropped largely and the company would be later acquired by Getco LCC



Source: Fox Business

Market abuse with trading algorithms

- **Banging the close**: trade a large number of futures during their closing period to bias the settlement price of the future that day.

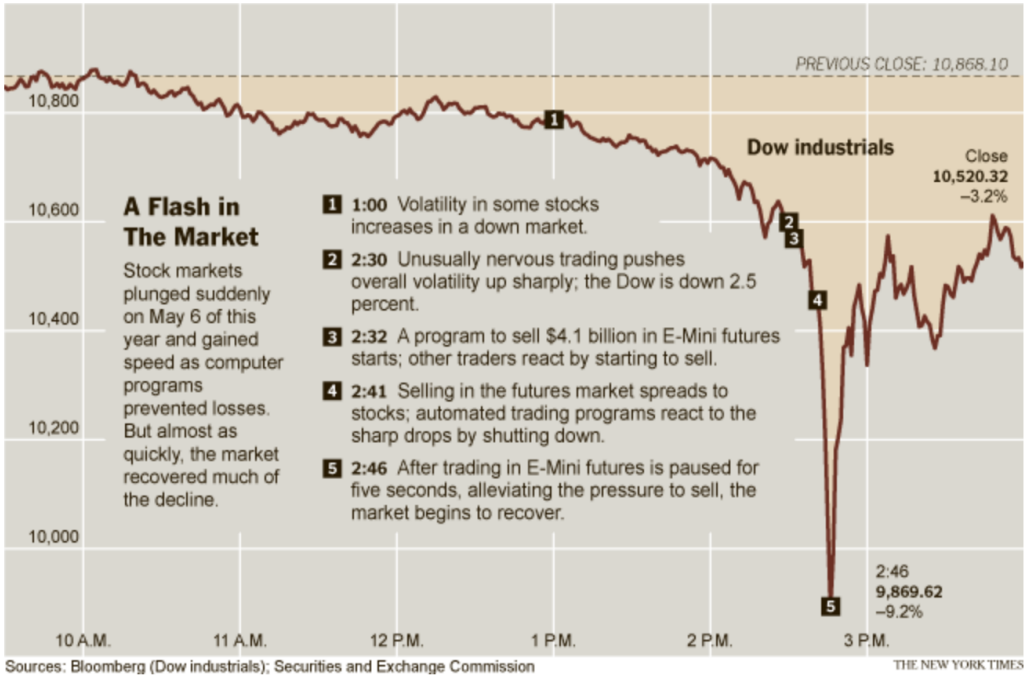
- **Spoofing**: place a large limit order on the best bid or ask and then another order improving the best opposite. This tries to incentive other agents (typically algorithms that look at volume order imbalance as an indicator of short term market move) to cross the spread, filling the improved price order. The large limit orders are then cancelled before they are filled (which is less likely due to priority). This way the spoofer manages to buy or sell at better prices than the available ones.
    - An alternative version waits for the best bid or ask to readjust after placing the large limit order, and then cancel the order and consume the liquidity at new levels with a market order

- **Layering**: similar to spoofing, but multiple limit orders are placed a few ticks apart. This is used typically when the volume at the best bid/ask is thin, an placing a single (fake) limit order is risky

- **Front running**: With side information, sell or buy orders for one's own account before executing a large order for a customer account on the same direction.

Example: 2010 Flash Crash

- On May 6th 2010, within five minutes, **Dow Jones index lost about 10%**. The crash started at 2:45 and lasted about 36 minutes before it partially recovered.
- On April 21th, 2015, **Navinder Singh Sarao**, a London based trader operating from his residence, was convicted of using a trading algorithm seeking to manipulate the market for S&P 500 futures contracts ("E-minis") on the Chicago Mercantile Exchange. In particular, he was accused of using **layering** to manipulate prices
- **HFT algorithms** seem to have been involved in the crash, first by triggering the sell-off following the false signal generated by the layering strategy, then withdrawing quickly from the stressed market



VIII. Regulating Algorithmic Trading

- In the **European Union**, MiFID 2 / MiFIR have introduced an extensive regulation of Algorithmic Trading, High Frequency Trading and Direct Electronic Access
- In the **UK**, after the end of the transition period on 31 December 2020, MiFID 2 is no longer applicable. The FCA has amended some of the directives from MiFID 2, but most of its corpus has been so far maintained
- In the **USA**, there is no such a comprehensive regulation as MiFID 2. Regulation depends on the government body that supervises the firm engaging in Algorithmic Trading:
  - The Security and Exchange Commission (SEC) regulates the securities market. The Regulation Systems Compliance and Integrity (Reg SCI) is the main regulation affecting Algorithmic Trading for entities supervised by the SEC, such as national securities exchanges and broker-dealers via FINRA (the Financial Industry Regulatory Authority)
  - The Commodity Futures Trading Commission (CFTC) regulates the derivatives market. Since 2015, it is working in a comprehensive regulation for Algorithmic Trading, the Regulation Automated Trading (Reg AT), which would have a similar scope that MiFID 2. There is not yet a clear timescale for its adoption.

Main features of MiFID 2 for Algorithm Trading & HFT

Investment firms engaged in Algorithmic Trading activities have the following obligations:

- **Governance**:
  - Have procedures for authorising all the steps in algos development and deployment
  - Have an inventory of algos with all the information required by the regulators
  - Ensure every algo has a unique ID that is attached to every algorithmic order
  - Ensure all the staff performing algorithmic trading tasks has the necessary skills
- **Testing**:
  - Develop test environments and tools to perform conformance testing, adverse scenario and backtesting of all the algorithms that are put in production
- **Resilience**:
  - Develop systems and mechanisms that allow monitoring algorithms on-line
  - Develop systems and mechanism to detect and prevent market abuse
  - Ensure the existence of kill button mechanisms to stop individual algos
  - Pre-trade controls on order entry (manual or algorithmic) must be set in order to avoid sending orders with unintentional parameters that can disrupt the markets
  - Post-trade controls must also exist: order reconciliation with the exchanges, market and credit exposure and market abuse detection

Main features of MiFID 2 for Algorithm Trading & HFT

- **Post-deployment**
  - Set-up an annual self-assessment for the algorithmic trading activity, including review of activities that fall under HFT definition
  - Set-up annual trading platform stress tests: volume and messages
- **High Frequency Trading**:
  - Monthly assessment of activities that qualify as HFT, using a year of data given by the exchanges.
  - Qualifying as HFT has extra requirements:
    - order timestamps have to be down to microseconds
    - maximum UTC offset of 100 microsecond (requiring firms to install a nuclear clock)
    - all placed orders must be stored for five years (executed, cancellation and quotations)
- **Algorithmic Market Making**
  - Sign market making agreements with the Trading Venues when the activity qualifies as market making according to MiFID 2 rules