Contents

Contents 3

Introduction 9

The Target Audience, and Overview of Contents 10

What's In This Book 10

What's Not In This Book 12

About Trading Systems 13

Market Prices and Returns 14

Two Types of Automated Trading Systems 15

The Agony of Believing the Computer 16

Future Leak is More Dangerous Than You May Think 17

The Percent Wins Fallacy 18

Pre-Optimization Issues 19

Assessing and Improving Stationarity 20

The STATN Program 22

Improving Location Stationarity By Oscillating 26

Extreme Stationarity Induction 28

Measuring Indicator Information With Entropy 29

Computing the Relative Entropy of an Indicator 32

Entropy Impacts Predictive Model Quality 33

Improving the Entropy of an Indicator 35

Monotonic Tail-Only Cleaning 38

The ENTROPY Program 41

Optimization Issues 43

Regularizing a Linear Model 44

Overview of the Regularized Model 45

Beta Adjustment With Guaranteed Convergence 50

Differential Case Weighting 51

Rapid Computation With Covariance Updates 52

Preparatory Code 53
Outline of the Beta Optimization Process 56
Code For Beta Optimization 58
Descending a Lambda Path 65
Optimizing Lambda With Cross Validation 69
The CD_MA Program 73
Making a Linear Model Nonlinear 77
Differential Evolution: A Universal Nonlinear Optimizer 79
The DIFF_EV.CPP Routine for Differential Evolution 85

Post-Optimization Issues 101
Cheap Bias Estimates 102
The StocBias Class 102
Cheap Parameter Relationships 108
Parameter Sensitivity Curves 121
Putting It All Together Trading OEX 125
The Trading System 125
Linking Criterion Routines 128
Application to Trading OEX 130

Estimating Future Performance I:
Unbiased Trade Simulation 135
In-Sample and Out-of-Sample Performance 136
The TrnBias Program to Demonstrate Training Bias 138
Selection Bias 140
Interlude: What Does *Unbiased* Really Mean? 141
Selection Bias, Continued 142
The SelBias Program 144
Walkforward Analysis 146
Future Leak by Unobvious IS/OOS Overlap 148
Error Variance Inflation with Multiple-Bar Lookaheads 150
The General Walkforward Algorithm 152
C++ Code for the Algorithm 153

Date-Dependent Walkforward 155
Exploring Walkforward Blunders 156
Testing Robustness Against Nonstationarity. 162
Cross-Validation Analysis 163
Unobvious IS/OOS Overlap 163
The Fully General Cross-Validation Algorithm 164
C++ Code for the General Algorithm 166
Cross Validation Can Have Pessimistic Bias 170
Cross Validation Can Have Optimistic Bias. 170
Cross Validation Does Not Reflect Real Life 171
Special Precautions for Algorithmic Trading. 172
Converting Unknown-Lookahead Systems to Single-Bar . . . 176
Unbounded Lookback Can Subtly Happen. 177
Comparing Cross Validation With Walkforward: XVW 178
Computationally Symmetric Cross Validation 180
The CSCV Algorithm: Intuition and General Statement 181
What Does This Test Actually Measure? 186
C++ Code for the CSCV Superiority Test 187
An Example with SPX 192
Nested Walkforward Analysis. 195
The Nested Walkforward Algorithm 197
A Practical Application of Nested Walkforward 203
An Example Using S&P 100 Components 212
Cross Validation Nested Inside Walkforward. 213

Estimating Future Performance II:
Trade Analysis 217
Handling Dynamic Trading Systems 218
Unknown Lookahead to Single Bars, Revisited 218
Profit Per Bar? Per Trade? Per Time? 220
Analyzing Completed Trade Returns is Problematic. 221
The PER_WHAT Program 223

A Lower Bound for Mean Future Returns 235
Brief Digression: Hypothesis Tests 236
So How *Do* We Use This Probability? 239
Parametric P-Values. 243
Parametric Confidence Intervals 246
Lower Confidence Bounds and Hypothesis Tests 250
Bootstrap Confidence Intervals 251
The Pivot and Percentile Methods. 251
The BC_a Bootstrap Algorithm. 254
The BOOT_CONF.CPP Subroutines 257
The BOUND_MEAN Program and Results With SPX. 262
Beware of Bootstrapping Ratios. 269
Bounding Future Returns 273
Deriving a Lower Bound From Empirical Quantiles 274
Confidence in the Computed Lower Bound 276
What About an Upper Bound on Future Returns? 280
The CONFTEST Program: Overview 281
The CONFTEST Program: Code 285
The BND_RET Program 291
Bounding Drawdown 296
Intuition Gone Wrong 298
Bootstrapping Drawdown Bounds 300
The DRAWDOWN Program 303
Experiments With the DRAWDOWN Program 313
The CHOOSER_DD Program 315

Permutation Tests 319
Overview of Permutation Testing 320
Testing a Fully Specified Trading System 322
Testing the Training Process. 323
Walkforward Testing a Trading System Factory 324
Permutation Testing of Predictive Models. 327

The Permutation Testing Algorithm 330
Extending the Algorithm For Selection Bias 331
Partitioning Total Return of a Trading System 333
Essential Permutation Algorithms and Code 338
Simple Permutation 338
Permuting Simple Market Prices 339
Permuting Multiple Markets With an Offset 341
Permuting Price Bars 346
Example: P-Value and Partitioning 351
Example: Training With Next Bar Returns 354
Example: Permuting Multiple Markets 358

Index 361

1 Introduction

Before we delve into the meat (or tofu, if you prefer) of this book, we should be clear on what you will and will not find here, as well as what degree of preparation is expected of readers.

The Target Audience, and Overview of Contents

This book is intended for readers who have a modest statistics background (Stat 101 is plenty), have some programming skill in any language (C++ with a strong bent toward traditional C is used in the examples here), and are interested in trading financial markets with a degree of mathematical rigor far beyond that of most traders. Here you will find a useful collection of algorithms, including sample code, that will help you tweak your ideas into trading systems that have above-average likelihood of profitability. But there are many things that you will *not* find in this book. We begin with an overview of the material included in this book.

What's In This Book

- If your system involves optimization of parameters, and most do, you will learn how to determine if your optimized system has captured authentic market patterns, or if it has simply learned random noise patterns which will never again appear.

- You will learn how to modify linear regression in a way that makes it even less susceptible to overfitting than it already is, and that as a bonus separates predictors into those that are valuable and those that are worthless. You will also learn how to modify linear regression to enable its use in moderately nonlinear situations.

- You will discover an extremely general and powerful nonlinear optimization algorithm that is applicable to both predictive-model-based trading systems as well as traditional algorithmic systems.

- All trading systems assume a degree of consistency in the market being traded; if the pattern on which your system is based has occurred regularly over recent history, we must assume that this same pattern will continue into at least the near future. Some trading systems are robust against moderate changes in market patterns, while other systems are rendered worthless by even tiny changes in market patterns. You will learn how to assess the degree to which your system is robust against such changes.

- If you have designed your own proprietary indicators, you will learn how to confirm that they are reasonably stationary (a critical property for any effective indicator), or massage them into stationarity if they are not. You will also learn how to compute them so as to maximize their information content, minimize their noise, and supply them to your trading system in an effective, efficient manner so as to maximize their utility.

- Most trading system developers are familiar with walkforward testing. But not so many are aware that ordinary walkforward algorithms are often insufficient for correct validation of trading system candidates and can produce dangerously optimistic results for subtle reasons. You will learn how to embed one walkforward algorithm inside a second layer of walkforward, or perhaps embed a layer of cross validation inside a walkforward analysis. This 'validation-within-validation' scenario is often not only the best way to test a trading system, but the only truly correct way.

- You will learn how estimate the range of possible future profits that your system can be expected to produce. If you discover that your system has almost certain future profitability, but there is a high probability that this profit will be small relative to the risk incurred, you will know that your system is not yet ready to be traded.

- You will learn how to estimate the probability of catastrophic drawdown, even when your system is operating 'correctly'.

- You will learn about rigorous statistical testing algorithms that are resistant to the occasional large wins and losses that invalidate many 'traditional' validation algorithms.

- Many trading system developers prefer to use the 'spaghetti-on-the-wall' approach to trading system development. Although frequently scorned, this is actually a legitimate approach, as long as it is done intelligently. You will learn how to determine if the 'best' of numerous competing systems is truly worthwhile.

What's Not In This Book

- This book is not "An Introduction to Statistics for Market Traders" type of book. It is assumed that the reader is already familiar with concepts like mean and standard deviation, normal distribution, p-values from hypothesis tests, and so forth. Nothing more advanced than these concepts is required; the advanced statistical techniques presented here are built up from basic ideas that anyone who's passed Statistics 101 or even a good Statistics for Psychology course can handle. But if you have no idea what a standard deviation is, you will find this book rough going.

- This is also not "An Introduction to Trading Financial Markets" book. It is assumed that you know the meaning of terms like opening and closing a trade, long and short positions, and mean return per trade. If you are totally new to trading financial markets, you need to study background material before tackling this book.

- You will find little or nothing in the way of actual, proven trading systems here. Those are a dime a dozen, and usually worth the price. But if you have your own idea for a trading system, you will learn how to implement, test, and tweak it so as to maximize its profit potential.

- You will find no top-secret super-duper sure-fire indicators in this book. The few indicators presented are either common sense or widely available in the public domain. But if you have your own ideas for indicators, you will learn how to maximize their utility.

About Trading Systems

As different testing procedures are presented in this text, they will necessarily be demonstrated in the context of various trading systems. Please note the following items of interest:

- I am not endorsing any of these systems as money-makers. Rather, I am keeping the systems as simple as possible so that the focus can be on their testing, not on their practical utility. This book assumes that the reader has his or her own ideas for trading systems; the goal here is to provide advanced statistical methods for tweaking and rigorously testing existing systems.

- All of the trading systems used for demonstrations assume that we are working with day bars, but this is never a requirement. Bars can be any length, from a fraction of a second to months. In fact, most demonstrations use only the open or close of each bar, so applying these algorithms to trading tick data is feasible as well. Days bars are simply most convenient, and test data is most readily available as day bars.

- Most of the demonstration systems open and close trades on the close of a bar. Naturally, in real life this is difficult or impossible; a more fair and conservative approach is to make a trade decision on the close of a bar and open or close the trade at the open of the next bar. But that would add needless confusion to the algorithms shown here. Remember, our goal is to present statistical algorithms in the most straightforward context, keeping the spotlight on the statistical test. In most cases, small modifications to the implementation do not materially change the results of rigorous statistical tests.

- In these tests, trade costs (slippage and commissions) are deliberately omitted, again to keep the focus on the statistical test without added confusion. The supplied code and accompanying description make clear how trade cost can be incorporated into the computation if desired.

Market Prices and Returns

Most equity markets cover a wide range of prices, perhaps beginning their life trading at a few dollars a share, and trading today at hundreds or thousands of dollars a share after split adjustment. When we compute the return of a trade, we don't dare just subtract prices at the open and close of a trade. A $1 move from $1 to $2 is enormous, while a move from $150 to $151 is almost trivial. Thus, many people compute percent moves, dividing the price change by the starting price and multiplying by 100. This solves the scale problem, and it is intuitive. Unfortunately, it has a problem that makes it a poor method in many statistical analyses.

The problem with percent moves is that they are not symmetric. If we make 10 percent on a trade, and then lose 10 percent on the next trade, we are not back where we started. If we score a move from 100 to 110, but then lose 10 percent of 110, we are at 99. This might not seem serious, but if we look at it from a different direction we see why it can be a major problem. Suppose we have a long trade in which the market moves from 100 to 110, and our next trade moves back from 110 to 100. Our net equity change is zero. Yet we have recorded a gain of 10 percent, followed by a loss of 9.1 percent, for a net gain of almost one percent! If we are recording a string of trade returns for statistical analysis, these errors will add up fast, with the result that a completely worthless trading system can show an impressive net gain! This will invalidate almost any performance test.

There is a simple solution which is used by professional developers and which I will use throughout this book: convert all prices to the log of the price, and compute trade returns as the difference of these logs. This solves all of the problems. For example, a trade that captures a market move from 10 to 11 is 2.39789–2.30258=0.09531, and a trade that scores a move from 100 to 110 is 4.70048–4.60517=0.09531. If a trade moves us back from 110 to 100, we lose 0.09531 for a net gain of zero. Perfect.

A nice side benefit of this method is that smallish log price changes, times 100, are nearly equal to the percent change. For example, moving from 100 to 101, a 1 percent change, compares to 100*(4.61512–4.605)=0.995. Even the 10 percent move seen above maps to 9.531 percent. For this reason, we will treat returns computed from logs as approximate percent returns.

Two Types of Automated Trading Systems

Originally, all forms of automated market trading were what might be called *algorithmic* or *rule-based.* The system developer comes up with a set of rigorously defined rules which guided the opening and closing of positions. The rules might state that if some combination of conditions becomes true, one would open a long position, and hold that position until some other combination of conditions becomes true. One classic chestnut of algorithmic trading is a moving-average crossover system. One computes short-term and a long-term moving averages, takes a long position if the short-term MA is above the long-term MA, and takes a short position otherwise. Training this primitive trading system is performed by finding the short-term and long-term lookbacks which provide optimal performance on a historical dataset. Algorithmic systems, many involving dozens of conditions, are still in widespread use today.

In more recent times, many developers (including myself) have formed the opinion that *model-based* systems are more powerful, despite their common disadvantage that they frequently involve blind trust in black-boxes whose inner workings are largely unfathomable. In model-based automated trading we compute one or more (usually many more) *indicators* which are variables that look backward in time and measure market characteristics. These might include trend, volatility, short-term cyclic behavior, and so forth. We also compute a *target* variable which looks into the future and describes near-term market behavior. Targets might be things like the size and direction of market movement over the next bar or few bars. A target might also be a binary flag which tells us whether the market first touches a predefined profit goal before touching a protective stop. We then train a predictive model to estimate the value of the target variable, given the values of the indicator variables. In order to trade this system, we present the trained model with current values of the indicators and consider the model's prediction. If the prediction is strong enough (indicating confidence) we take a market position in accord with the predicted move.

The advantage of model-based trading over rule-based algorithmic trading is that we can take advantage of the many recent developments in the field of artificial intelligence, letting sophisticated programs running on powerful computers discover trading systems that are perhaps so complex or obscure that no human could possibly hope to discover and program in explicit rules. Of course, this comes at a high price: we often have no idea exactly what 'rules' the model has discovered, and we must accept the model's decisions on blind faith.

Because both styles of trading system development are in widespread use today, this text will cater to both schools of thought. Unavoidably, there are a few statistical tests presented here which are applicable to only one or the other. But an attempt is always made to design testing procedures that can be used by practitioners in either style.

The Agony of Believing the Computer

For many people, especially seasoned seat-of-the-pants traders, the most difficult part of moving toward automated trading is accepting the trade decisions of a computer when they conflict with your gut, not to mention your many years of successful trading. I'll give one specific example from my own personal experience. I had developed on contract a short-term intraday trading system. My extremely thorough, rigorous statistical testing of the system showed unequivocally that its profits were maximized when it was operated by taking numerous small profits while running the risk of occasional large losses (a very loose protective stop). This grated on the trader responsible for calling signaled trades onto the floor. He constantly bombarded me with his mantra of, "Cut your losses and let your wins run." That's a truism for some trading styles, but not for this particular system. He couldn't help himself; he kept overruling the computer's trade decisions. The system would call for a winning trade to be closed, but he would keep it open, hoping for an even larger gain. Or the market would move against an open position and he would close it out for a small loss long before the system's stop was hit. He kept telling me how much money would have been lost if he had let it keep sliding instead of cutting the loss early. The fact that the computer simulation that ran in parallel made a lot of money, while his modified version made much less, had no impact on his opinion. He'd been a successful discretionary trader for many years, he knew how to trade, and no #$%^ computer was going to tell him otherwise. Our relationship never succeeded. The moral of the story: forget automated trading if you don't have the guts to believe in it.

Future Leak is More Dangerous Than You May Think

FutureLeak is the illegal leakage of future knowledge into a testing procedure. It happens in the development and testing of a trading system when some aspect of future market behavior finds its way into a simulation of how a trading system will perform in real life. Since we will obviously not know the future when we are trading our system, this leakage results in optimistic performance estimates.

More than once I have been amazed at how casually otherwise serious system developers take this form of cheating. I have had intelligent, educated developers patiently explain to me that yes, they do understand that some small degree of future knowledge took part in their performance simulation. But then they go to great pains to explain how this 'unavoidable' leakage is so tiny that it is insignificant and could not possibly impact their results to any material degree. Little do they know. This is why a recurring focus of this text is methods for avoiding even the tiniest touch of future leak. In my early years of system development, I was often amazed at how subtle this leakage can be.

Just to pound the point home, Figure 1.1 below shows the equity curve of a nearly random Win1 / Lose 1 trading system with just a one percent winning edge. This curve, which would be on average flat if it were truly random (worthless), is quite respectable from just this tiny edge. Future leak is far deadlier than you imagine. Take it seriously.

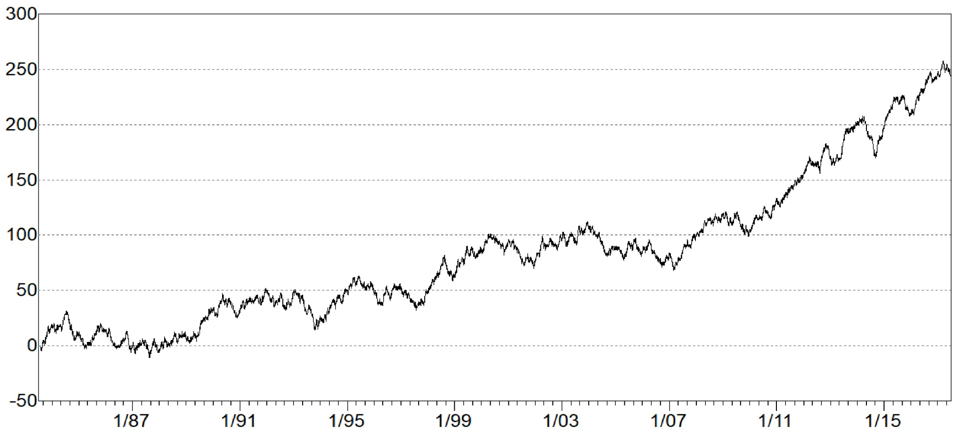

Figure 1.1: Equity curve of random system with 1 percent edge

The Percent Wins Fallacy

There is a simple mathematical formula, essential to trading system development and evaluation, that seems to be difficult for many people to accept on a gut level, even if they understand it intellectually. See Equation (1.1) below.

$$\mathit{ExpectedReturn} = \mathit{Win} * P(\mathit{Win}) - \mathit{Loss} * P(\mathit{Loss}) \tag{1.1}$$

This formula says that the expect return on a trade (the return that we would obtain on average, if this situation were repeated many times) equals the amount we would win times the probability of winning, minus the amount that we would lose times the probability that we will lose.

It's easy to accept that if we flip a fair coin, winning a dollar if we get heads and losing a dollar if we get tails, our expected return is zero; if we were to repeat the coin toss many times, over the long term our average return per coin toss is zero. It's also easy to accept that if the coin is fair and we win two dollars but lose only one dollar, we are in an enviable position.

Now think about trading a market that is a true random walk; among other properties, the changes from one bar to the next are all independent of one another and have zero mean. It is impossible to develop a trading system that has anything other than zero expectation (ignoring transaction costs, of course). But we can easily shift the expected size of wins and losses, as well as their frequencies.

For example, suppose we open a long position and set a profit target 1 point above the entry price, and set a stop loss exit 9 points below the entry. Every time we experience a loss, it will be painfully large, 9 times what we win. But if we execute a large number of such trades on our hypothetical random market, we will find that we win 9 times more often than we lose. We win 9/10 of the time. By Equation (1.1) above, our expected return per trade is still zero. The takeaway here is that win/loss sizes and probabilities are inextricably related. If someone brags about how often their trading system wins, ask them about the size of their wins and losses. And if they brag about how huge their wins are compared to their losses, ask them how often they win. Neither exists in isolation.