CHAPTER 1

The Basics of Algorithmic Trading

An algorithmic trading strategy feeds market data (historical or live) into a computer (backtest or automated execution) program. The program then submits orders to the broker through an API, and receives order status notifications back from the broker. The flowchart in Figure 1.1 illustrates this process.

Notice that I deliberately use the same box to indicate the computer program that generates backtest results and live orders: This is the best way to ensure we are trading the exact same model that we have backtested.

In this chapter, I will discuss the latest services, products, and their vendors applicable to each of the blocks in Figure 1.1. In addition, I will describe my favorite performance metrics, the way to determine the optimal leverage, and the simplest asset allocation method. Though I have touched on many (but not all) of these issues in my previous books, I have updated them here based on the state of the art. The FinTech industry has not been standing still, nor has my understanding of issues ranging from brokers' safety to subtleties of portfolio optimization.

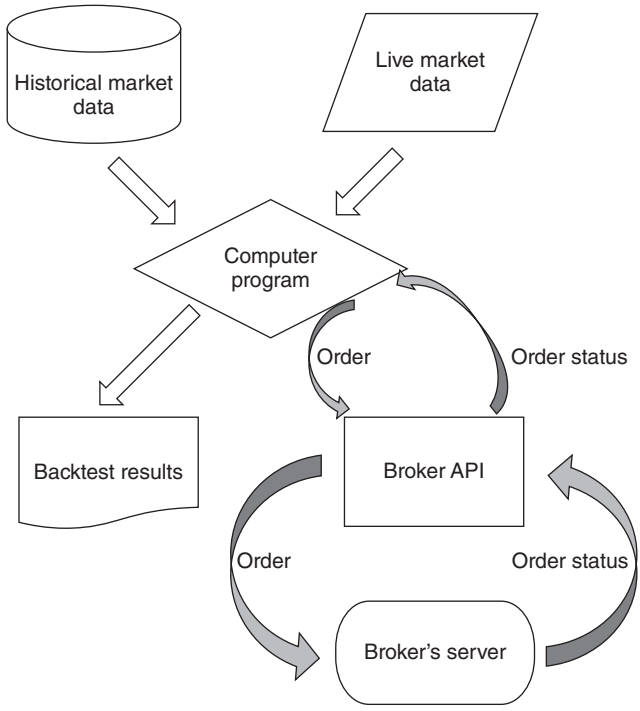

FIGURE 1.1 Algorithmic trading at a glance

Historical Market Data

For **daily historical data in stocks and futures**, I have been using CSI (csidata.com) for a long time. CSI has a very flexible, and robust, desktop application. The beauty of this application is that we can set a time in the evening when the data are automatically updated through an Internet connection with CSI's server. Also, the data can be stored in various convenient formats such as .txt, .csv, or .xlsx. We can ask it to automatically adjust historical stock (and ETF) prices for splits and dividends. For a little extra, CSI can also provide delisted stocks' historical data, so that you can have a survivorship-bias-free data set.[1] (By the way, CSI data powers Yahoo! Finance's historical stock data.) For futures, we can choose different rollover methods to create continuous contracts. Or not, since the original contract prices are also available, and many professional traders prefer to backtest futures using original contract prices instead of back-adjusted continuous contract prices. This is because the latter depends on a particular roll method, and may have look-ahead bias embedded (see Chan, 2013,

for a detailed exploration of this issue). Finally, CSI has excellent customer support through email and phone.

An alternative to CSI is Quandl.com, which is a consolidator of many kinds of data from many different vendors. It also provides an API in different languages (including MATLAB, which I use in this book, or Python, which many other traders use) that we can use for data selection and download. Some of Quandl's data are free (daily data for stocks is one example), and others require payment. I have purchased, for example, fundamental stock data from them (see Chapter 2, Factor Models), and they are much more economical than established vendors such as Compustat.

Serious traders or academic finance researchers may prefer stock, ETF, and mutual fund data from CRSP (www.crsp.com). Their historical data are carefully compiled to be survivorship-bias-free, and dividends and splits are provided separately so you can decide how to utilize them in a backtest. But most importantly, they provide the best bid and offer (BBO) prices at the close. These are important because, as is explained in Box 6.2 in Chapter 6, using the usual consolidated closing prices from CSI or Quandl can inflate backtest performances for certain strategies. A similar issue arises from using the consolidated opening prices. The best open and close prices to use are the auction prices from the primary exchange. (See also Box 6.2 for an explanation of how we can extract such auction prices from tick data.) The second best open and close prices to use are the BBO prices that can be used to compute the midprices at the open and close. Unfortunately CRSP does not provide the BBO prices at the open, so one must use intraday data for that purpose. For academic researchers, CRSP data can be obtained at a lower cost through WRDS (wrds-web.wharton.upenn.edu), which is a consolidator of many high quality historical databases for scholarly research.

Of course, those serious traders who can afford to buy data from CRSP may also be able to afford a Bloomberg terminal subscription. One advantage of a Bloomberg terminal is that we can download the "primary exchange" close price for US stocks. Of course, a Bloomberg subscription also includes access to many historical and live data spanning a wide variety of instruments and, importantly, breaking news on every stock. I have found Bloomberg's news service to be superior to many other vendors'. Often, we will see a stock moves suddenly, and are not able to find the reason anywhere else but on Bloomberg's news feed. They do capture the most obscure news on the most obscure stocks in the shortest time frame, which is important when you have an event-driven strategy. Bloomberg's historical US stock data are also survivorship-bias-free. (To be fair to Thomson Reuter's Eikon platform,

which is a keen competitor to Bloomberg's, I have not tested its news feed. So it is possible that it provides just as wide and timely coverage as well. There is one feature on Eikon that impressed me in a demo: I was able to see the geographical locations of individual oil tankers and where they were headed. Apparently, this is useful to oil traders to predict short-term oil inventory, supply, and demand.)

For **futures traders**, daily data does not present much of a problem. CSIs and the free data on Quandl are as good as any.

Daily options data can be obtained from ORATS.com as well as ivolatility.com. Both offerings are survivorship-bias-free. The institutional trader or academic researcher may also purchase from Option Metrics, which is often part of the WRDS package (see above). One nice feature of all these databases: They do not include just option closing prices, but also the bid-ask quote at the close as well. This is important because some options, especially ones that are out-of-the-money or have long tenor, may be traded infrequently. So the last trade price of the day may be very different from the bid-ask quotes at the close, and is not indicative of the price we can actually trade at or near the close. These databases also include auxiliary but important information such as the Greeks and implied volatilities. Furthermore, they may include an implied volatility surface. This uses implied volatilities computed on actual options and interpolates them to yield implied volatilities for strikes and tenors that did not actually exist.

Options historical data tend to be more expensive than stock or futures data, due to their voluminous nature. However, it can be cheaper to rent *intraday* option prices from QuantGo.com than to buy *daily* option prices from other vendors. We will talk more about QuantGo when we discuss intraday data in general. It would be a trivial programming exercise to extract the end-of-day quotes from the intraday data, by looking for the quotes with timestamps at or just before the daily closing time.

Beyond daily price data, there are, of course, **fundamental financial data** for companies. I already mentioned that Quandl carries such data. Institutional traders would most likely look to Compustat for such data. For corporate earnings estimates by analysts, Thomson Reuters' IBES database is the standard. Both Compustat and IBES are available from WRDS. Meanwhile, crowd-sourced earnings estimates are available from Estimize. There is some research that suggests Estimize's contributors can more accurately forecast actual earnings than traditional sell-side analysts (Wang et al., 2014). An example strategy using Estimize's data is discussed in Deltix (2014). Short interest data are available from Compustat and

SunGard's Astec database. SunGard's data have a lot more details culled from stock lenders and prime brokers around the Street than a simple short interest number. In addition, their data are available on an intraday basis as a live feed, though the historical data do not have historical time stamps.

News data is another type of data that is becoming fashionable. Many vendors sell elementized news feeds (i.e., news that is machine-readable, which makes it easier to capture keywords, categories, and the like), including Bloomberg, Dow Jones, and Thomson Reuters. If it is too much trouble for your strategy to generate trading signals from raw news data, you can also buy news sentiment data from Ravenpack, Thomson Reuters News Analytics, MarketPsych, or Accern. (AcquireMedia's NewsEdge database is similar, but they provide only impact scores. This is a kind of unsigned sentiment score that doesn't tell you which way the stock will move, only that it will move, which may be suitable for options traders.) However, there is one problem for sentiment data: Different vendors have different ways to compute sentiment scores from the raw news. So a trading model depends to some extent on which vendors' sentiment scores are most predictive.

We will leave the topic of buying or renting intraday data to Chapter 6 on Intraday Trading, because the features associated with intraday data are intimately tied to the market microstructure. Here, we will just note that some of the historical intraday data vendors include tickdata.com, nanex.net, CQG, QuantGo.com, kibot, and of course, the various exchanges themselves.

Finding, buying, or renting data is both expensive and time-consuming, though consolidators like Quandl and QuantGo have made it much less so. Another way to avoid dealing with acquiring data directly is to adopt a trading platform that comes integrated with data (though you may have to pay for the data separately). A good example is Quantopian.com, which provides free US stock trades data with one-minute bars, together with many other forms of fundamental and news data at lower frequency. (I have been told that futures data will be available soon.) We will talk more about platforms like these in the section "Backtesting and Trading Platforms."

Live Market Data

Most if not all brokerages provide live market data to their clients, and if you are trading a daily strategy (i.e., you trade only at the market open or close), such data are usually more than sufficient. However, if you engage

in intraday trading, then the quality of data becomes a bigger issue. As we will discuss more thoroughly in Chapter 6, low latency market data can be quite expensive to obtain. Vendors that provide data suitable for intraday trading that can tolerate a latency of more than 25 ms (ms = millisecond) include eSignal, IQFeed, CQG, Interactive Data, Bloomberg, and many others. But vendors that provide data feed with latency of below 10 ms are far fewer: They include S&P Capital IQ (formerly QuantHouse), SR Labs, and Thomson Reuters. Of course, you can also subscribe to the direct feeds from the exchange, but that is strictly for high frequency traders who find the high expense justified by the high return. (That is true unless you are after currency data, where most FX exchanges will give their customer a free direct feed.)

As with historical market data, many trading platforms also include live market data feeds. These will be discussed in the following section.

Backtesting and Trading Platforms

Traditionally, we would backtest our trading strategy on one platform (e.g., R) and once successful, write a different program to automate execution of the strategy that utilizes a broker's API. However, this proves to be quite bug-prone: there is no way to ensure that the backtest and the live trading program encapsulate exactly the same trading logic. Fortunately, most backtest platforms nowadays have extended their ability to execute live as well; hence, we will combine the discussions on backtesting and trading platforms here.

As I mentioned in the Preface, MATLAB has been my favorite backtesting platform. It has a very comprehensive and user-friendly interface for developing and debugging programs, and it has a wide array of toolboxes that cover almost every arcane mathematical or computational technique you will likely encounter in trading strategy development. One of these toolboxes, the Trading Toolbox, enables a MATLAB program to connect to a number of brokerages' APIs to receive market data, submit orders, and receive order status notifications. If you prefer not to buy the Trading Toolbox, there are at least three adaptors developed by third-party vendors that enable you to do the same: exchangeapi.com's quant2ib, undocumentedmatlab.com's IB-Matlab, and Jev Kuznetsov's free MATLAB-to-IB API available at MATLAB's File Exchange. I have discussed these options in some depth in Chan (2013). Finally, MATLAB is fast. (See Chapter 6 for

a comparison of performance speed among MATLAB, R, and Python.) The only drawback for this platform is that it isn't free, but the "Home" license costs only $150, with each additional toolbox costing an extra $45. If you plan to buy MATLAB's Toolboxes, here are the three I recommend (in decreasing order of importance): Statistics and Machine Learning, Econometrics, and Financial Instruments (for options traders).

For those who prefer free, open-source platforms, there are always R and Python.

R is very similar to MATLAB: It is an array-processing language, and it has a large variety of specialized "packages" (the analogue of MATLAB's toolboxes), many of them perhaps more sophisticated than MATLAB's due to the large number of academic researchers who use R. There is a GUI development platform called RStudio, but I find its user interface to be quite crude compared to that of MATLAB, and hence debugging productivity is lower. R is also the slowest among the three languages, and the slowness is all the more problematic because, unlike MATLAB or Python, it cannot be compiled into C or C++ code for speeding up. (You can, however, use the Rcpp package to access compiled C++ code from within R.) As for automating executions, you can connect an R program to Interactive Brokers through a package called "IBroker."

Python is a language in the ascendant, though I know of quants who used it for backtesting back in 1998. Aside from being a standalone language of choice for many quantitative traders, platforms such as Quantopian also use it as their strategy specification language. Native Python is not an array processing language (though one can use SciPy packages which do have this feature). While array processing is convenient for backtesting a large number of instruments simultaneously (e.g., portfolio of stocks), it is not useful for writing an automated execution program. Hence, if we were to insist on using the same program for both backtesting and live execution, we can disregard this feature altogether. One major advantage of Python is that its codes can be developed and debugged using Microsoft's Visual Studio, thus piggybacking on the full power of this well-polished development environment. Another integrated development environment[2] (IDE) for Python that received good reviews is PyCharm. Python's pandas library is a data analysis package similar to R, and the rpy2 package provides an interface to access all R packages. Python isn't as fast as MATLAB, though it is faster than R, and can be compiled into C or C++.[3] You can connect a Python trading program to Interactive Brokers for executions via IBPy or a number of other packages.

TABLE 1.1 **Ranking of Three Programming Languages for Quant Trading (Ranked from 1 to 3 where 1 = best ranking and 3 = poorest ranking.)**

Feature	MATLAB	R	Python
Ease of use	1	2	2
IDE polish	1	3	1
Speed	1	3	2
Toolboxes	2	1	1
Compilation to C/C++	1	N/A	1
Connectivity to brokers	1	2	2
Customer support	1	N/A	N/A
Price	2	1	1

In Table 1.1, I provide my personal, arguably subjective, ranking of the various features and aspects of the three scripting languages most widely used in trading strategy development.

You may be wondering why I have left out some of the most common programming languages such as C/C++, Java, or C#. The fact is, the most productive way to develop trading strategies is to quickly build prototype programs to test a lot of them, and to quickly discard them if they fail to live up to their promises. Scripting languages (also called REPL[4] languages) like MATLAB, R, or Python allow us to shorten this research cycle. However, productivity in research is not the same as productivity in building an automated trading system. The latter usually involves the usual bells and whistles of software development such as object-oriented design, exception handling, version control, memory and speed optimization, and so forth. Such bells and whistles are actually quite cumbersome to implement in a scripting language, and, furthermore, it is quite easy to introduce bugs without a robust and extensible software architecture. Typically, once object-oriented design is imposed on a scripting language, it will run too slowly to be useful as an execution system.

To benefit from the best of both worlds, in our firm we do our initial research mostly in MATLAB or Python (though I often ask our research associates to test their strategies on Quantopian first, just to make sure their codes do not have look-ahead bias). After we settle on a strategy, our CTO, Roger, will then independently build a system in C# that can both backtest

and execute live the same strategy, as a way to confirm the correctness of the initial backtest. Once it is confirmed, a figurative turn of the key will allow us to trade live, with much lower latency than if we were to execute using a scripting language. In building the execution system, we can often reuse existing classes that we have written for other strategies. This way, we have compressed the research life cycle without sacrificing software correctness, component reuse, or execution efficiency.

There are now many backtesting and automated execution platforms available that purport to make it easier to develop and deploy automated trading strategies both quickly and robustly, just like what I described in the last paragraph, but without using two different languages. I have written extensively about this in Chan (2013), so here I will restrict myself to those platforms that fit the following criteria:

1. It has integrated historical and live market data, or provides adaptors for popular data vendors.
2. It allows maximum flexibility in strategy design by relying on generic programming languages such as Python or Java.
3. It allows connection to popular broker APIs.
4. It allows backtesting and live trading in US equities, among other instruments.

The platforms that satisfy these criteria are Quantopian, QuantConnect, Ninjatrader, Marketcetera, AlgoTrader, OpenQuant, Deltix, and QuantHouse. (This is, of course, not an exhaustive list.) Some of these platforms, such as Quantopian, are free, while others, such as QuantHouse, have a price tag suitable for institutional traders.

If you are trading a high frequency strategy, it is possible that many of these platforms will not be adequate: either because they may be too slow, or because they are missing level 2 quotes. But there are platforms such as Lime Brokerage's Strategy Studio that are designed specifically for such demanding tasks. More about this will be discussed in Chapter 6.

Though these platforms have certainly made it easier to backtest and automate trading strategies, they usually come with some constraints on the exact type of strategies, data, or instruments that we are allowed to test. Not surprisingly, the more expensive a platform is, the more flexible it is. But if absolute flexibility coupled with low development cost is required, you will probably have to do what we do in our firm.

Brokers

In this day and age, practically every broker will offer customers an API where they can subscribe to market data, submit orders, and receive order status notifications: See the last two blocks on the right of Figure 1.1. Of course, the ease of use and comprehensiveness of their APIs differ widely. If you would rather not deal directly with the vagaries and low-level details of each broker's API, you can use one of the automated execution platforms that I described in the previous section.

Meanwhile, commissions are generally so low that it also isn't a crucial factor in deciding who to trade with. Brokers have other ways to make money from you besides commissions. One popular way, for some stockbrokers, is "payment for order flow." Essentially, when you send your order to this broker, it will forward your order to a particular market maker or a specific exchange for a rebate (e.g., a penny per share), and let them execute this order. Brokers who do this must inform their customers of this practice when they open their accounts. If you prefer not to let your broker earn this rebate, and instead want to pocket it yourself, you can use brokers with Direct Market Access (e.g., Interactive Brokers or Lime Brokerage). For some FX brokers, a popular way to earn money is to widen the bid-offer spread on a currency pair. This way, they earn the difference between the spread they offer you and the spread they have to pay in the interbank market where they execute your trade. This can make it hard to compare costs among different FX brokers, since this extra spread may change over time, but of course this is precisely the reason why some brokers adopt the practice. If you prefer transparency, you can restrict your search to FX brokers that only charge commissions, not spread.

If you are a futures or currency trader, there is one more item about your broker that you have to worry about: financial stability. For US securities (i.e., stocks trading) accounts, many traders know that their cash[5] is insured up to $250,000 by the Securities Investor Protection Corporation (SIPC). Many brokers also provide additional insurance through a private insurance company such as Lloyd's of London. However, none of this is applicable to commodities futures trading accounts—hence the furor in wake of the MF Global and PFGBest bankruptcies (MarketWatch, 2012). It would be pointless to advise you to ascertain the financial strength of a commodities broker before opening an account: If the National Futures Association (NFA) couldn't detect that some of these large brokers (called Futures Commission Merchants, or FCM) are not meeting capital requirement or are

committing fraud, what chance do we have? However, there is one way out of this conundrum: If a securities broker is also an FCM, it might offer a way to automatically sweep the excess cash[6] from an uninsured commodities account into an insured securities account. (For example, Interactive Brokers offers such a cash sweep service.)

Currency traders may have heard of *Herstatt risk,* or settlement risk (Galati, 2002). To illustrate, let's say we have sold 1 million EURUSD to a bank, and have delivered 1 million EUR to it. But before this bank can send us the corresponding amount of USD, it collapses (as was the case with Bankhaus Herstatt in Germany, 1974). We won't be getting our USD any time soon, and have lost 1 million EUR. It is a risk that is not solved by having an account at a reputable FX broker, because your counterparty may be some bank in a faraway country, and your broker isn't liable for your loss. This scenario is often used by some uninformed financial advisors to scare customers off investing in foreign currency strategies or funds. However, since the establishment of the CLS bank in 2002, an institution owned by some of the largest banks in the world, this risk has been largely eliminated. The CLS bank is a US financial institution regulated by the US Federal Reserve, and it facilitates simultaneous settlement of both legs of a currency transaction. It can do so because it maintains accounts with 18 central banks around the world. In our example above, we will receive the USD payment at the same time our counterparty receives our EUR payment, all via the CLS bank. It is almost like transacting on a stock exchange, where we are guaranteed that if we sold some shares, payment for those shares would be forthcoming, even if the trader who bought our shares went bankrupt. The 18 central banks whose currencies settle in CLS are listed at www.cls-group.com/About/Pages/Currencies.aspx, and these are the currencies that have little or no settlement risk.

We have been discussing the risks that other parties (brokers or counterparties) fail to pay us in a transaction. But what if we are the ones who fail? More specifically, what if in a levered transaction, we lost more money in the brokerage account than our account equity? Are we liable to the negative equity? Can we lose more than what we invested in the account? The answer is yes and yes, if we invested as individuals. In a particularly poignant example, on January 15, 2015, the Swiss National Bank (Switzerland's central bank) suddenly announced that the Swiss Franc would no longer be pegged to the euro (Economist, 2015). EURCHF plummeted by about 10 percent in seconds, and ultimately lost 29 percent in a day. Many FX traders' account equities went negative. Some smaller FX

brokers failed because their customers wouldn't pay them back the losses. In a situation like this, how do we make sure we won't be liable? The way to protect ourselves is not to invest in our own personal or family's name, but through a limited liability vehicle that we fully own, such as a corporation (or S-corporation in the United States), limited liability company (in the United States), or limited partnership. Investing in someone else's limited partnership, such as a hedge fund, will also work. In case of negative equity, the limited liability vehicle will have to declare bankruptcy. This isn't great, but is not catastrophic.

Performance Metrics

What performance metrics should a backtest program generate? I am typically interested in only five of them: CAGR (compound annual growth rate), Sharpe ratio, Calmar ratio, maximum drawdown, and maximum drawdown duration. I have discussed most of these except the Calmar ratio in detail in my previous books, so I will just briefly highlight some of their usage here.

CAGR is a bit different from average annualized returns: It assumes we do not transfer cash into or out of an account each time period, while maintaining the same leverage throughout. To take an extreme example, if a strategy returns 1 percent per trading day, CAGR will be $1.01^{252} - 1 = 1127$ percent. That's compounding at work. On the other hand, the average annualized return would be just $252 \times 0.01 = 252$ percent, and it is the return we would get if we withdraw profit or add cash to make up for losses each time period. But I emphasize that we must keep the leverage constant to achieve this compounded growth. In other words, your positions or orders must be resized each day based on the account equity and the leverage—something that an automated trading program should be able to do quite easily.

In a backtest, I recommend we set the leverage to one. Otherwise, the higher the leverage we use, the higher will be the CAGR, up to a point as determined by the Kelly formula below. So it is quite meaningless to pick an arbitrary leverage for a backtest. But to ensure that the leverage is one, one must make sure that the returns are measured by taking the Profit and Loss (P&L) and divide that by the total gross market value of the position(s). For example, if we are long $100 of stock A while short $100 of stock B, and the P&L is $1, the unlevered return is just 0.5 percent.

I have written a lot before about using the Kelly formula (Chan, 2009):

$$Optimal\ leverage = \frac{Mean\ of\ Excess\ Returns}{Variance\ of\ Excess\ Returns}$$

If we leverage our strategy or portfolio higher than this optimal leverage, CAGR may start to go down with increasing leverage, and is almost guaranteed to be −100 percent when the leverage is high enough. But often, it isn't safe even if the leverage is equal to the Kelly optimal. I have found that Kelly's leverage is typically too high to be of practical use. The Kelly formula shown above assumes Gaussian distribution of returns: Risk is measured only by the second moment of returns. So let's consider a simulated excess returns series with a mean of about 0.05 percent and standard deviation of about 1 percent. This is not too different from the actual daily returns statistics of SPY minus the risk-free rate. Kelly formula would therefore recommend a leverage of 5. But suppose one of the days has a return of −20 percent, which is also close to the actual return of SPY on Black Monday, October 19, 1987. If we are long SPY with a leverage of 5, our equity would have been wiped out on that day, no thanks to Kelly. As a practical matter, I would recommend lowering the leverage until you are comfortable with the maximum drawdown in the backtest over a period that includes several financial crises.

Sharpe ratio is a risk-adjusted return measure that everybody uses, but its utility is limited because it also implicitly assumes that the returns distribution is Gaussian. Risk is again measured only by the standard deviation (volatility) of returns. But as everyone knows, returns distribution have "fat tails": abnormally large returns of either sign occur far more frequently than is predicted by a Gaussian distribution. Maximum drawdown is a much better measure of tail risks, as it does not rely on measuring just the second moment of returns distribution. (Maximum drawdown is also asymmetric: We are not concerned about periods with extremely positive returns.) Hence, I have been in favor of using the Calmar ratio instead: it is defined as the CAGR divided by the absolute value of the maximum drawdown over the most recent three years. Some people prefer to use the MAR ratio instead, which is the same as the Calmar ratio except that the maximum drawdown is over the entire history of the backtest (or live track record). One can see that Calmar ratio is better defined than the MAR ratio: The longer the

backtest, the higher the absolute value of the maximum drawdown. So it doesn't quite make sense to have a measure that depends sensitively on the length of a backtest or a track record. Three years is a way of normalizing this measure. I prefer to trade strategies that have a backtest Calmar ratio of at least 1.

Portfolio Optimization

When we are trading multiple strategies, or holding positions in multiple instruments in a portfolio, how we allocate capital among them can be as important to our total return as how these strategies perform individually. This issue of portfolio optimization has been well-studied in academic finance research, although there are some subtleties that are worth highlighting.

Before we get into the details, we should differentiate between maximizing the return of a portfolio over one period (e.g., a day, a month, or a year) versus maximizing the compound return (CAGR) of the portfolio over an infinite horizon. The former is dealt with by the familiar Markowitz portfolio optimization method. Its objective is to maximize the expected return subject to a constant variance of returns constraint; or equivalently and more conventionally, to minimize the variance of returns subject to a constant expected return constraint.

In order to compute the expected return and variance of returns over one period, let's assume our portfolio has N instruments (stocks, bonds, futures, etc.) or strategies (mean-reverting, momentum, event-driven, options, etc.) From here on, we will use the generic term *instruments* to denote such components of a portfolio. Each instrument i has an expected return m_i. We assume that these returns have a covariance $C_{i,j}$. Note that by return here we mean net return, not log return. This distinction will become important later on. To clarify, net return in this context is calculated as

$$\textit{Net return}(t) = \frac{\textit{Price}(t)}{\textit{Price}(t-1)} - 1,$$

while log return is calculated as

$$\textit{Log return}(t) = \log(\textit{Price}(t)) - \log(\textit{Price}(t-1)).$$

Net returns can never have Gaussian distributions in principle (the "why" is left as an exercise), but log returns can. There is an interesting relationship between the mean log return μ and the mean net return m of a price series if the log returns have normal distribution:

$$\mu \approx m - \frac{s^2}{2} \tag{1.1}$$

where s is the standard deviation of the net return. This is also approximately equal to the standard deviation of the log return; hence, we can use the same symbol to represent both. The approximate equality in equation 1.1 will become exact as we subdivide the periods into smaller and smaller subperiods, so that we approach continuous time (see Box 1.1). It can be derived from Ito's Lemma, famous in continuous finance as a foundational formula for Black-Scholes options pricing formula. This equality will be useful when we try to maximize the expected compound growth rate later on.

Box 1.1: The mean of net vs. log returns

In this example, we will demonstrate numerically that equation 1.1 is approximately true when log returns have a Gaussian distribution, and that as we subdivide a time period into more and more subperiods, the approximation becomes exact.

Let's say that the log return over some time period has a mean of 100 percent and a standard deviation of 200 percent. We will use the *randn* function to simulate N subperiod log returns r_i with a mean of $\mu = 100\%/N$ and a standard deviation of $s = 200\%/\sqrt{N}$. We will try N = 100, 10,000, and 1,000,000.

```
r=1/N+2/sqrt(N)*randn(N, 1);
```

Their corresponding net returns R_i are just $e^{r_i} - 1$:

```
R=exp(r)-1;
```

You can compute the sample mean and standard deviation of the log returns

```
mu=mean(r);
sigma=std(r);
```

TABLE 1.2 Mean of Net vs. Log Returns

N	m	μ	s	$m - s^2/2$
100	2.00e-2	2.99e-4	2.00e-1	-6.41e-04
10,000	2.24e-4	2.30e-5	2.00e-2	2.28e-05
1,000,000	4.66e-6	**2.65e-6**	2.00e-3	**2.65e-06**

and compare these to sample mean and standard deviation of the net returns

```
m=mean(R);
s=std(R);
```

and you will find that $m - s^2/2 \to \mu$, as $N \to \infty$. This is shown in the Table 1.2, where I have boldfaced the two quantities that converged.

The complete code can be downloaded as netVsLogRet.m from epchan.com/book3. The userid and password are both calmar.

If we denote by F_i the capital (or leverage) allocated to an instrument i, then the expected return of the portfolio over one period in matrix notation is simply $F^T M$, where F is the column matrix of F_i and M is the column matrix of m_i. Similarly, the expected variance of returns for the portfolio is $F^T C F$, where C is the covariance matrix $C_{i,j}$. For each fixed level of the expected return of the portfolio, we can minimize the expected variance of returns by varying F using a numerical quadratic programming method as illustrated in Box 1.2. The result of this constrained optimization can be plotted on a chart (Figure 1.2) that shows the portfolio's expected return against its expected minimum variance. This curve has a famous name: the *efficient frontier*. But what is the best trade-off between the expected return and variance? Markowitz said we should choose the *tangency portfolio* that maximizes the Sharpe ratio: that is, the ratio between the expected return[7] and the variance of the return. The tangency portfolio is noted on Figure 1.2, and it is so-called because if we draw a line from the origin representing a riskless portfolio (which has an expected return equal to the risk-free rate assumed here to be zero, and a variance of zero) to the efficient frontier, this line intersects the efficient frontier at a tangent right at the

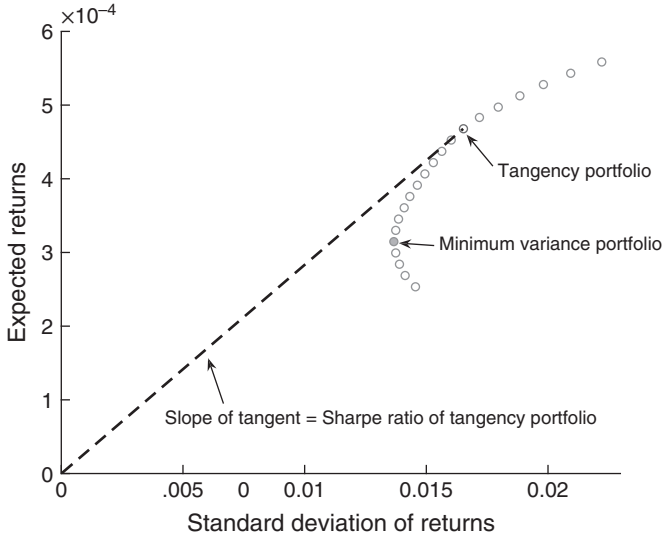

FIGURE 1.2 Efficient frontier

tangency portfolio. The slope of this line is the Sharpe ratio of the tangency portfolio.

Box 1.2: Calculating the efficient frontier using quadratic programming

In this example, we will numerically determine the efficient frontier of a portfolio of ETFs, with the constraint that no short position is allowed, and the weights of all the ETFs must add to 1 (i.e., no leverage on the portfolio is allowed). The set of ETF we use are EWC, EWG, EWJ, EWQ, EWU, and EWY. The efficient frontier is the set of minimum variances given a set of fixed expected returns. The first task in constructing the efficient frontier is to find the maximum and minimum expected returns possible. Given the no-leverage constraint, we know that the maximum (minimum) expected return is obtained when we have a weight of 1 on the ETF with the highest (lowest) expected return, and a weight of 0 on all other ETFs. If R is the array with each column representing the daily returns of one ETF over many days, we can compute the mean and the covariance of returns as

```
mi=mean(R, 1); % average return of each stock i.
C=cov(R); % covariance of returns
```

and the set of expected returns we want to consider for our efficient frontier is

```
m=[min(mi):(max(mi)-min(mi))/20:max(mi)];
```

These expected returns are the constraints for our variance minimization function. We will use MATLAB's Optimization Toolbox's quadratic programming function *quadprog* for this task. The objective function to be minimized is a quadratic function of the capital allocation vector *F*. From MATLAB's help manual, `X = quadprog(H,f,A,b,Aeq,beq)` attempts to solve the quadratic programming problem:

$$\min 0.5\, x'H\, x + f'\, x \quad \text{subject to: } Ax \leq \text{b and } Aeqx = beq.$$

In our problem, we want to solve the quadratic programming problem

$$\min F^T\, C\, F \text{ subject to: } F \geq 0 \text{ and } F^T\, M = m,$$

where F, C, and M are defined in the main text as the capital allocation vector, the covariance matrix, and the column matrix of each ETF's expected return, while m is the expected return of the portfolio. Hence, we have the mapping

$$x = F,\ H = 2\, C,\ f = 0,\ A = -I \text{ (negative of the identity matrix)},$$
$$b = 0,\ Aeq = M, \text{ and beq} = \text{m. In MATLAB,}$$

```
H=2*C;
f=zeros(1, length(mi));
A=-eye(length(mi));
b=zeros(length(mi), 1);
Aeq=[mi; ones(1, length(mi))];
beq=[m(i); 1];
```

Putting these in a loop over each $m(i)$ gives

```
for i=1:length(m)
    beq=[m(i); 1];
    [F, v(i)]=quadprog(H, f, A, b, Aeq, beq);
end
```

where the output *F* is the optimal allocation we desire, and $v(i)$ is the corresponding minimum variance. We have plotted the set of *F* and *v* results in Figure 1.2.

The tangency portfolio is found by maximizing the Sharpe ratio:

```
sd=sqrt(v);
sharpeRatio=m./sd;
[~, idx]=max(sharpeRatio);
beq=[m(idx); 1];
[F]=quadprog(H, f, A, b, Aeq, beq)
```

The resulting weights are $F = (0.45, 0.26, 0.00, 0.00, 0.00, 0.29)$. The minimum variance portfolio to be discussed in the main text is found by (what else?) minimizing the variance or standard deviation:

```
[~, idxMin]=min(sd);
scatter(sd(idxMin), m(idxMin), 'green', 'filled');

beq=[m(idxMin); 1];
[F]=quadprog(H, f, A, b, Aeq, beq)
```

The resulting allocations are $F = (0.38, 0.00, 0.60, 0.00, 0.01, 0.00)$.

You can see that both F's satisfy the constraints that their components are all non-negative and sum to 1. The complete code can be downloaded as ef.m.

Maximizing the Sharpe ratio may seem intuitively like a good thing, but actually what many investors (like myself) want is to maximize the expected compound growth rate of the portfolio over an infinite (or indefinite) time horizon. Fortunately, it turns out that maximizing the expected compound growth rate is equivalent to maximizing the Sharpe ratio. To prove this, we first recognize that the expected compound growth rate is just μ, the expected log return (see exercise 1.21 for proof). By equation 1.1, this is equal to $F^T M - \frac{1}{2} F^T C F$ for the portfolio. To maximize this, we take the partial derivative with respect to each F_i and setting it to zero. The solution is $F^* = C^{-1} M$, where F^* denotes the optimal allocation. But this is precisely the Kelly optimal allocation I wrote about in Chan (2009). To show that this allocation also maximizes the Sharpe ratio under the constraint of a finite, fixed, leverage, note that the Sharpe ratio of the portfolio is $F^T M / (F^T C F)^{1/2}$. We want to maximize this, subject to the constraint that the sum of F_i is 1, or $F^T 1 = 1$ (where 1 is a column vector of 1s, and it won't matter if we set $F^T 1$ to some other arbitrary constant that represents the leverage we apply). Using Lagrange multipliers, it is shown in Box 1.3 that $C^{-1} M / \mathbf{1}^T C^{-1} M$ is also a solution.

In other words, it is the same as the Kelly optimal allocation F^*, but in a normalized form such that the total leverage is 1. So F^* doesn't just maximize the compound growth rate, it also maximizes the Sharpe ratio and specifies the tangency portfolio (to within a constant factor). It turns out that maximizing 1-period return subject to a variance constraint has the same solution as maximizing compound growth over an infinite horizon. This maximum compound growth rate can be expressed in terms of the maximum Sharpe ratio: It is just $\frac{S^{*2}}{2}$.

Box 1.3: Maximizing the Sharpe ratio of a portfolio

(This exposition is based on my blog post "Kelly vs. Markowitz Portfolio Optimization": epchan.blogspot.com/2014/08/kelly-vs-markowitz-portfolio.html, which in turn was drawn from faculty.washington.edu/ezivot/econ424/portfolioTheoryMatrix.pdf.)

In Box 1.2, we have minimized the variance of a portfolio for a given level of expected return. We did this numerically, using quadratic programming techniques. However, if we simply want to find the capital allocation corresponding to the tangency portfolio, we can do this analytically by maximizing the Sharpe ratio of the portfolio, subject to a constant leverage constraint. (The constant leverage can be set to 1 without loss of generality.) In other words, we want to maximize the Sharpe ratio

$$F^T M/(F^T C F)^{1/2} \text{ subject to: } F^T \mathbf{1} = 1$$

where $\mathbf{1}$ is a column vector of 1s.

Using the Lagrange multiplier method, this is the same as an unconstrained maximization of

$$\frac{F^T M}{(F^T C F)^{\frac{1}{2}}} - \lambda(F^T \mathbf{1} - 1), \tag{1.2}$$

where λ is the Lagrange multiplier.

Normally, the next step would be to take the derivative of the expression 1.2 with respect to each F_i. But taking the partial

derivatives of this fraction with a square root in the denominator is unwieldy. To make things easier, we can equivalently maximize the logarithm of the Sharpe ratio subject to the same constraint, resulting in

$$\log(F^T M) - \frac{1}{2}\log(F^T CF) - \lambda(F^T \mathbf{1} - 1). \tag{1.3}$$

Taking the partial derivatives of equation 1.3 with respect to F_i, and setting each partial derivative to zero gives

$$\frac{1}{(F^T M)} M - \frac{1}{F^T CF} CF = \lambda \mathbf{1}. \tag{1.4}$$

If we multiply this whole equation by F^T on the right, we get

$$\frac{1}{(F^T M)} F^T M - \frac{1}{F^T CF} F^T CF = \lambda F^T 1. \tag{1.5}$$

The left-hand side comes out to be exactly zero, while the right-hand side is just λ because $F^T \mathbf{1} = 1$ due to our constraint. Hence, λ must be zero. A Lagrange multiplier that turns out to be zero means that the constraint won't affect the solution of the optimization problem up to a proportionality constant. This makes sense because we know that if we apply an equal leverage on all the assets, the maximum Sharpe ratio shouldn't be affected. So equation 1.4 becomes

$$CF = \frac{(F^T CF)}{(F^T M)} M. \tag{1.6}$$

It is easy to see that $F = C^{-1}M$ is a solution to this equation, which is the Kelly optimal allocation formula I have described in Chan (2009), and which maximizes the Sharpe ratio of a portfolio of instruments.

All this classical financial mathematics is elegant and satisfying, but reality often intrudes. The main problem is simple to understand: Past returns are not a good predictor of future returns. So the optimal allocation for the past is not necessarily the optimal allocation for the future. On the other hand, covariances are easier to predict than returns. Also, while the estimates of expected returns do not improve with larger sample size, estimates of

covariances do (Ang, 2014). If we don't have returns estimates but still have covariances estimates, we can construct a *minimum variance portfolio*—that is, an allocation that minimizes the variance of returns of the portfolio. This is also marked on Figure 1.2. A side benefit of the minimum variance portfolio is that research has shown that low volatility stocks consistently outperform high volatility stocks (Baker et al., 2011). In the last 20 years, investors have increasingly favored minimum variance over tangency portfolio, giving rise to such ETFs as SPLV, ACWV, EEMV, and USMV.

Some investors have gone even further: They distrust even the covariance of returns and rely solely on the volatilities of individual stocks in making their capital allocation. Such is the case with the so-called *risk parity portfolio*, where capital allocated to each asset is inversely proportional to its volatility (Asness, 2012). In essence, this is targeting the same risk for each asset in the portfolio, hence the name *risk parity*. The ETF SPLV is constituted using this principle, and so is the famous $70 billion Bridgewater Associates All Weather fund. However, many risk management strategies have backfired due to their penchant for creating contagion. In other words, if everybody is adopting the same risk management strategy, everybody's risk will increase instead. For example, let's say everybody follows Kelly formula's advice to keep the leverage of their portfolio constant, and everybody's portfolio contains similar stocks. Then a loss in value of this portfolio will cause everybody to liquidate holdings, driving the (net asset) value of this portfolio even lower, and forcing further liquidation. The same problem is present for risk parity portfolios. If the market suffers a meltdown, volatility tends to increase, forcing risk parity funds to liquidate. Well, you know the rest of the story. This happened in the fall of 2015, when a combination of Chinese economic woes and crashing oil prices caused a market panic. The All Weather fund lost 7 percent in 2015, and was blamed for starting just such a contagion (Copeland and Martin, 2015).

In addition to the contagious liquidation problem discussed above, targeting the same volatility for each component of a portfolio also relies on the implicit Gaussian assumption that volatility is bad. But volatility isn't what we should be afraid of—tail risk is. As discussed in the section "Performance Metrics," maximum drawdown is a good measure of tail risk. So I suggest we should target the same maximum drawdown for each component of a portfolio if you are into the risk parity allocation method.[8]

Summary

This chapter touches on all the basic components of building an algorithmic trading strategy, from data to software, brokers, performance metrics, and finally to ways of building and optimizing a portfolio of assets or strategies. These issues are relevant to all the strategies that we will be discussing in the following chapters.

Exercises

1.1. What is an API?
1.2. What is a good IDE for Python?
1.3. What is the most useful MATLAB toolbox for an algorithmic trader?
1.4. Can Python access all of R's packages? If so, how?
1.5. Among MATLAB, R, and Python, which one is the slowest?
1.6. How would you speed up a computationally intensive MATLAB or Python program? Can you do the same for R?
1.7. What is a possible platform you can use to backtest and automate a live trading strategy if you do not wish to learn the details of a broker's API?
1.8. What is the benefit of using the same program to backtest and live-trade a strategy?
1.9. Suppose you have a stock selection strategy on SPX stocks. What data would you need to ensure that your backtest is survivorship-bias-free? Would it be sufficient to have prices for all the stocks that were ever listed in the US exchanges?
1.10. What is the best kind of closing prices to use for testing daily strategies? What is the second best kind? How do these differ from the usual consolidated closing prices?
1.11. What are the implied volatility surfaces that options data vendors often provide? How are they computed?
1.12. What is a REPL language, and why is it convenient for backtesting or research in general?
1.13. How much is a MATLAB home license? How much is each of its toolboxes under this license?
1.14. What is the difference between Calmar ratio and MAR?
1.15. Do FX transactions suffer from settlement risks since 2002?

1.16. What is the CLS bank?
1.17. Are commodity futures trading accounts protected by SIPC insurance?
1.18. What is the benefit of opening a brokerage account in the name of an LLC or LP?
1.19. Why can't net returns ever have a Gaussian distribution in principle?
1.20. Prove that the expected market value at time t of a security that undergoes a geometric random walk from an initial value of \$1 is e^{mt}, with m is the expected net return over a unit time interval. (*Hint:* Divide a time interval into n segments, and express the market value as an exponential of the sum of log returns. Also, remember that if Z is a Gaussian random variable with mean μ and standard deviation s, then e^Z has mean $e^{\mu+s^2/2}$ and standard deviation e^s. Finally, make use of equation 1.1.)
1.21. Prove that the expected compound growth rate of a security that undergoes a geometric random walk is μ, its expected log return over a unit time interval. (*Hint:* The expected compound growth rate is defined as $\mathrm{E}[\log(V(t)/V(0))]/t$, where $V(t)$ is the market value at time t. Express this expectation as a sum of log returns by dividing a time interval into n segments, and take the limit as $n \to \infty$.)
1.22. What is the formula for the Kelly optimal allocation? What does this allocation maximize? What's the name of the portfolio that uses Kelly optimal allocation?
1.23. Why is the minimum variance portfolio often favored over the tangency portfolio?
1.24. What is the difference between a minimum variance portfolio and a risk parity portfolio?
1.25. Why is it possible to generate a contagion in a financial crisis if everybody owns a risk 8 portfolio?
1.26. What is the drawback of measuring risk by computing volatility of returns? What are the alternatives?

Endnotes

1. If we are backtesting a portfolio strategy for stocks in a stock index, we will also need to know when the stock was added and/or removed from that index. Just having historical prices for delisted stocks won't be enough. Check out www.masteretfdata.com for historical index composition.

2. Integrated Development Environment, or IDE, is essentially an editor and debugger for programmers. It is crucial for software development productivity.
3. Numba is one of the Python packages that facilitates compilation to C/C++, and NumbaPro is the commercial version that allows parallel processing using a GPU (graphics processing unit).
4. REPL (Read-Eval-Print-Loop) languages do not require compilation; hence, debugging and modifications can be done much faster.
5. The maximum coverage for securities and cash combined is $500,000.
6. Excess cash means cash balance that exceeds the margin requirement for the existing futures positions.
7. Strictly speaking, we need to subtract the risk-free rate from the returns before taking the mean and variance. But since 2008, we can just assume the risk-free rate is zero.
8. For the financial sophisticates, there are also value-at-risk (VaR) and expected shortfall (ES) that one can use as stand-ins for volatility. These also do not rely on the Gaussian assumption. (See Ruppert, 2015.)