An in-depth review of the
Fundamentals of Algorithmic Trading

BACK TO BASICS
An Introduction to Algorithmic Trading

What is it?
How to Succeed
The Importance of Backtesting

BY
KRIS LONGMORE

"The future belongs to those who learn more skills and combine them in creative ways."
- Robert Greene

CONTENTS

About the Author

Kris Longmore is a self-taught systematic trading expert with a special interest in machine learning. Kris started blogging on RobotWealth.com to solidify his learnings in algo trading as he went. Robot Wealth soon gathered a dedicated following and has developed into a community of like-minded DIY traders interested in furthering their trading through learning and applying the best practices in algorithmic trading.

Kris left his engineering career behind in 2016 to enter the world of finance full-time. He has worked as a quant at a boutique hedge fund and now runs his own consulting company advising finance companies on how machine learning can transform their approach to the markets.

Kris lives in Sydney, and enjoys long walks on the beach with his wife and exploring the local rockpools with his two kids.

CHAPTER 1:
An Introduction to Algorithmic Trading

At its most basic level, algorithmic trading is simply the automated buying and selling of financial instruments, like stocks, bonds and futures. It requires a networked connection to an electronic exchange, broker or counterparty, and a means of programmatically buying, selling and performing other tasks related to trading, such as monitoring price action and market exposure.

Algorithmic trading is enabled thanks to the rise of electronic exchanges – a relatively recent phenomenon. Once upon a time, financial products were traded in the so-called 'pit' inside the exchange building using the 'open outcry' method. This consisted of brokers and traders being physically present in the pit and shouting prices at which they were willing to buy and sell. Participants even used hand signals to convey their intentions. This gradually began to give way to telephone trading and eventually to electronic trading. The shift started sometime in the 1980s and continues to this day, however the vast majority of exchanges around the world are now electronic.

Naturally, this evolution changed the dynamics of the trading process. Anecdotally, pit traders could sometimes read each other's intentions through the physical contact that comes with being in the pit – obviously this is incredibly implausible when market participants trade electronically and can be separated by potentially vast spaces. Stories of life in the pit makes for interesting and often amusing reading.

Some curated links:

- The Chicago Business School compiled the reflections of a number of pit traders
- This post makes for an interesting insight into life in the pits
- As do these 'Tales of the Pit' and this blog about pit trading

Exchange-traded Markets vs Over-the-counter Markets

It is also worth noting that algorithmic trading is not just for exchange-traded markets: over-the-counter (OTC) markets are also traded algorithmically. An OTC market is one where orders are not executed through a central exchange, but rather between two parties. OTC algorithmic trading typically takes place via an Electronic Communication Network (ECN) or dark pool. The former is typically used by market makers to disseminate and match orders with their network of counterparties. The latter is more like a private execution venue where liquidity is provided by the participants of the dark pool, away from the exchange.

Pros and Cons of Algorithmic Trading

Advocates of electronic trading point out the attendant increased market efficiency and reduced opportunity for manipulation. Electronic trading is also typically less expensive and with the advent of cheap Internet, is accessible to anyone with a decent connection. This means that an individual can buy or sell a financial product from their living room.

It must also be pointed out that as electronic trading has taken off, the instance of **'flash crashes**' – huge spikes in volatility over short periods of time – has also increased. A case can be made that suggests that algorithms exacerbate such a crash because they act much faster than a human can intervene. But on the other hand, exchange operators are finding ways to handle this new environment in safer ways, for example electronic mechanisms to curb extreme volatility, order routing co-ordination between exchanges and re-thinking the role of market makers. Whether it is fair to blame flash crashes on electronic trading is a huge and sometimes contentious topic.

Algorithmic Trading in a Nutshell

In order to execute trades algorithmically, we use a computer program connected to the exchange (either directly or via a broker) that executes our desired trading behavior on our behalf. Such a program or algorithm is simply a set of detailed instructions that a computer understands. A trivial example might go something like:

> *"read some price data, calculate its mean and standard deviation, then if the most recent value of the price data is above its average and the standard deviation is less than some threshold, send a buy order to market."*

That's a trivial example and of course most trading algorithms are much more complex, but you get the idea.

The simple algorithm described above had some of the common aspects of an algorithmic trading system:

- A method to acquire data (“read some price data”), noting that this in itself could be quite a complex standalone algorithm and requires connection to a source of market data, usually in real time.
- Some analysis of that data (“calculate its mean and standard deviation”).
- A means of checking if some condition has been fulfilled based on the previous analysis (“if the most recent price is above its mean and the standard deviation is less than some threshold”).
- Execution of the trading logic, which again can be quite a complex standalone algorithm requiring a means of communicating with a broker or exchange, managing that communication link and keeping track of orders and fills.

Other common components of such systems include:

- Risk management modules, for example position sizing calculations, exposure tracking and adjustment, and tools to track a system’s performance and behaviour.
- Portfolio management tools, which are somewhat related to the above.
- Data handling and storage.
- Post-trade reconciliation and analysis.

Confusing Terminology

The assumption behind the description of the algorithm above is that its purpose is the implementation of a signal-based trading system which essentially follows the fundamental logic of “if this happens, buy, and if that happens, sell.” The term algorithmic trading can actually have a slightly different nuance, particularly in an institutional setting, that it pays to be aware of. In this setting, algorithmic trading can refer to the automated splitting of a large order to get the best execution possible. Such algorithms typically split up a large order into smaller pieces and send the pieces to market in a way that optimizes the overall cost of the transaction.

Consider a large hedge fund that wants to hold a significant position in a stock, say a couple of percent of its market capitalization. Buying such an order in one transaction would have an impact on the price of that stock (depending on its liquidity) that would be disadvantageous to the fund. Entering the position gradually in line with the capacity of the stock to absorb the order often reduces the overall cost of the transaction, sometimes significantly.

Many in the institutional space will refer to algorithmic trading as the type that splits up a big order. These same folks will typically refer to the signal-based system as an automated trading system. In my experience, the terms are largely used interchangeably, and it therefore pays to understand the context when talking about algorithmic trading.

I generally use the terms systematic, algorithmic and quantitative trading interchangeably to refer to strategic trading algorithms that look to profit from market anomalies, deviation from fair value, or some other statistically verifiable opportunity.

Different Strokes for Different Folks

The type of algorithmic trading that most readers of Robot Wealth are interested in is the kind that seeks to identify opportunities to profit by buying low and selling higher. This is the signal-based type of system mentioned above. Within this broad category, there are different subsets of trading algorithm. While there is no one accepted nomenclature for classifying algorithmic trading systems, they can generally be described as follows.

Technical Analysis

Technical Analysis (TA) refers broadly to the analysis of patterns of price and volume to predict future market movement. It is therefore based on the assumption that there exist repeating patterns in the price action of a market. The TA toolkit consists of a collection of indicators (price/volume transformations) like the Relative Strength Index

(RSI) and Moving Average Convergence Divergence (MACD). It also includes the suite of trend lines, support and resistance lines, formations like ‘flag’ and ‘pennant’ and patterns like ‘head and shoulders.’ There is even a catalog of candlestick patterns like ‘engulfing bear’ that allegedly portend the future direction of the market. On the more esoteric end of the spectrum, we have things like Elliot Wave Theory which asserts that markets move in predictable ‘waves’.

While some of the tools of TA are sometimes used in more scientifically rigorous quantitative trading (see below), opinion is divided on whether TA as an approach to predicting the market has any utility; indeed, there are published academic papers that support both sides of this argument (see for example Lo, Mamaysky and Wang, 2000 and Chan, Jagadeesh and Lakonishok, 1996).

Quantitative Trading

Quantitative trading means different things to different people. For some, it may be simply another name for TA-based trading. For me, the distinguishing feature of quantitative trading is the removal of subjectivity (decisions are based on quantifiable information). This implies that quantitative trading is based on some sort of mathematical or statistical model of market behavior.

There are an infinite number of models for market behavior, however finding one that is accurate enough to generate profits is no trivial endeavor. Sometimes, a tool from the TA world might be used in a quantitative model, hence the cross-over that I mentioned above. An example of a popular model is the **cross-sectional momentum strategy**, which essentially boils down to buying winners and selling losers. Another popular one is the **mean-reversion strategy**, which essentially equates to selling winners and buying losers.

As we move along the complexity scale, we might encounter a **cointegrating pairs model**. In this model, we find a pair of securities that can be combined such that together they form a mean-reverting series. Such strategies are appealing because they can be engineered to keep the trader market neutral, or close to it, with the goal of minimizing market risk. Just be careful with the underlying assumptions of such a strategy – that is, that a past cointegrating relationship will continue into the future.

We can also build quantitative models based on fundamental data. Fundamental data like earnings announcements are just numbers, and we now have the tools to efficiently and automatically process the news releases and company filings from which these numbers are taken.

There is a subset of quantitative trading that is currently undergoing a huge upsurge in interest. I am of course referring to **machine learning and artificial intelligence**, which seems to have captured the imagination of both technologists and lay people around the world. With good reason, I might add. At its most basic level, machine learning is simply the derivation of insights from data using statistical models. As such, linear regression can be considered a low-level machine learning algorithm. Today we have much more complex tools, such as artificial neural networks and deep learning, support vector machines and boosting algorithms. Such tools are already widely used to support business decision-making and improve the performance of complex systems. It seems a very natural and obvious step to apply these tools to the markets.

The reality is that while such tools are incredibly powerful, it is difficult (but not impossible) to use them to model the markets directly. Applying the 'classic' data science approach generally doesn't work too well on financial data, at least in my experience. The more celebrated machine learning applications in finance seem to be around efficiently extracting insights from large and complex non-market data sets, like libraries of satellite images, social media feeds and other proprietary and open data sets. Entire books could be written about this topic, but if you are really interested in machine learning, there is enormous scope to apply it to financial decision making – just don't expect an easy ride.

High Frequency Trading

High Frequency Trading (HFT) is admittedly not something I have a lot of direct experience with, but I have worked with folks who are directly engaged in the activity. HFT by definition must be algorithmic since it occurs on the scale of microseconds – or less. No human could engage in HFT without the support of computers. While HFT is generally signal based – that is, something occurs to trigger a buy or sell signal – speed and latency are generally more important than the actual signal itself. The implication of this is that co-location of the algorithm either in the exchange or as close as possible is a prerequisite, and code must be optimized for speed and usually written in a low level language like C++. This results in barriers that are simply too high for DIY traders, and indeed for many trading firms. From my conversations with HFT traders, it is something of an arms race, and an expensive one at that.

Why Should You Care about Algorithmic Trading?

The trend towards algorithmic trading, and automation more generally, has been underway in the institutional trading space for some time. For instance, Bloomberg (2016) recently reported that:

> *"Hedge funds almost doubled their use of algorithmic trading in the forex market last year … executing 61% of their trades via automated computer systems."*

This trend can be seen in other markets too. In 2006, 40% of all orders were entered by algo traders at the London Stock Exchange, rising to 60% in 2007. In 2010, that estimate stood at over 80% (thefinancer.com, 2010).

So why this shift towards algorithmic trading?

Most algo traders that I speak to say that they would never trade any other way, typically quoting similar reasons that relate to overcoming human limitations:

- Computers process enormous amounts of data in the blink of an eye, enabling them to constantly scan dozens or hundreds of markets simultaneously for trading opportunities. A human trader can't keep up with that sort of workload.
- Computers are not prone to the same execution difficulties that humans face: calculation errors, time required to manually enter order details into a broker platform, "fat fingers" errors.

- Computers have no emotional attachment to a trade or a market. Either there is a trade opportunity present or there isn't.
- Computers can execute a system consistently and continuously. Contrast this with a human who is limited to a few hours chart time per day, suffers from fatigue and needs to pursue a social life (not that I would ever advocate leaving an algorithm to run with no human oversight).

What I also find interesting is that most algo traders that I know have an enormous respect for successful manual or discretionary traders. That's because anyone who has made money through algo trading knows precisely how difficult it really is, even with all the advantages of algorithms and automated computation described above. These things are powerful tools for navigating the markets, and folks who can beat the market without them deserve tremendous respect.

Personally, in addition to the advantages listed above, I view algorithmic trading as a means of allocating tasks to the resource most capable of performing them. This needs more explanation:

Reading the points above, you would get the impression that humans are becoming obsolete in the world of trading. Not true! Just as there are many tasks that algos are better suited for, likewise there are certain things that humans are simply brilliant at. We have the creativity to view the markets in novel ways and come up with new ideas for trading systems. We can perform research and stay abreast of the macroeconomic environment in ways that a machine simply can't (for now, at least).

Using algos to trade the markets frees up the trader to pursue meaningful tasks that make use of their skill set. I don't know about you, but I don't get much joy out of staring at charts all day. Nor am I particularly good at distilling meaning from them. My time is better spent researching and overseeing than looking for and executing trading opportunities.

Another nice by-product that arises through algorithmic trading is an automated research and development environment. If a trader writes an algorithm for executing trades in the live market, it is possible to test the algorithm on historical or synthetic data before taking it live. This provides crucial feedback about the algorithm's past performance as well as insight into when, where and why it might fail. Such feedback is difficult, if not impossible, to get with a manually traded or discretionary system.

Obviously I am firmly in the algorithmic trading camp and have so far focused on the advantages that a trader receives by taking this approach to the markets. Of course, there is another side to every story and this one is no different. Some of the drawbacks to algorithmic trading include:

- It requires certain skills which one either needs to acquire personally or rely on others to provide. Programming is the obvious prerequisite, but it is also useful to know about market microstructure and computer hardware, software and networking. The tools that become available through computational trading, like optimisation and machine learning, are incredibly powerful and require specific knowledge to use appropriately. Algo trading is actually very difficult and requires skills from multiple disciplines.

- Algorithmic trading comes with certain infrastructure considerations, such as backup power and network connectivity. This is less of a problem with the rise of affordable managed private servers and cloud-based services, but definitely needs to be considered.
- Hardware dependency – what happens if the server hosting the algorithm goes down?
- A (perceived) lack of control over the behaviour of the algorithm.
- For some traders, the loss of discretion or 'gut feel' that comes with algorithmic trading is problematic.

Set and Forget?

I have also noticed from time to time a misconception that algorithmic trading systems can be simply set up and then forgotten about. This is most definitely not the case! Managing an algorithmic trading system actually represents a significant amount of work and it takes a lot of oversight. Any system that trades at the intra-day frequency would ideally be monitored in real time. Where this is not possible, my personal preference is to have alerts sent to my phone when trades are entered or closed or when my system loses its connection to my brokerage account. Trades also need to be reconciled, preferably on a daily basis. This is important to ensure that the system is behaving as expected, as well as to monitor any deviations between simulated and actual performance. Bugs can and do creep into any computer program and a trading algorithm is no exception!

CHAPTER 2:

How to Succeed at Algorithmic Trading

Where to Start?

There is a lot of information about algorithmic and quantitative trading in the public domain today. The type of person who is attracted to the field naturally wants to synthesize as much of this information as possible when they are starting out. As a result, newcomers can easily be overwhelmed with "analysis paralysis" and wind up spending a lot of their valuable spare time working on algorithmic trading without making much meaningful progress. This chapter aims to address that by sharing the way in which I would approach algorithmic trading as a beginner if I were just starting out now, but with the benefit of many years of hindsight.

This chapter is somewhat tinged with personal experience, so please read it with the understanding that I am describing what works for me. I don't claim to be a guru on personal or professional development, but I did manage to independently develop my algorithmic trading skills to the point where I was able to leave my day job for a career in the markets – so maybe I have some personal experiences and insight that might be beneficial.

If you're new to algorithmic trading, I hope Chapter 1 whet your appetite for finding out more and maybe even convinced you that algorithmic trading is a sensible approach to the markets. In this chapter, we will go a little further and investigate the things that people who are just starting out should think about. In particular, I aim to provide you with something of a roadmap for getting started and making progress as efficiently as possible, whatever your goals might be, by sharing some of the practical things that I've learned along the way. This chapter will cover:

- What to learn in order to succeed;
- How to learn it; and
- Important practical considerations.

What to do in order to succeed

Active doing is so much more important than passive learning.

Learning the theoretical underpinnings is important – so start reading – but it is only the first step. To become proficient at algorithmic trading, you absolutely must put the theory into practice. This is a theme that you will see repeated throughout this chapter; **emphasizing the practical** is my strongest message when it comes to succeeding in this field.

Having said that, in order to succeed in algorithmic trading, one typically needs to have knowledge and skills that span a number of disciplines. This includes both technical and soft skills. Individuals looking to set up their own algorithmic trading business will need to be across many if not all of the topics described below; while if you are looking to build or be a part of a team, you may not need to be personally across all of these, so long as they are covered by other team members. These skills are discussed in some detail below.

Technical skills

The technical skills that are needed for long-term successful algorithmic trading include, as a minimum:

1. Programming
2. Statistics
3. Risk management

There are other skills I would really like to add to this list, but which go a little beyond what I would call "minimum requirements." I'll touch on these later. But first, let's delve into each of these three core skills.

1. Programming

If you can't already program, start learning now. To do any serious algorithmic trading, you absolutely must be able to program, as it is this skill that enables efficient research. Forget "click and drag" type software programs that promise algorithmic trading success without the need to write code, and if any trading educator tells you that you don't need to code, turn around and run and don't look back. Accept that coding skills are prerequisite, and get to it. After a while, you'll probably find that you actually like it anyway.

It pays to become familiar with the syntax of a C-based language like C++ or Java (the latter being much simpler to learn), but to also focus on the fundamentals of data structures and algorithms at the same time. This will give you a very solid foundation, and while it can take a decade or longer to become an expert in C++, I believe that most people can reach a decent level with six months of hard work. This sets you up for what follows.

It also pays to know at least one of the higher-level languages, like Python, R or MATLAB, as you will likely wind up doing the vast majority of your research and development in one of these languages. My personal preferences are R and Python.

- **Python** is fairly easy to learn and is fantastic for efficiently getting, processing and managing data from various sources. There are some very useful libraries written by generous and intelligent folks that make data analysis relatively painless, and I find myself using Python more and more as a research tool.
- I also really like using **R** for research and analytics as it is underpinned by a huge repository of useful libraries and functions. It was written with statistical analysis in mind, so it is a natural fit for the sort of work that algorithmic traders will need to do. The syntax of R can be a little strange though, and to this day I find myself almost constantly on Stack Overflow when developing in R!
- Finally, I have also used **MATLAB** and its open source counterpart Octave, but I would almost never choose to use these languages for serious algo research. That's more of a personal preference, and some folks will prefer MATLAB, particularly those who come from an engineering background as they may have been exposed to it during their work and studies.

When you're starting out, I don't believe it matters greatly which of these high-level languages you choose. As time goes on, you will start to learn which tool is the most applicable for the task at hand, but there is a lot of cross-over in the capabilities of these languages so don't get too hung up on your initial choice – just make a choice and get started!

Simulation environments

Of course, the point of being able to program in this context is to enable the testing and implementation of algorithmic trading systems. It can therefore be of tremendous benefit to have a quality simulation environment at your disposal. As with any modelling task, accuracy, speed and flexibility are significant considerations. You can always write your own simulation environment, and sometimes that will be the most sensible thing to do, but often you can leverage the tools that others have built for the task. This has the distinct advantage that it enables you to focus on doing actual research and development that relates directly to a trading strategy, rather than spending a lot of time building the simulation environment itself. The downside is that sometimes you don't quite know exactly what is going on under the hood, and there are times when using someone else's tool will prevent you from pursuing a certain idea, depending on the limitations of the tool.

A good simulation tool should have the following characteristics:

- Accuracy – the simulation of any real-world phenomenon inevitably suffers from a deficiency in accuracy. The trick is to ensure that the model is accurate enough for the task at hand. As statistician George Box once said, "all models are wrong, but some are useful." Playing with useless models is a waste of time.
- Flexibility – ideally your simulation tool would not limit you or lock you in to certain approaches.
- Speed – at times, speed can become a real issue, for example when performing tick-based simulations or running optimization routines.
- Active development – if unexpected issues arise, you need access to the source code or to people who are responsible for it. If the tool is being actively developed, you can be reasonably sure that help will be available if you need it.

There are a number of options, but for the beginner there is probably none better than the Zorro platform, which combines accuracy, flexibility and speed with an extremely simple C-based scripting language that makes an ideal introduction to programming. The platform is being constantly refined and updated, with improvements being released roughly quarterly. Zorro may not look like much, but it packs a lot of power into its austere interface and is an excellent choice for beginners. I've also personally seen Zorro used as a research and execution tool in more than one professional trading setting. *Fundamentals of Algorithmic Trading* makes heavy use of the Zorro platform and includes detailed tutorials on getting started, aimed at the beginner.

2. Statistics

It would be extremely difficult to be a successful algorithmic trader without a good working knowledge of statistics. Statistics underpins almost everything we do, from managing risk to measuring performance and making decisions about allocating to particular strategies. Importantly, you will also find that statistics will be the inspiration for many of your ideas for trading algorithms. Here are some specific examples of using statistics in algorithmic trading to illustrate just how vital this skill is:

- Statistical tests can provide insight into what sort of underlying process describes a market at a particular time. This can then generate ideas for how best to trade that market.
- Correlation of portfolio components can be used to manage risk (see important notes about this in the Risk Management section below).
- Regression analysis can help you test ideas relating to the various factors that may influence a market.
- Statistics can provide insight into whether a particular approach is outperforming due to taking on higher risk, or if it exploits a genuine source of alpha.

Aside from these, the most important application of statistics in algorithmic trading relates to the interpretation of backtest and simulation results. There are some significant pitfalls – like data dredging or “p-hacking” (Head *et.al. (*2015)) – that arise naturally as a result of the strategy development process and which aren’t obvious unless you understand the statistics of hypothesis testing and sequential comparison. Improperly accounting for these biases can be disastrous in a trading context. While this issue is incredibly important, it is far from obvious and it represents the most significant and common barrier to success that I have encountered since I started working with individual traders. Please, spend some time understanding this fundamentally important issue; I can’t emphasize enough how essential it is.

It also turns out that the human brain is woefully inadequate when it comes to performing sound statistical reasoning on the fly. Daniel Kahneman’s *Thinking, Fast and Slow* (2013) summarises several decades of research into the cognitive biases with which humans are saddled. Kahneman finds that we tend to place far too much confidence in our own skills and judgements, that human reason systematically engages in fallacy and errors in judgement, and that we overwhelmingly tend to attribute too much meaning to chance. A significant implication of Kahneman’s work is that when it comes to drawing conclusions about a complex system with significant amounts of randomness, we are almost guaranteed to make poor decisions without a sound statistical framework. We simply can’t rely on our own interpretation.

As an aside, Kahneman’s *Thinking, Fast and Slow* is not a book about trading, but it probably assisted me with my trading more than any other book I’ve read. I highly recommend it. Further, it is no coincidence that Kahneman’s work essentially created the field of behavioural economics.

3. Risk Management

There are numerous risks that need to be managed as part of an algorithmic trading business. For example, there is infrastructure risk (the risk that your server goes down or suffers a power outage, dropped connection or any other interference) and counter-party risk (the risk that the counter-party of a trade can't make good on a transaction, or the risk that your broker goes bankrupt and takes your trading account with them). While these risks are certainly very real and must be considered, in this section I more concerned with risk management at the trade and portfolio level. This sort of risk management attempts to quantify the risk of loss and determine the optimal allocation approach for a strategy or portfolio of strategies. This is a complex area and there are several approaches and issues of which the practitioner should be aware.

Two (related) allocation strategies that are worth learning about are **Kelly allocation** and **Mean-Variance Optimization (MVO)**. These have been used in practice, but they carry some questionable assumptions and practical implementation issues. It is these assumptions that the newcomer to algorithmic trading should concern themselves with.

Probably the best place to learn about Kelly allocation is in Ralph Vince's *The Handbook of Portfolio Mathematics*, although there are countless blog posts and online articles about Kelly allocation that will be easier to digest. One of the tricky things about implementing Kelly is that it requires regular rebalancing of a portfolio that leads to buying into wins and selling into losses – something that is easier said than done.

MVO, for which Harry Markowitz won a Nobel prize, involves forming a portfolio that lies on the so-called "efficient frontier" and hence minimizes the variance (risk) for a given return, or conversely maximizes the return for a given risk. MVO suffers from the classic problem that new algorithmic traders will continually encounter in their journey: the optimal portfolio is formed with the benefit of hindsight, and there is no guarantee that the past optimal portfolio will continue to be optimal into the future. The underlying returns, correlations and covariance of portfolio components are not stationary and constantly change in often unpredictable ways. MVO therefore does have its detractors, and it is definitely worth understanding the positions of these detractors (see for example Michaud (1989), DeMiguel (2007) and Ang (2014)). A more positive exposition of MVO, governed by the momentum phenomenon and applied to long-only equities portfolios, is given in the interesting paper by Keller *et.al*. (2015).

Another way to estimate the risk associated with a strategy is to use **Value-at-Risk (VaR)**, which provides an analytical estimate of the maximum size of a loss from a trading strategy or a portfolio over a given time horizon and under a given confidence level. For example, a VaR of $100,000 at the 95% confidence level for a time horizon of one week means that there is a 95% chance of losing no more than $100,000 over the following week. Alternatively, this VaR could be interpreted as there being a 5% chance of losing at least $100,000 over the following week.

As with the other risk management tools mentioned here, it is important to understand the assumptions that VaR relies upon. Firstly, VaR does not consider the risk associated with the occurrence of extreme events. However, it is often precisely these events that we wish to understand. It also relies on point estimates of correlations and volatilities of strategy components, which of course constantly change. Finally, it assumes returns are normally distributed, which is usually not the case.

Finally, I want to mention an empirical approach to measuring the risk associated with a trading strategy: **System Parameter Permutation**, or **SPP** (Walton (2014)). This approach attempts to provide an unbiased estimate of strategy performance at any confidence level at any time horizon of interest. By "unbiased" I mean that the estimate is not subject to data mining biases or "p-hacking" mentioned above. I personally think that this approach has great practical value, but it can be computationally expensive to implement and may not be suitable for all trading strategies.

So now you know about a few different tools to help you manage risk. I won't recommend one approach over another, but I will recommend learning about each, particularly their advantages, disadvantages and assumptions. You will then be in a good position to choose an approach that fits your goals and that you understand deeply enough to set realistic expectations around. Bear in mind also that there may be many different constraints under which portfolios and strategies need to be managed, particularly in an institutional setting.

One final word on risk management: when measuring any metric related to a trading system, consider that it is not static – rather, it nearly always evolves dynamically with time. Therefore, a point measurement tells only a tiny fraction of the true story. An example of why this is important can be seen in a portfolio of equities whose risk is managed by measuring the correlations and covariance of the different components. Such a portfolio aims to reduce risk through diversification. However, such a portfolio runs into problems when markets tank: under these conditions, previously uncorrelated assets tend to become much more correlated, nullifying the diversification effect precisely when it is needed most!

Taking it Further

To the three core skills I described above, I would also like to add **numerical optimization**, **machine learning** and **big data analysis** as I think they are incredibly important, however they go a little beyond what I would call "minimum requirements". These skills are nice to have in your toolkit and will make your life as an algorithmic trader easier, but unlike the other skills I described, they are not absolutely critical.

For the adventurous and truly dedicated, I can also recommend learning about **behavioural finance**, **market microstructure** and **macroeconomics**. Again, these are not minimum requirements, but will provide insights that can augment one's ability to navigate the markets.

Finance and economics helps with generating trading ideas, but you don't need formal education in these areas. In fact, I know several folks who are responsible for the hiring and firing that goes on in the professional trading space, and some of these people actually shy away from finance and economics graduates. If you hold such a degree, don't despair though – just recognise that there is more to the practicalities of trading successfully than what you learned in your formal education.

Finally, it would be remiss of me not to mention the soft (that is, non-technical) skills that come in handy. Singularly most important of these is a **critical mindset**. You will read mountains of information about the markets through your algorithmic trading journey, and every page should be read with a critical eye. Get into the habit of testing ideas yourself and gathering your own evidence rather than relying on other people's claims.

Other soft skills that are worth cultivating include perseverance in the face of rejection (you will unfortunately be forced to reject the majority of your trading ideas) and the ability to conduct high-quality, reproducible and objective research.

Your shortest path to proficiency

This section describes what I think is the best approach to acquiring as efficiently as possible the skills I listed above. You will notice that I repeatedly emphasize the **practical application** of the theory that underpins the skills. I very much advocate reading widely and voluminously, but it is critical that you practice implementing the things you read in order to really internalize the skills.

Learning these skills is a process. No one wakes up one day and finds that they are an excellent programmer or an expert in statistics. Like the acquisition of any skill, it takes time and of course effort. My advice is to accept that your skills will gradually improve with time, and that the best way to learn is by *doing*. While it is a good idea to study these topics through formal or structured channels, it is critical that you put them into practice as you go along. Try to tackle problems that are just slightly out of your comfort zone and practice applying what you learn to the markets. Such an approach will see the pace of your learning go exponential.

Further, set realistic expectations around the pace of your learning. There is no point down-playing it: the journey is indeed a long one. If you are a beginner, expect to spend at least a couple of years working hard before you see much success.

The approach I would take, if I were starting out again

When I was starting out with algorithmic trading, I read everything I could get my hands on that related to the markets. I literally read nothing but books and articles that related in some way to the markets for the first three years of my journey. I think this is important – if you want to become proficient at this, you really need to live and breathe it, at least until you gain enough skills to start making some money.

This immersive approach is imperative, but I did make some mistakes. For starters, I didn't start implementing the things I was reading about or doing my own investigations and verifications for some time. I was content to just read. In hindsight, I now realize that this was a well-intentioned, but somewhat lazy approach that didn't really push me too far outside of my comfort zone. If there was something I didn't understand deeply enough to internalize, I could just keep reading. No harm done, right?

I now realize that had I insisted on taking the hard path and putting the things I read about into practice, I would have literally shaved years off the journey to proficiency and ultimately the success I was craving. **By implementing the ideas you read about, you will not only gain the technical skills you need to succeed much more quickly, but you will also develop the mindset of critical thinking and creativity that drives success in this field.** If you can find a mentor or a community to give you feedback on your work or to guide you when you are stuck, your progress will go even faster.

I can't emphasize enough just how crucial this idea of *doing* is. No one ever found success without doing the hard things and being willing to fail. Failure is not a bad thing: it does not mean defeat until you make the decision to stop trying. *Doing* and risking failure is what brings proficiency and eventually mastery.

I also learned that the time you spend practicing needs to be high quality; it is about so much more than just putting in the time. You can ensure your practice is of a high quality by engaging in what others have referred to as ***deliberate practice***. This sort of practice is hard, uncomfortable and tiring. On the other end of the spectrum, reading is passive and not overly draining, but you generally don't get the rewards if you leave it at that. Deliberate practice requires an attitude that demands that you constantly challenge yourself. You need to continually set yourself problems that are just slightly out of your reach or beyond your current level of skill.

When you decide to learn a particular skill, divide it into sub-tasks and then master each one individually and systematically. For example, say you want to learn Python programming. You might divide this into smaller learning tasks that consist of, say, syntax familiarity, data types, variables/expressions/statements, loops, functions, conditionals, input/output, debugging and object oriented programming. Each one of these sub-tasks would be divided into smaller tasks. Such an approach forces you to incrementally improve by ticking off one sub-task after another. It has the nice by-product of being quite motivating to be able to measure your progress in terms of the sub-tasks you've completed.

Deliberate practice forces you to avoid relying on crutches or limiting yourself to researching ideas that are within your current skillset. However, as I mentioned previously, it is uncomfortable, and we have an unfortunate natural tendency to gravitate towards things that we find easy. This can be incredibly limiting and you must think bigger if you want to succeed. If you find yourself shying away from something, chances are it's because it is difficult for you. This is a strong indication that you should tackle it head on! A personal example:

Several years ago, I was learning to use a C-based language for trading research, and I shied away from anything that required significant amounts of string manipulation, like text parsing and web scraping (I now know that C isn't the best tool for these tasks, but that's another story). This limited my trading research to data that I was able to obtain in a relatively clean format. However, after a while I just bit the bullet and forced myself to learn. Now, having become quite accomplished at those skills in more than one programming language, I have access to exponentially more and varied data to use in my research than I had before, which has in turn provided significant inspiration for new strategies.

The lesson is that if something is challenging or uncomfortable, tackle it head-on! While easy to say, this is hard to do because it is very difficult to force ourselves to do things we find uncomfortable, especially when there are more comfortable alternatives (maybe I'll just test another variation of that futures breakout model). So how do we deal with being uncomfortable? It's easy to say "just tackle it head on", but how does one do this in real life, on a consistent, day-to-day basis?

The best and only advice I can give in this regard is to forget about motivation altogether. Yes, you read that correctly. Motivation doesn't work. Think about it: motivation is an incredibly fickle beast, waxing and waning on a daily or even hourly basis. Motivation can be sapped by things as common as being tired or hungry! How then can it be relied upon to deliver something as important as your life goals? My experience tells me that if you rely on your motivation, you are almost guaranteeing your own failure.

So if motivation isn't the answer, what is? The answer is simple: discipline. You must develop the discipline to put in the hours doing the difficult things.

OK, this sounds like another one of those "easy to say, hard to do" things, right? How do you actually put that into practice? Well, it can be difficult to develop a strong internal discipline that you can consistently rely upon, but there are things you can do to set yourself up for success. Some mechanisms and systems that have worked for me over the years include:

- **Seeking accountability**: For example, tell the people who are close to you what you are going to achieve and by when. Making your objectives public and giving them a deadline makes you that much more likely to follow through.

- **Taking a systematic approach**: As mentioned above, break down a goal into sub-tasks and tick them off one by one. Keep a record of your progress and review it at least weekly. Documenting what you're learning and researching also helps with this systematic approach. Keep track of the research you do, but also make notes of things you've learned or things you don't understand.

- **Forming habits**: Try to build habits that are easy to make part of your routine. For example, when I was learning to write code, I would code every morning for one hour before having breakfast. I simply wouldn't eat until I'd spent an hour writing and debugging code. By tying the habit to something that was already part of my day (breakfast), it was so much easier for the habit to stick. You don't need to do exactly what I did, but find something that works with your lifestyle.

- **Being brutally honest with oneself**: For example, don't surf the internet for 45 minutes, then code for 15 minutes and tell yourself you did an hour of work.
- **Surrounding oneself with the right people**: Joining and participating in a community of likeminded people certainly helps keep one focused. Telling these people your goals will also help keep you accountable. Having a community (and ideally a mentor) essentially creates a positive feedback loop that helps you identify exactly where your areas of weakness lie, which can drastically reduce the amount of time it takes to get really good at something. Watch what others who are more proficient or successful do and emulate them. Once you can do what they do, put your own twist on things and make them your own.
- **Reward yourself**: Celebrate little wins and value the progress you make.

If constantly pushing yourself into uncomfortable places sounds daunting or even awful, remember that if you can do this, you will put yourself at a distinct advantage over 99% of people, regardless of how intelligent or otherwise you may be. The consistent application of a disciplined approach trumps intelligence and natural talent every single day of the week. I don't know about you, but I find that incredibly comforting and empowering! I can tell you that the essential reason that I was able to leave my day job for a career in finance, the one thing that it all boils down to, is that I was and am willing to do the things that most people aren't. For example, I would get up early and put in a couple of hours before going to work. I would forego events that I would really have liked to attend in order to work evenings and nights. I've lost count of the number of weekends I've given up over the last decade. If you want to achieve something extraordinary, you simply can't do what everyone else does. That is reality.

Important Practical Matters

Finally, I want to cover some of the practical considerations that I think are important to be aware of when starting out.

Expectations

When you are learning algorithmic trading, you will find that it can be an emotional experience. This will pass as your experience and proficiency develops, but during the early years, your emotional state may become somewhat tied to your success or otherwise in the markets. This can obstruct progress, so it is worth understanding and addressing this issue.

In life, our happiness is often tied up with our expectations. Therefore, it makes sense to set ambitious yet realistic expectations for your algorithmic trading journey right at the outset. Your mental state will thank you for it.

First of all, this is a cliché, but one that rings true: *trading is the hardest way to make easy money*. Don't expect an easy ride or fast riches. Rather, expect at least a couple of years of unrewarded effort and slow riches, if any riches at all.

Related to this, don't expect to make multiples of your money in short periods of time. However, once proficient, you should expect to outperform the market over the long term (potentially significantly). Otherwise, what's the point of doing this at all? If you look through the Barron's list of top 100 hedge funds from last year, you'll see that the best performing funds have a 3-year compounded annual return of just under 30%. Do you think it is reasonable that you could out-perform these top-performing funds, with their quant teams and enormous financial resources? That was kind of a loaded question, because perhaps surprisingly, the answer is yes! Funds with billions under management face completely different constraints than a do-it-yourself trader, the most interesting of these being related to capacity. For example, an individual trading say a half-million-dollar futures account can take a completely different approach to a fund that aims to generate returns on billions. There may exist market phenomena that can generate returns that are significant compared to the position sizing of a retail account, but which are not capable of carrying the trades of a larger fund. Therefore, while on the surface, it may appear that a retailer is at a significant disadvantage, there are also opportunities.

One last comment about expectations: avoid becoming fixated on how much you can make. The amount of reward you can gain is inextricably tangled up with the amount of risk you are willing to take. **Thinking about reward in terms of risk** rather than in isolation will lead you to much more sensible expectations.

We need to talk about frequency

The frequency at which a strategy trades is another significant consideration. Lower frequency systems might hold trades for days to months. Intra-day systems might hold trades for minutes to hours. High frequency trading generally refers to systems with holding periods on the order of milliseconds to seconds.

Trade frequency (or holding period) is an important consideration due to the effect of **transaction costs**. As the average holding period decreases, the average price change between the trade entry and exit also decreases. That is, the average profit potential of individual trades decreases. However, transaction costs don't decrease, in fact they remain constant. A trade that was held for ten seconds has the same cost as a trade that was held for ten hours (assuming no swap), but their profit potentials are likely to be quite different.

The costs of trading are dependent on your broker, and to an extent your infrastructure, and will vary depending on individual set ups. Therefore, I can't prescribe a cut-off holding period, but I will recommend that you understand the impact of trading costs across various time horizons in order to make sensible decisions. I have certainly noticed an obsession with "trading off the five-minute chart" amongst a number of retail traders that I've worked with, and while I do understand the lure and the excitement of such a trading style, one should really **understand its viability before spending a lot of time on it**.

Trading at high frequencies typically requires low-latency execution systems, server co-location and detailed knowledge of order book dynamics. This is not impossible for independent traders, but there is other lower hanging fruit that is much more accessible.

My advice for beginners with regards to trade frequency: start slow! Investigate systems with holding periods on the order of days, weeks or even months. The odds are not quite as severely stacked against you. You are much more likely to find a decent strategy if you take this approach, even as a relative beginner. By the time you've got one or two systems in your portfolio, you will have learned a lot and be in a much stronger position to consider higher frequency trading. Again, this is what I would do if I were starting over.

What to read and when

Hopefully you got some of your maths, statistics, and/or programming knowledge via your formal education. In my experience with graduates from various fields, Computer Science, Physics, Mathematics, Engineering, and Econometrics degrees are quite useful in terms of the background knowledge they afford. Finance and economics degrees are (maybe surprisingly) less so. Regardless of your formal education, there will almost certainly be gaps in your knowledge, so even if you don't have formal education in one of the fields I mentioned, don't despair, simply start learning now.

What to read and in what order really depends on what your background is. Therefore, I can't prescribe a recommendation that would suit everyone. Instead, I'll suggest you start with the three pillars of algorithmic trading I described above (statistics, programming and risk management) and refer you to Robot Wealth's recommended reading list. What you should read also depends on the direction in which you want to go. For example, if you are interested in options trading, you'll want to study volatility and pricing models, which I've barely mentioned in the blog so far. Likewise, if you are interested in high frequency trading, you'll need a solid grounding in market microstructure and order book dynamics. Other directions include derivatives pricing and portfolio management and you could spend a lifetime learning about any one of these topics. Having said that, relationships exist between these topics and there is benefit in holistic understanding, so try not to limit yourself to one particular field.

Personally, when I was starting out I read voraciously anything to do with trading, and I know that even with such an appetite one can't possibly get through all the texts in the reading list in great detail. But happily, you don't need to. Far more important than details is understanding the practical application and where to find the detailed information when you need it. For example, take Tsay's *Analysis of Financial Time Series* – this weighty tome is a classic econometric text that you could easily pore over for months, if not years. However, it is far more useful from a practical perspective to understand it to the point that you could, for example, describe the inputs to, and the uses and limitations of an ARMA or GARCH model. Knowing where to find more detailed information around the implementation and diagnostics when you actually need them is enough.

Finally, understand that when you read the work of others, whether that be in an academic paper, a blog post or a book, that the author is likely not giving away the keys to the successful implementation of any trading idea that may be described (Robot Wealth included…sorry folks). Often the way a strategy is tuned, optimized and applied is the key to making it work. Therefore, if you use the work of others, you absolutely must apply your own unique twist to it. Read everything with an evidence-based and critical mindset, and don't believe anything until you prove it to yourself.

Infrastructure

We've discussed the technical and non-technical skills that are required for algorithmic trading, but have barely mentioned another key ingredient: infrastructure. Without good infrastructure and execution – that is, a system that can read market and other data, perform computations and send instructions to a broker or exchange – you can't trade. It therefore pays to understand how a trading interface receives and sends information over a network. For this, you'll need to know how to use Application Programming Interfaces (APIs), which is just a fancy way of describing how to communicate with a particular software application. Most brokers offer trading via an API, but the quality of the API documentation varies greatly. Poor quality API documentation can make developing trading infrastructure quite painful, so it pays to look into this when shopping for brokers.

While most brokers offer their own API for electronic trading, there is an industry standard protocol, namely FIX, or Financial Information Exchange. While broker APIs vary, the FIX protocol is an industry standard and can be used across a range of brokers and financial institutions. At its core, FIX is a simple messaging protocol, but in practice it can be tricky to get started with, particularly if you're not a software developer (at least that's what I found). A good place to start is QuickFIX, which is an open source implementation of the FIX protocol. It is a C++ library, but comes with bindings for a number of languages. Expect some pain getting to know QuickFIX (the documentation isn't amazing), but I strongly believe it is worth your time. In the future, I'll share some trading applications built on FIX in both Python and .Net frameworks.

The actual hardware required for an algorithmic trading business is less of an issue than it was in years gone by thanks to the rise of cloud computing and commercial hosting services. For example, it makes sense for many traders to outsource their hardware requirements to an external provider rather than maintaining dedicated trading machines with backup power and network connectivity in their own homes. The exceptions are low-latency systems, over which the trader will probably want to retain ultimate control.

Another piece of the puzzle is data: where to get it, how much to pay for it, and how to clean, process and manage it. These are all important questions. Remember that when you build trading models, they are only as good as the data that was used. Obtaining end-of-day data for some markets is relatively simple these days via sources like Yahoo! and Google. It is hard to find good, free, intra-day data and usually you will wind up paying for it or collecting it yourself or both. It is also worth remembering that the data relevant to financial markets extends far beyond price and trade history data. Data aggregation services like Quandl provide a large and growing repository of information that you may be able to use in a trading strategy, and you can also gather your own alternative data sets like social media sentiment, machine readable earnings announcements, economic data releases and the like. Creativity in this space is a good thing!

CHAPTER 3:
The Importance of Backtesting

Nearly all research related to algorithmic trading is **empirical** in nature. That is, it is based on **observations and experience**. Contrast this with **theoretical** research which is based on **assumptions, logic and a mathematical framework**. Often, we start with a theoretical approach (for example, a time-series model that we *assume* describes the process generating the market data we are interested in) and then use empirical techniques to test the validity of our assumptions and framework. But we would never commit money to a mathematical model that we *assumed* described the market without testing it using real observations, and every model is based on assumptions (to my knowledge no one has ever come up with a comprehensive model of the markets based on first principles logic and reasoning). So, empirical research will nearly always play a role in the type of work we do in developing trading systems.

So why is that important?

Empirical research is based on observations that we obtain through experimentation. Sometimes we need thousands of observations in order to carry out an experiment on market data, and since market data arrives in real time, we might have to wait a very long time to run such an experiment. If we mess up our experimental setup or think of a new idea, we would have to start the process all over again. Clearly this is a very inefficient way to conduct research.

A much more efficient way is to simulate our experiment on historical market data using computers. In the context of algorithmic trading research, such a simulation of reality is called a ***backtest***. Backtesting allows us to test numerous variations of our ideas or models quickly and efficiently and provides immediate feedback on how they might have performed in the past. This sounds great, but in reality, backtesting is fraught with difficulties and complications, so I decided to write this chapter that I hope illustrates some of these issues and provides some guidance on how to deal with them.

Why Backtest?

Before I get too deeply into backtesting theory and its practical application, let's back up and talk about why we might want to backtest at all. I've already said that backtesting allows us to carry out empirical research quickly and efficiently. But why do we even need to do that? Everyone knows that we should just buy when the Relative Strength Index drops below 30, right?

OK, so that was obviously a rhetorical question. But I just wanted to highlight one of the subtly dangerous modes of thinking that can creep in if we are not careful. Now, I know that for the vast majority of Robot Wealth readers, I am preaching to the converted here, but over the last couple of years I've worked with a lot of individuals who have come to trading from non-mathematical or non-scientific backgrounds who struggle with this very issue, sometimes unconsciously. This is a good place to address it, so here goes.

In the world of determinism (that is, well-defined cause and effect), natural phenomena can be represented by tractable, mathematical equations. Engineers and scientists reading this will be well-versed for example in Newton's laws of motion. These laws quantify a physical consequence given a set of initial conditions and are solvable by anyone with a working knowledge of high school level mathematics. The markets however are not deterministic (at least not in the sense that the information we can readily digest describes the future state of the market). That seems obvious, right? The RSI dropping below 30 does not portend an immediate price rise. And if price were to rise, it didn't happen because the RSI dropped below 30. Sometimes prices will rise following this event, sometimes they will fall and sometimes they will do nothing. We can never tell for sure, and often we can't describe the underlying cause, beyond more people buying than selling. Most people can accept that fact. However, I have observed time and again a paradox: namely, that the same person who accepts that markets are not deterministic will believe in a set of trading rules because they read them in a book or on the internet.

I have numerous theories about why this is the case, but one that stands out is that it is simply easy to believe things that are nice to believe. Human nature is like that. This particular line of thinking is extraordinarily attractive because it implies that if you do something (simple) over and over again, you will make a lot of money. But that's a dangerous trap to fall into. And you can even fall into it if your rational self knows that the markets are not deterministic, but you don't question the assumptions underlying that trading system you read about.

I'm certainly not saying that *all* DIY traders fall into this trap, but I have noticed it on more than a few occasions. If you're new to this game, or you're struggling to be consistently profitable, maybe this is a good thing to think about.

I hope it is clear now why backtesting is important. Some trading rules will make you money; most won't. But the ones that do make money don't work because they accurately describe some natural system of physical laws. They work because they capture a market characteristic that over time produces more profit than loss. You never know for sure if a particular trade is going to work out, but sometimes you can conclude that in the long run, your chances of coming out in front are pretty good. Backtesting on past data is the one tool that can help provide a framework in which to conduct experiments and gather information that supports or detracts from that conclusion.

Simulation versus Reality

You might have noticed that in the descriptions of backtesting above I used the words **simulation of reality** and **how our model *might* have performed in the past**. These are very important points! No simulation of reality is ever exactly the same as reality itself. Statistician George Box famously said "All models are wrong, but some are useful" (Box, 1976). It is critical that our simulations be accurate enough to be useful. Or more correctly, we need our simulations to be fit for purpose, after all, a simulation of a monthly ETF rotation strategy may not need all the bells and whistles of a simulation of high frequency statistical arbitrage trading. The point is that any simulation must be accurate enough that it supports the decision-making process for a particular application, and by "decision making process" I mean the decisions around allocating to a particular trading strategy.

So how do we go about building a backtesting environment that we can use as a decision-support tool? Unfortunately, backtesting is not a trivial matter and there are a number of pitfalls and subtle biases that can creep in and send things haywire. But that's OK, in my experience the people who are attracted to algorithmic trading are usually up for a challenge!

At its most simple level, backtesting requires that your trading algorithm's performance be simulated using historical market data, and the profit and loss of the resulting trades aggregated. This sounds simple enough, but in practice it is incredibly easy to get inaccurate results from the simulation, or to contaminate it with bias such that it provides an extremely poor basis for making decisions. Dealing with these two problems requires that we consider:

1. The accuracy of our simulation; and
2. Our experimental methodology and framework for drawing conclusions from its results.

Both these aspects need to be considered in order to have any level of confidence in the results of a backtest. I can't emphasize enough just how important it is to ensure these concepts are taken care of adequately; compromising them can invalidate the results of the experiment. Most algorithmic traders spend vast amounts of time and effort researching and testing ideas and it is a tragic waste of time if not done properly. The next sections explore these concepts in more detail.

Simulation Accuracy

If a simulation is not an accurate reflection of reality, what value is it? Backtests need to be as true a reflection of real trading as necessary to make them fit for their intended purpose. In theory, they should generate the very same trades with the same results as live trading the same system during the same time period.

In order to understand the accuracy of a backtest, you need to understand how the results are simulated and what the limitations of the simulation are. The reality is that no model can precisely capture the phenomena being simulated, but it is possible to build a model that is useful for its intended purposes. It is imperative that we create models that are as accurate a reflection of reality as possible, but equally that we are aware of the limitations. Even the most realistic backtesting environments have limitations.

Backtesting accuracy can be affected by:

- The parameters that describe the **trading conditions** (spread, slippage, commission, swap) for individual brokers or execution infrastructure. Most brokers or trading setups will result in different conditions, and conditions are rarely static. For example, the spread of a market (the difference between the prices at which the asset can be bought and sold) changes as buyers and sellers submit new orders and amend old ones. Slippage (the difference between the target and actual prices of trade execution) is impacted by numerous phenomena including market volatility, market liquidity, the order type and the latency inherent in the trade execution path. The method of accounting for these time-varying trading conditions can have a big impact on the accuracy of a simulation. The most appropriate method will depend on the strategy and its intended use. For example, high-frequency strategies that are intended to be pushed to their limits of capacity would benefit from modelling the order book for liquidity. That approach might be overkill for a monthly ETF rotation strategy being used to manage an individual's retirement savings.

- The **granularity** (sampling frequency) of the data used in the simulation, and its implications. Consider a simulation that relies on hourly open-high-low-close (OHLC) data. This would result in trade entry and exit parameters being evaluated on every hour using only four data points from within that hour. What happens if a take profit and a stop loss were evaluated as being hit during the same OHLC bar? It isn't possible to know which one was hit first without looking at the data at a more granular level. Whether this is a problem will depend on the strategy itself and its entry and exit parameters.

- The **accuracy of the data** used in the simulation. No doubt you've head the modelling adage "Garbage in, garbage out." If a simulation runs on poor data, obviously the accuracy of the results will deteriorate. Some of the vagaries of data include the presence of outliers or bad ticks, missing records, misaligned time stamps or wrong time zones, and duplicates. Financial data can have its own unique set of issues too. For example, stock data may need to be adjusted for splits and dividends. Some data sets are contaminated with ***survivorship bias***, containing only stocks that avoided bankruptcy and thus building in an upward bias in the aggregate price evolution. Over-the-counter products, like forex and CFDs, can trade at different prices at different times depending on the broker. Therefore a data set obtained from one source may not be representative of the trade history of another source. Again, the extent to which these issues are problems depends on the individual algorithm and its intended use.

The point of the above is that the accuracy of a simulation is affected by many different factors. These should be understood in the context of the strategy being simulated and its intended use so that sensible decisions can be made around the design of the simulation itself. Just like any scientific endeavour, the design of the experiment is critical to the success of the research!

As a practical matter, it is usually a good idea to check the accuracy of your simulations before deploying them to their final production state. This can be done quite easily by running the strategy live on a small account for some period of time, and then simulating the strategy on the actual market data on which it was run. Regardless of how accurate you thought your simulator was, you will likely see (hopefully) small deviations between live trading and simulated trading, even when the same data is being used. Ideally the deviations will be within a sensible tolerance range for your strategy, that is, a range that does not significantly alter your decisions around allocating to the strategy. If the deviations do cause you to rethink a decision, then the simulator was likely not accurate enough for its intended purpose.

Development Methodology

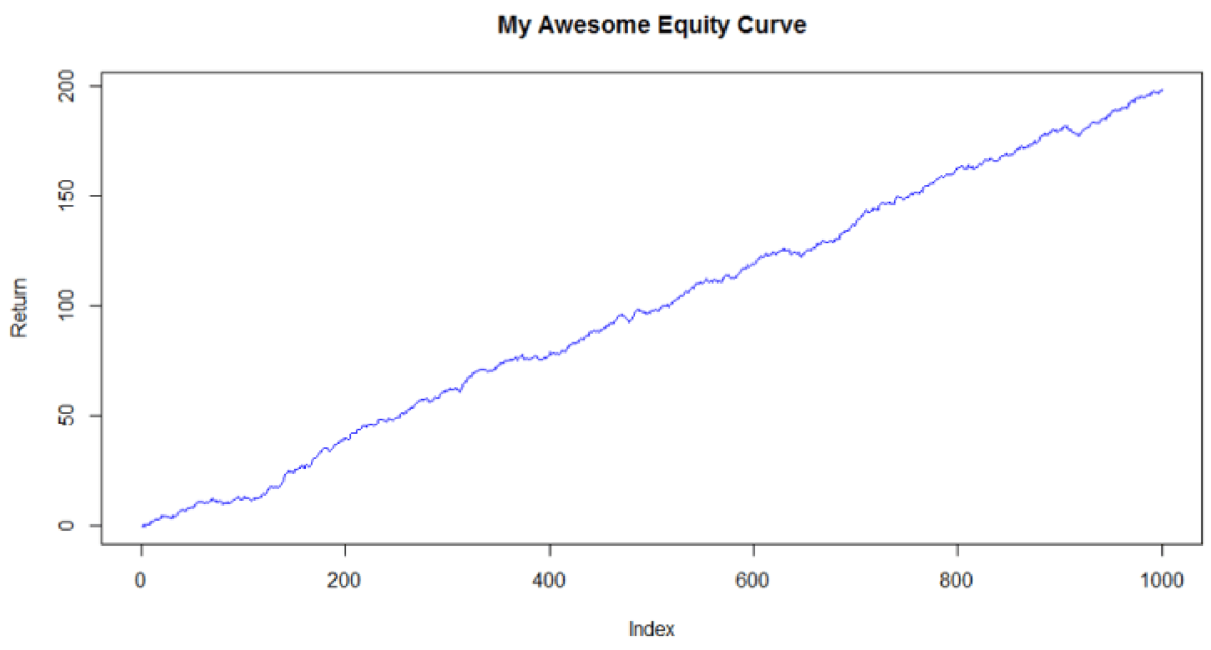
In addition to simulation accuracy, the experimental methodology itself can compromise the results of our simulations. Many of these biases are subtle yet profound: they can and will very easily creep into a trading strategy research and development process and can have disastrous effects on live performance. Accounting for these biases is critical and needs to be considered at every stage of the development process.

Robot Wealth's course *Fundamentals of Algorithmic Trading* details a workflow that you can adopt to minimize these effects as well as an effective method of maintaining records which will go a long way to helping identify and minimize bias in its various forms. For now, I will walk through and explain the various biases that can creep in and their effect on a trading strategy. For a detailed discussion of these biases and a highly interesting account of the psychological factors that make accounting for these biases difficult, refer to David Aronson's *Evidence Based Technical Analysis* (2006).

Look-Ahead Bias, Also Known As Peeking Bias

This form of bias is introduced by allowing future knowledge to affect trade decisions. That is, trade decisions are affected by knowledge that would not have been available at the time the trade decision was taken. A good simulator will be engineered to prevent this from happening, it is surprisingly easy to allow this bias to creep in when designing our own backtesting tools. A common example is executing an intra-day trade on the basis of the day's closing price, when that closing price is not actually known until the end of the day.

When using custom-built simulators, you will find that you will typically need to give this bias some attention. I commonly use the statistical package R and the Python programming language for various modelling and testing purposes and when I do, I find that I need to consider this bias in more detail. On the positive side, it is easy to identify when a simulation doesn't properly account for it, because the resulting equity curve will typically look like the one shown below, since it is easy to predict an outcome if we know about the future!

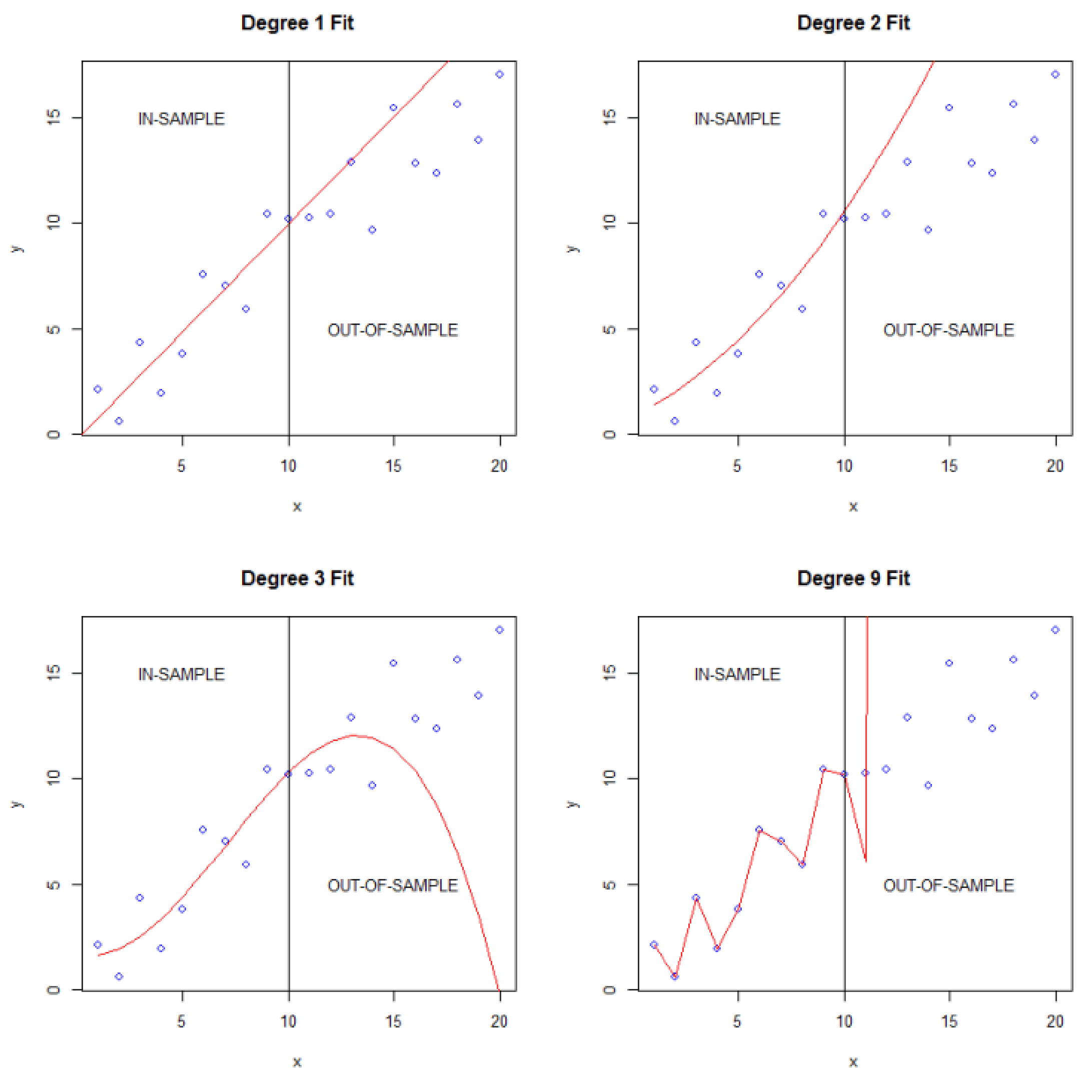


Another way this bias can creep in more subtly is when we use an entire simulation to calculate a parameter and then retrospectively apply it to the beginning of the next run of the simulation. Portfolio optimization parameters are particularly prone to this bias.

Curve-Fitting Bias, Also Known As Over-Optimization Bias

This is the bias that allows us to create magical backtests that produce annual returns on the order of hundreds of percent. Such backtests are of course completely useless for any practical trading purpose.

The following plots show curve-fitting in action. The blue dots are an artificially generated linear function with some noise added to distort the underlying signal. It has the form $y=mx+b+\epsilon$where ϵ is noise drawn from a normal distribution, and we have 20 points in our data set. Regression models of varying complexity were fit to the first 10 points of the data set (the in-sample set) and these models were tested against the unseen data consisting of the remaining 10 points not used for model building (the out-of-sample set). You can see that as we increase the complexity of the model, we get a better fit on the in-sample data, but the out-of-sample performance deteriorates drastically. Further, the more complex the model, the worse the deterioration out of sample.



The more complex models are actually **fitting to the noise in the data rather than the signal**, and since noise is by definition random, the models that predict based on what they know about the in-sample noise perform horrendously out-of-sample. When we fit a trading strategy to an inherently noisy data set (and financial data is extremely noisy), we run the risk of fitting our strategy to the noise, and not to the underlying signal. The underlying signal is the anomaly or price effect that we believe provides profitable trading opportunities, and this signal is what we are actually trying to capture with our model. If we fit our model to the noise, we will essentially end up with a random trading model, which of course is of little use to anyone, except perhaps your broker.

Data-Mining Bias, Also Known As Selection Bias

Data mining bias is another significant source of over-estimated model performance. It takes a number of forms and is largely unavoidable – therefore rather than eliminating it completely, we need to be aware of it and take measures to account for it. Most commonly, you will introduce it when you select the best performer from a number of possible algorithms, algorithm variants, variables or markets to take forward with your development process. If you try enough strategies and markets, you will eventually (relatively often actually) find one that performs well due to luck alone.

For example, say you develop a trend following strategy for the forex markets. The strategy does exceptionally well in its backtest on the EUR/USD market, but fails on the USD/JPY market. By selecting to trade the EUR/USD market, you have introduced selection bias into your process, and the estimate of the strategy's performance on the EUR/USD market is now upwardly biased. You must either temper your expectations from the strategy, or test it on a new, unseen sample of EUR/USD data and use that result as your unbiased estimate of performance.

It is my experience that beginners to algorithmic trading typically suffer more from the effects of curve-fitting bias during the early stage of their journey. That may be because these effects tend to be more obvious and intuitive. Selection bias on the other hand can be just as severe as curve-fitting bias, but is not as intuitively easy to understand. A common mistake is to test multiple variants of a strategy on an out-of-sample data set and then select the best one based on the out-of-sample performance. While this is not necessarily a mistake *per se*, the mistake is treating the performance of the selected variant in the out-of-sample period as an unbiased estimate of future performance. This may not be the end of the world if only a small number of variants were compared, but what if we looked at hundreds or even thousands of different variants, as we might do if were using machine learning methods? Surely among hundreds of out-of-sample comparisons, at least one would show a good result by chance alone? In that case, how can we have confidence in our selected strategy?

This begs the question of how to account for this data-mining bias effect. In practical terms, you will quickly run into difficulty if you try to test your strategy on new data after each selection or decision point since the amount of data at your disposal is finite. Other methods to account for data mining bias include comparing the performance of the strategy with a distribution of random performances, White's Reality Check and its variations, and Monte Carlo Permutation Tests.

Conclusion

This final *Back to Basics* chapter described why we take an experimental approach to algorithmic trading research and detailed the main barriers to obtaining useful, accurate and meaningful results from our experiments. A robust strategy is one that exploits real market anomalies, inefficiencies, or characteristics, however it is surprisingly easy to find strategies that appear robust in a simulator, but whose performance turns out to be due to inaccurate modelling, randomness or one of the biases described above. Clearly, we need a systematic approach to dealing with each of these barriers and ideally a workflow with embedded tools, checks and balances that account for these issues. I would go as far as to say that in order to do any serious algorithmic trading research, or at least to do it in an efficient and meaningful way, we need a workflow and a research environment that addresses each of the issues detailed in this chapter. That's exactly what we provide in *Fundamentals of Algorithmic Trading*, which was written in order to teach new or aspiring algorithmic traders how to operate the tools of the trade, and even more importantly, how to operate them *effectively* via such a workflow. If you would like to learn such a systematic approach to robust strategy development, head over to the www.RobotWealth.com to find out more.

References

Aronson, D. 2006, Evidence Based Technical Analysis, Wiley, New York

Box, G. E. P. 1976, *Science and Statistics*, "Journal of the American Statistical Association" 71: 791-799

Back to Basics: Recommended Reading

Below is a curated list of books that I recommend you start with on your journey towards learning algorithmic trading. For a more guided approach to learning the craft, check out Robot Wealth's Fundamentals of Algorithmic Trading course.

Aronson, D. (2006). *Evidence-Based Technical Analysis – Applying the Scientific Method and Statistical Inference to Trading Signals,* John Wiley & Sons.

Something of a call to arms for the technical trading community to adopt a scientific approach to the markets. More than that, it provides some practical tips on how to do it following the work of White (2000). I particularly enjoyed learning about the various cognitive biases to which the human brain is subject and how to account for them.

Jaeckle, E and Tomasini, E. (2009). *Trading Systems – A New Approach to System Development and Portfolio Optimization*, Harriman House Ltd.

A useful starting point for traders from a technical analysis background who are interested in making their systems development process more robust. Introduces optimization, curve fitting, monte carlo analysis and trend bias.

Vince, R. (2007). *The Handbook of Portfolio Mathematics – Formulas for Optimal Allocation and Leverage*, John Wiley & Sons.

This is compelling reading that is a must for systems traders. Vince provides a mathematical treatment of money management including ideas borrowed from gambling, key probability concepts, reinvestment, optimal growth, Kelly betting, modern portfolio theory and the 'leverage space' portfolio.

Pardo, R. (2008). *The Evaluation and Optimization of Trading Strategies*, John Wiley & Sons.

Pardo provides a useful introduction to the use of simulation to evaluate and optimize trading systems. This is another beginner level text that includes a useful treatment of simulation accuracy, statistical measures of performance, optimization, curve fitting and walk-forward analysis.

Chan, E. (2009). *Quantitative Trading – How to Build Your Own Algorithmic Trading Business*, John Wiley & Sons.

Aimed at retail traders, this is a highly practical guide to succeeding at the business of quantitative trading. Includes oft-neglected topics such as sourcing good data, choosing a development environment, advice on working as a proprietary trader, and the importance of good execution. Also introduces a number of approaches to strategy development and includes a coverage of risk management.