Wiley Trading Series

ULTIMATE ALGORITHMIC

ADING SYSTEM OLBOX

Using Today's Technology to Help You Become a Better Trader

WILEY

The Ultimate Algorithmic Trading System Toolbox

+ Website

Using Today's Technology to Help You Become a Better Trader

George Pruitt

WILEY

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Names: Pruitt, George, 1967- author.
Title: The ultimate algorithmic trading system toolbox + website : using today's technology to help you become a better trader/George Pruitt.
Description: Hoboken : Wiley, 2016. | Series: Wiley trading | Includes index.
Identifiers: LCCN 2016010889 (print) | LCCN 2016011196 (ebook) | ISBN 9781119096573 (hardback) | ISBN 9781119262961 (pdf) | ISBN 9781119262978 (epub)
Subjects: LCSH: Electronic trading of securities. | Investment analysis. | Futures. | BISAC: BUSINESS & ECONOMICS/Finance.
Classification: LCC HG4515.95 .P788 2016 (print) | LCC HG4515.95 (ebook) | DDC 332.640285/4678—dc23
LC record available at http://lccn.loc.gov/2016010889

Cover Design: Wiley
Cover Images: © agsandrew/Shutterstock

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

The Ultimate Algorithmic Trading System Toolbox

The Wiley Trading series features books by traders who have survived the market's ever changing temperament and have prospered—some by reinventing systems, others by getting back to basics. Whether a novice trader, professional, or somewhere in between, these books will provide the advice and strategies needed to prosper today and well into the future. For more on this series, visit our website at www.WileyTrading.com.

Founded in 1807, John Wiley & Sons is the oldest independent publishing company in the United States. With offices in North America, Europe, Australia, and Asia, Wiley is globally committed to developing and marketing print and electronic products and services for our customers' professional and personal knowledge and understanding.

ABOUT THE AUTHOR

It was March of 1989 as I drove my beat-up Dodge up Hillside Rd. in Hendersonville, NC. In an attempt to pay for my last semesters of college I was answering a classified ad that was looking to hire a computer programmer. As I drove up the thin drive I passed several houses and then through a gate attached to two large stone pillars. I stopped the car and looked down at the ad again to make sure I was at the right place. I proceeded down the country lane and the view opened up into a large meadow. At the end of the lane was a circular drive and large farm house. As I circled and went back down the road I thought to myself I must have the wrong address or directions. So I followed the small road back down the main highway and then to a small convenient store. Once there I asked myself again what type of business was this Futures Truth and if I should call and get directions or just simply forget about it. Curiosity and the need for money were too much so I used the store's pay phone and called the number once again.

"Hello—Futures Truth, may I help you?" a lady's voice answered.

"Yes, this is George Pruitt and I made an appointment for an interview but I can't seem to find your office."

"Do you drive a red Dodge?" she asked.

"Yes I do. How did you know?"

"We saw you drive right by the office. When you come through the two stone pillars turn immediately to the left. Don't go all the way down the drive—that's the owner's house."

So I follow the directions and find myself in front of a small house. I knock on the door and John Fisher opens and invites me in. We go through the normal Q and A for a job interview and he finally asks if I knew FORTRAN. My first college programming class was FORTRAN so I confidently answered, "Sure!"

He then asked me if I knew anything about the Futures market. I vaguely remembered the term from one of my economics classes and of course from the Eddie Murphy movie and answer him with the question, "You mean like Trading Places with Eddie Murphy?"

John Fisher said "Sort of like that—yes."

He went on to explain how Futures Truth tried to determine market direction in the most widely traded futures contracts by using trading systems. The trading systems were programmed in FORTRAN and they needed help with the programming. In addition to trading they also published a newsletter in which they tracked publicly offered trading systems.

I asked, "Do people really buy these programs?"

John Fisher said yes and by that time an older gentlemen walked into the office and stated that he had spent thousands of dollars on these programs and was ultimately ripped off. John Hill stated this was the main reason he started Futures Truth. He wanted to bring truth to the trading system industry. Both Johns told me that most traders couldn't afford to validate the trading systems because of the cost of the computer equipment, data, and software. John Fisher pointed to the computer he was working on and asked, "How much do you think this Macintosh II cost?"

I answered him, "I am not that familiar with Macs but I know they aren't cheap."

My mouth fell open when he said "$4,000 and we have three of them." Remember this was way back in 1989 when computers were not cheap.

I was thinking to myself that they got ripped off because they could have got a much cheaper and better computer with the IBM PS/2. And what was up with using FORTRAN? Did they not know "C" was the new programming language of the 1990s? John Fisher chose the Apple Macintosh because of its easy-to-use graphical user interface (GUI) and FORTRAN because many traders and hobbyist programmers had knowledge of this language.

John Fisher also said that he and John Hill had developed what they considered the best testing platform, "Excalibur." This platform could load decades of daily and intraday data and test any trading idea that could be defined in an algorithmic form. He also said the only thing that was missing was a charting application and that was where they also needed help.

I explained that I would be wrapping up my degree after summer and both Johns agreed that I could work part time in the evening until I graduated and then we could go from there.

Well that was 27 years ago and I did work part time until I graduated with a degree in computer science from the University of North Carolina at Asheville. The "Excalibur Chart" project turned into my senior project, which blew my professors away. Over the years I have worked with many trading firms in the development of trading algorithms and testing platforms. I have seen it all and have had the great pleasure to be educated by some of the greatest minds in the industry, including John Fisher, John Hill Sr. and John Hill Jr. Even with this experience and education the ultimate trading system still eludes me. As John Hill has stated many times, "A speculator who dies rich, dies before his time!" This may be true, but I have seen traders make millions, lose millions, and make millions again. The one thing they always do when they fail is get right back up, dust themselves off, and start searching for the next great trading algorithm.

INTRODUCTION TO THE ULTIMATE ALGORITHMIC TRADING SYSTEMS TOOLBOX

If you want to learn more about high-frequency trading utilizing special order placement/replacement algorithms such as Predatory trading, Pinging, Point of Presence, or Liquidity Rebates, then this book is not for you. However, if you want to learn about trading algorithms that help make a trading decision, trade size, money management, and the software used to create these algorithms, then you're in the right place.

This book is designed to teach trading algorithm development, testing, and optimization. Another goal is to expose the reader to multiple testing platforms and programming languages. Don't worry if you don't have a background in programming; this book will provide enough instruction to get you started in developing your own trading systems. Source code and instructions will be provided for TradeStation's EasyLanguage, AmiBroker's AFL, and my own Python and Excel testing engines. I chose these platforms because they give a nice overview of different scripting languages and trading platforms. Users of different testing/trading platforms may criticize my decision to use just these platforms, but the EasyLanguage source code that will be provided can be easily ported into Multi-Charts, and AmiBroker's unique and powerful platform provides a complete trading solution. My Python and Excel software, including all source code, are included on the associated website as well as the EasyLanguage and AFL source code for the other platforms. I didn't include the use of Python's scientific libraries, NumPy or SciPy, because I wanted to keep things as simple as possible. Also I used the bare-bones IDLE (Python's own simple Integrated Development Environment) to cut down on the learning curve—I wanted to get to the bare essentials of Python without muddying the water with a sophisticated IDE. Many successful Quants utilize **R**

(a GNU project for statistical computing), but again to keep things simple I stuck with the easy-to-learn Python. The majority, if not all algorithms were tested utilizing commodity and futures data only. All the testing platforms in the book can be used to test stocks and ETFs, and all the included trading algorithms can be applied to these assets as well. Stock and ETF data is very simple to acquire. Getting commodity and futures data in an easily usable format is a little more difficult. Deep histories for commodity and futures can be acquired for as little as $100 from Pinnacle Data. I have used CSI data since the late 1980s and it is the data I used for a good portion of the testing carried out in the book. I would definitely take a look at Pinnacle and CSI data, especially if you wanted your database updated daily. If you are not familiar with Quandl, then you might want to take the time to do so. Quandl is a search engine for numerical data. I was pleasantly surprised to find a free continuous futures database (Wiki Continuous Futures) on Quandl. Keep in mind this data is free and is no way as good as premium data such as CSI and Pinnalce—it is missing multiple days and data points and the continuous data is simply created by concatenating data from individual contracts. The gaps between contracts are included, which cannot be utilized on any testing platform. In real life, a futures position is "rolled-over" from one contract to another by liquidating the front-month position and initiating the same position in the next contract. This "rollover" trade eliminates the gap. I have written a Python application that takes the Wiki Futures data and creates a back-adjusted continuous contract that can be imported into the Python and Excel System Back Tester software. Since I needed the data to do testing, I have also included a 10-plus-year ASCII back-adjusted futures database for 30-plus markets on the companion website. Directions on how to use the software and download futures data from Quandl are included along with the software.

The one thing I really wanted to include in this book was the "Holy Grail" of algorithmic trading systems. I have analyzed many algorithms that claimed to be the Grail, but after rigorous testing they failed to break even. So go ahead and check this off your list. Even though the "Holy Grail" will remain hidden you will find the following:

- Twenty-seven years of experience working with non-programmers in the development of their own trading algorithms
- The tools or building blocks that are used most often in the development cycle
- The core trading models that make up the majority of publicly offered trading systems
- The most important and simplest programming techniques to transform a non-quant into a not-so-non-quant

- Simple examples and explanations of complex trading ideas such as Walk Forward and Genetic Optimization and Monte Carlo simulation
- A complete toolbox to help algorithm development from idea to a finished tradable solution

The majority of successful trading algorithms utilize quantitative analysis (QA). QA is simply the application of mathematical formulae to a financial time series. This book will solely focus on this type of analysis in the design of trading algorithms. Fundamental analysis, which is used in many trading plans, will be used, too, but it will be reduced and simplified into a pure and easily digestible data format. Fundamental data is huge and diverse and in many cases market movement reacts to it in an unpredictable manner. A good example that I have dealt with for many years is the monthly unemployment report. At the time of the writing of this book unemployment has been on a downward trend, which is usually a bullish indicator for the stock market. However, with interest rates at the time being close to 0% the market could react opposite due to the fear of the Federal Reserve doing away with quantitative easing and raising rates. This type of fundamental analysis requires many different inputs and trying to reduce it down to something testable is nearly impossible.

Quantitative analysis focuses on just the data included in a chart. Price action and price translations are easily definable and therefore can be tested. The ability to test and evaluate a trading algorithm is a tremendous tool as it shows how a model can accurately map a market's behavior. If you can interpret a market's behavior, you can take advantage of its inefficiencies. If an algorithm has been capable of exploiting a market's inefficiencies on a historic basis, then there is a possibility it will do so in the future. This hope of future performance is the only leg an algorithmic trader has to stand upon. We all know historic performance is not necessarily an indicator of future results, but what else do we have? An algorithmic trader who quickly defines and tests his system and immediately takes a leap of faith because the historic performance looks great is doomed. Doesn't this contradict what I just said about historical performance being a system trader's only gauge of quality? A good trading algorithm not only demonstrates profitability but also robustness. Robustness is an expression of how well a trading system performs on diverse markets and diverse market conditions. An algorithm can be improved to a point where the trader can feel somewhat confident putting on those first few trades as well as continuing to put trades on after a losing streak. Improving an algorithm is not simply tweaking it until the historic results look utterly fantastic (aka curve fitting); it is taking the time to learn and work with tools that are designed to

make a trading algorithm fail. That's the ultimate objective—making your trading algorithm fail before any money is put on the line. Remember the absence of failure is success and if your algorithm survives the brutal gauntlet of in depth analysis, then you know you might, just *might* have a winner.

This book starts out simple in Chapter 1 with the definition and examples of algorithms. The chapter is a little longwinded but I know that the inability to put a trading idea onto paper and then into pseudocode and finally actual computer code is the biggest stumbling block for traders who want to test their own trading ideas. All trading algorithms that are reducible to a set of instructions can be properly programmed using one of two different modeling methods or paradigms. These two paradigms, Finite State Machine and Flow Chart, are fully discussed and utilized to translate written descriptions first into diagrams and then into actual pseudocode. The diagrammatic approach as well as the simple pseudocode language used to formulate trading algorithms is introduced in this chapter. It doesn't matter how sophisticated your testing software is if you can't define a testable algorithm and this chapter shows you how to do so.

Chapter 2 may be a refresher for those who are familiar with the basic building blocks of trading algorithms, indicators; however, the chapter not only explains the logic behind the indicators but shows how they can be incorporated into complete entry and exit techniques. Diagrams and pseudocode are carried on through this chapter to aid in the understanding of each indicator, its purpose, and its place in a trading algorithm. In addition, the first look at indicator-based trading algorithm performance is presented as well.

Chapter 3 introduces complete trading algorithms and their associated historical performance. Most, if not all, testing was performed on historical commodity/futures data. This data gave rise to the concept of systematic trading more than 50 years ago. Now this doesn't mean the ideas aren't transferable to the stock market. In most cases they are. However, I stuck with commodity data because that is where my expertise lies. The complete pseudocode and actual computer code of these algorithms are revealed as well. The key metrics for determining algorithm robustness are explained and utilized in the evaluation of the algorithms' results.

Chapter 4 starts the section that highlights different testing/trading software platforms that can either be purchased or leased. AmiBroker is introduced in this chapter and the most important components of a trading platform are highlighted: integrated development environment and its associated scripting/programming language, individual market and portfolio testing, and algorithm performance metrics. These components are then highlighted again in Chapter 5 with VBA for Excel, Chapter 6 with Python, and finally Chapter 7 with TradeStation.

Chapter 8 delves into the concepts of Genetic and Walk Forward Optimization, Walk Forward Analysis, and Monte Carlo simulation. A genetic optimizer is built using VBA and used to help explain the ideas of synthesizing computers with

biology. The core concepts of Genetic Algorithms, fitness, selection, reproduction, and mutation are fully explained and illustrated utilizing Excel. Artificial intelligence is here to stay in the study of trading algorithms and this chapter tries to pull back the veil of mystery and show how these tools should be used, and in some cases, must be used to develop that elusive robustness. Along these lines, Machine Learning has become a very highly discussed and somewhat controversial topic in today's trading. Also "Big Data" analysis has found its way to the front as well. These topics are highly advanced and I felt beyond the scope of this book. I can state I have worked with the algorithms that were derived with machine-only input and they have stood the test of time.

A trading algorithm must work over a diverse portfolio of markets before it can be sufficiently considered useful and robust. Chapter 9 utilizes the portfolio-level testing capabilities of TradeStation and AmiBroker to demonstrate different money and portfolio management techniques. The Fixed Fractional approach, by far the most popular, will be highlighted.

The complete source code for the Python System Back Tester is included on the website. Python is the new language of many a quant and the source shows how the language can be used to develop a simple, yet powerful, back tester. Important language concepts and syntax are used to open ASCII files, and import the data into a LIST data structure, create classes and modules, and loop through the entire database while applying a trading algorithm. All the parts of building a testing platform are revealed in the source code, including Monte Carlo and Start Trade Drawdown simulation.

Most traders have Microsoft Excel on their computers and the complete source for a more simplified version of the Python back tester using VBA is included on the website as well.

This book is a toolbox and a guide and touches upon many different facets of algorithmic trading. As with any toolbox it will take time and effort to apply the tools found within to replicate the trader's ideas in a form that not only can be tested and evaluated but fully implemented.

CONTENTS