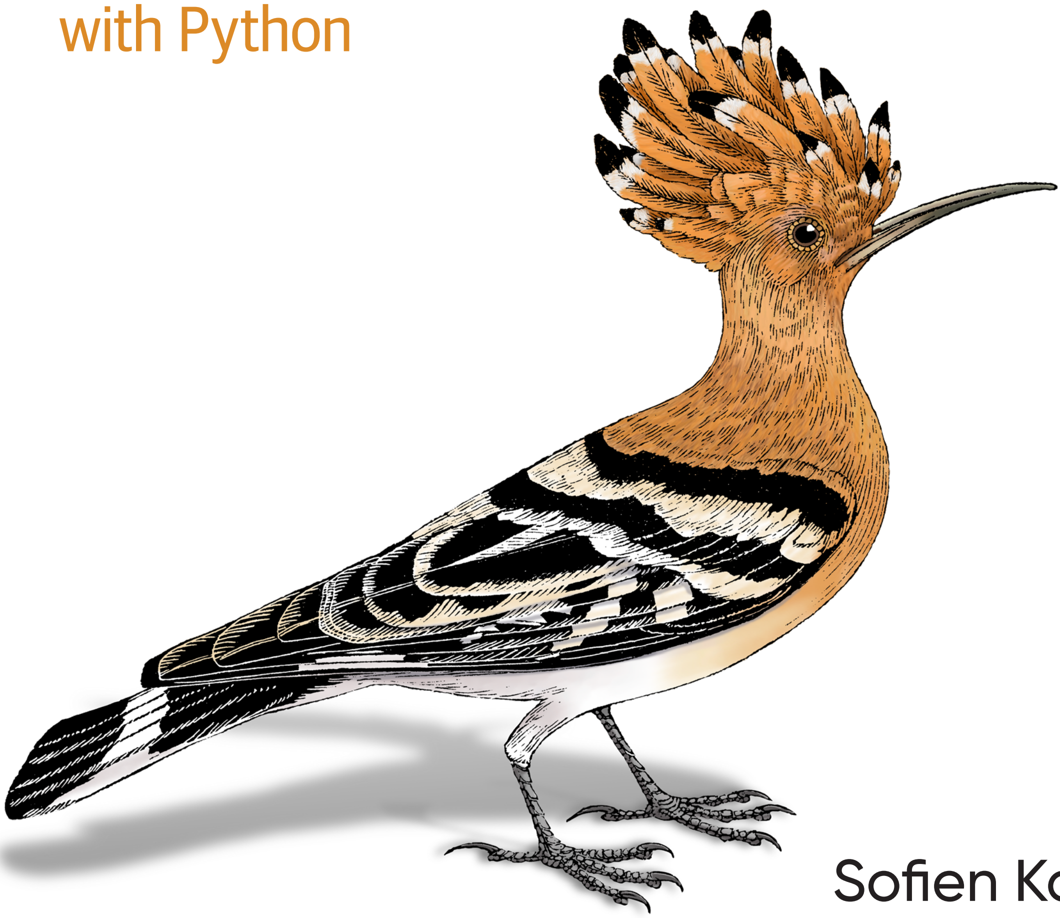
O'REILLY®

Mastering Financial Pattern Recognition

Finding & Back-Testing Candlestick Patterns with Python



Sofien Kaabar

O'REILLY®

Mastering Financial Pattern Recognition

Candlesticks have become a key component of platforms and charting programs for financial trading. With these charts, traders can learn underlying patterns for interpreting price action history and forecasts. This A-Z guide shows portfolio managers, quants, strategists, and analysts how to use Python to recognize, scan, trade, and back-test the profitability of candlestick patterns.

Financial author, trading consultant, and institutional market strategist Sofien Kaabar shows you how to create a candlestick scanner and indicator so you can compare the profitability of these patterns. With this hands-on book, you'll also explore a new type of charting system similar to candlesticks, as well as new patterns that have never been presented before.

With this book, you will:

- **Create and understand the conditions required for classic and modern candlestick patterns**
- **Learn the market psychology behind them**
- **Use a framework to learn how back-testing trading strategies are conducted**
- **Explore different charting systems and understand their limitations**
- **Import OHLC historical FX data in Python in different time frames**
- **Use algorithms to scan for and reproduce patterns**
- **Learn a pattern's potential by evaluating its profitability and predictability**

"Very well written. The book makes complicated stuff an easy read and is a must if you are passionate about trading."

—Saby Upadhyay
CEO, White Swan Global Markets

"Extremely useful for those interested in technical analysis, especially candlestick analysis. Highly recommended for beginners and veterans alike."

—Sattam Al-Sabah
President, OneUp Trader

Sofien Kaabar is a financial author, trading consultant, and institutional market strategist specializing in the currencies market with a focus on technical and quantitative topics. Sofien's goal is to make technical analysis objective by incorporating clear conditions that can be analyzed and created with the use of technical indicators.

DATA SCIENCE

US $69.99 CAN $87.99
ISBN: 978-1-098-12047-4

Praise for *Mastering Financial Pattern Recognition*

Very well written. The book makes complicated stuff an easy read and is a must if you are passionate about trading.

—Saby Upadhyay, CEO, White Swan Global Markets

An extremely useful book for those interested in technical analysis, especially candlestick analysis. Easy to read and follow. I highly recommend this book for beginners and veterans alike.

—Sattam Al-Sabah, president, OneUp Trader

Provides a wealth of information on technical analysis and candlestick charting. Having the Python code as a starting point helps the reader implement and understand things that much faster.

—Timothy M. Kipper, Jr., founder, JDTK Investments

A comprehensive guide for trading using candlestick pattern-based signals, with detailed illustrations of indicators, easily implementable code, and a thorough overview of building your own systematic trading strategy.

—Ning Wang, quantitative investment strategies structurer, Barclays

Mastering Financial Pattern Recognition

Finding and Back-Testing Candlestick Patterns with Python

Sofien Kaabar

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

Mastering Financial Pattern Recognition
by Sofien Kaabar

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: Michelle Smith
Development Editor: Corbin Collins
Production Editor: Elizabeth Faerm
Copyeditor: Kim Wimpsett
Proofreader: Paula L. Fleming
Indexer: Potomac Indexing, LLC
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Kate Dullea

October 2022: First Edition

Revision History for the First Edition
2022-10-18: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781098120474* for release details.

978-1-098-12047-4

[LSI]

Table of Contents

Preface

Finding patterns is the essence of wisdom.

—Dennis Prager

With technological progress and the decentralization of financial information, coding and automated research have become integral parts of the trading world. Anyone who masters the art of trading and coding has a tremendous edge in the markets.

Trading techniques are numerous, and they can be based on many tools and concepts. For example, fundamental traders rely on economic and political analyses to derive a long-term view on different assets, while technical traders rely on more quantitative measures and a few psychological concepts to forecast the next likely moves of the markets.

Therefore, we can say that on a high level there exist two types of analyses, fundamental and technical. This book will present in detail a field in technical analysis called *candlestick pattern recognition*.

Why This Book?

I have spent my career researching trading strategies, patterns, and anything related to the financial world. I have developed a special passion for patterns, and specifically candlestick patterns, due to their widespread adoption but also their interesting performance results. Moreover, throughout the years, I have discovered a few candlestick patterns that I believe can at least rival the classical patterns. This brings us to the purpose of writing the book: I am aiming to present the totality of candlestick patterns, including my personal ones, and how to code a system that back-tests them across a wide variety of markets.

Machines can perform pattern recognition and detection better than humans because of their objectivity. Therefore, I have dedicated the first chapters of the book to creating the structure of a candlestick pattern recognition algorithm before moving on to dig deep into patterns and strategies in the later chapters. This means that the first skill you will learn is how to automate the data import process in Python.

There are many classical candlestick patterns, and it is everyone's duty to test them to see whether they actually are predictive. After all, if we use these patterns to forecast the markets, we should have objective results that prove they are indeed value-adders. We will get such results and interpret them, just as I do with the candlestick patterns that I have discovered over the years. We will also see the advantages and limitations of every pattern.

When we do find the good patterns that help with the predictive task, you should insert them in the overall trading framework, which includes other tools and a risk management system. You will learn how to code technical indicators and combine them with candlestick patterns to create trading signals. Finally, you will back-test these signals, and you will be able to optimize the parameters so that you get a good full-scale pattern recognition strategy.

Hence, the utility of this book is to show you how to automate your research by letting the algorithms you create evaluate the different candlestick patterns. Finally, you will learn how to determine your strategy, which will use the patterns and combine with other technical indicators.

Target Audience

This book is suited to aspiring students, academics, curious minds, and finance practitioners who are interested in candlestick pattern recognition and its applications in finance. You will benefit from this book if you are interested not only in using Python but also in developing strategies and technical indicators.

The book assumes you have basic background knowledge in both Python programming (professional Python users will find the code very straightforward) and financial trading. I take a clear and simple approach that focuses on the key concepts so that you understand the purpose of every idea.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This element signifies a tip or suggestion.

This element signifies a general note.

This element indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *https://github.com/sofienkaabar/mastering-financial-pattern-recognition*.

If you have a technical question or a problem using the code examples, please send email to *bookquestions@oreilly.com*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but generally do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Mastering Financial*

Pattern Recognition by Sofien Kaabar (O'Reilly). Copyright 2023 Sofien Kaabar, 978-1-098-12047-4."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

O'Reilly Online Learning

For more than 40 years, *O'Reilly Media* has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, and our online learning platform. O'Reilly's online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O'Reilly and 200+ other publishers. For more information, visit *https://oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *https://oreil.ly/mstrg-finan-pttrn-recog*.

Email *bookquestions@oreilly.com* to comment or ask technical questions about this book.

For news and information about our books and courses, visit *https://oreilly.com*.

Find us on LinkedIn: *https://linkedin.com/company/oreilly-media*.

Follow us on Twitter: *https://twitter.com/oreillymedia*.

Watch us on YouTube: *https://youtube.com/oreillymedia*.

Acknowledgments

Nothing would be the same without the support of my parents, which is why I can't help but acknowledge their direct and indirect impact on the book. I would also like to mention that none of this would be possible without the patience of my wife, Charline, who tolerated my late-night writing. I have nothing but gratitude toward her.

I would also like to acknowledge the debt I owe to the editors, Michelle Smith and Corbin Collins, as well as production editor Elizabeth Faerm, for their continued support and the amazing job they do and also for their patience. Similarly, I would like to thank every person involved at O'Reilly Media.

Additionally, my special thanks go to the great technical reviewers, Ning Wang, Timothy Kipper, and Kushan Vora, for their immense contributions. They have had a sizable impact on making this book readable, useful, and straightforward. I could not ask for better people to review my book.

Finally, I am deeply grateful to you, the reader, for investing your time into reading my work and for placing your trust in my research. I hope you find it useful.

CHAPTER 1

Importing and Processing Financial Data in Python

This chapter is dedicated to laying the foundation needed to analyze financial data through coding. This requires some preparation, such as downloading the right software and creating an algorithm that fetches historical data automatically.

By the end of the chapter, you should know how to automatically import historical financial data using Python, a skill that should save you time. So let's get started.

Installing the Environment

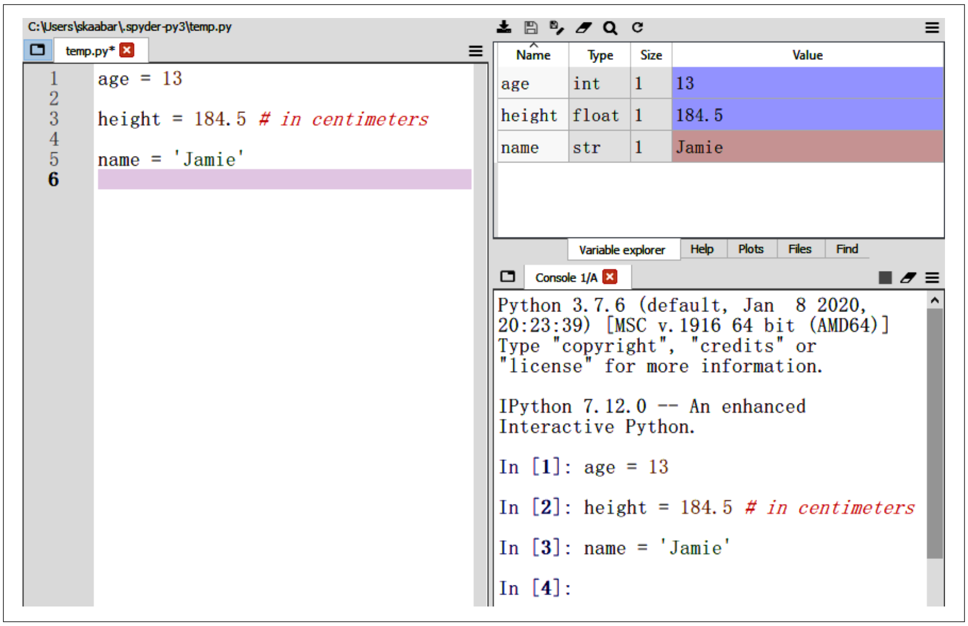
The first step is to prepare the environment and everything else necessary for the success of the algorithms. For this, you need two programs:

- A Python interpreter that you use to write and execute code
- Charting and financial software that you use as a database

Let's start with the Python interpreter. I use a software called SPYDER. Some people may be more familiar with other software such as Jupyter and PyCharm, but the process is the same. You can download SPYDER from the official website or, even better, download it as part of a bigger package called Anaconda, which facilitates installation and offers more tools. Note that it is open source, free-to-use software.

SPYDER's interface is split into three windows, as you can see in Figure 1-1. The window on the left is used to write the code that is later executed (the algorithm is told to run and apply the code). Typically, you will see multiple lines of code in that area.

The window on the upper right is the variable explorer. Every time a variable is stored, you can see it there. The window on the lower right is the console that shows the result of the code, whether it is an error or an output.



```
age = 13

height = 184.5 # in centimeters

name = 'Jamie'

```

Name	Type	Size	Value
age	int	1	13
height	float	1	184.5
name	str	1	Jamie

```
Python 3.7.6 (default, Jan  8 2020,
20:23:39) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or
"license" for more information.

IPython 7.12.0 -- An enhanced
Interactive Python.

In [1]: age = 13

In [2]: height = 184.5 # in centimeters

In [3]: name = 'Jamie'

In [4]:
```

Figure 1-1. SPYDER's interface

The types of data that you can define and use in the code are classified into several categories:

Integers
: These are whole numbers, which can be either positive or negative. Examples are −8 and 745. They are, however, limited to between −2147483648 and 2147483647. Any number falling outside this range is considered a different data type called a *long*. The difference between data types has to do with memory. Integers are 32 bits in width, whereas longs are 64 bits in width.

Floats
: These are real numbers with decimal points such as 18.54 and 311.52.

Strings
: These are words stored in a variable. More scientifically, they are a set of structured characters (text). In Python, you write strings between single or double quotes.

In line 1 of the code in Figure 1-1, I have defined a variable called `age` and set it to 13. When you run this code, you should see the creation of `age` in the variable explorer with type `int` (integer) and a value of 13. In line 3, I have executed the code that defines the `height` variable set to 184.50 (therefore, a float data type).

Notice that next to the definition of the variable, I have written the phrase `in centimeters`, preceded by a hashmark. This is called a *comment*. Comments are very important in Python for explaining the code. Therefore, anything preceded by `#` will not be executed. In the variable explorer, you can see the `height` variable with the `float` type. Line 5 defines a string, which in the variable explorer is shown as `str` type (string). In the console, you can see that the code has been successfully executed because there are no errors, which would be shown in red.

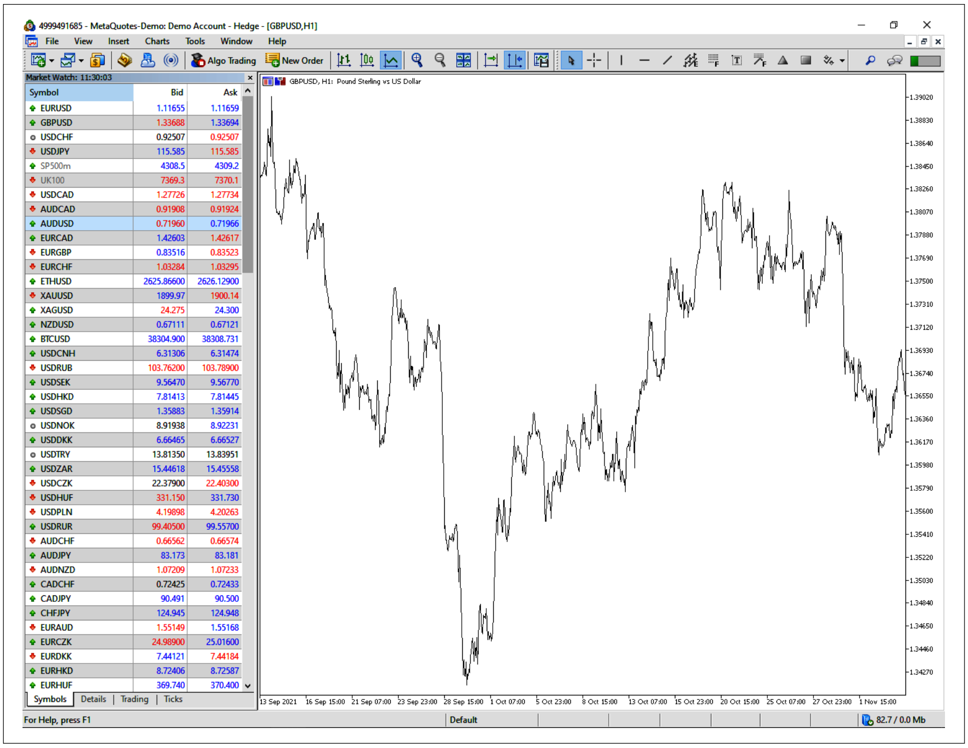
The next step in preparing the environment is to install the charting software that allows you to import historical data into SPYDER. Throughout the book, I use MetaTrader 5, a benchmark charting program used by many traders around the globe. Follow these steps:

1. Download SPYDER and familiarize yourself with how it works.
2. Download the MetaTrader 5 software.
3. Use SPYDER to import historical prices from MetaTrader 5.

From the official website, download and install MetaTrader 5. You need to create a demo account, which is simply a virtual account with imaginary money. The word *demo* does not refer to a limited duration of use but to the fact that it is not using real money.

To open an account, select File > Open an Account, choose MetaQuotes Software Corp, and then click Next. Next, choose the first option to open a demo account; this will let you trade virtual money. Finally, enter some basic information such as name, email, and account type. You will not receive a verification request or any type of confirmation as the demo should launch directly, allowing you to see the charts.

Figure 1-2 shows the platform's interface. By default, MetaTrader 5 does not show all the markets it covers, so you need to make them accessible for import and visualization if necessary. Click View, click Market Watch, and then right-click any of the symbols shown in the new tab and choose Show All. This way you can see the extended list with more markets.



4999491685 - MetaQuotes-Demo: Demo Account - Hedge - [GBPUSD,H1]

Market Watch: 11:30:03

Symbol	Bid	Ask
EURUSD	1.11655	1.11659
GBPUSD	1.33688	1.33694
USDCHF	0.92507	0.92507
USDJPY	115.585	115.585
SP500m	4308.5	4309.2
UK100	7369.3	7370.1
USDCAD	1.27726	1.27734
AUDCAD	0.91908	0.91924
AUDUSD	0.71960	0.71966
EURCAD	1.42603	1.42617
EURGBP	0.83516	0.83523
EURCHF	1.03284	1.03295
ETHUSD	2625.86600	2626.12900
XAUUSD	1899.97	1900.14
XAGUSD	24.275	24.300
NZDUSD	0.67111	0.67121
BTCUSD	38304.900	38308.731
USDCNH	6.31306	6.31474
USDRUB	103.76200	103.78900
USDSEK	9.56470	9.56770
USDHKD	7.81413	7.81445
USDSGD	1.35883	1.35914
USDNOK	8.91938	8.92231
USDDKK	6.66465	6.66527
USDTRY	13.81350	13.83951
USDZAR	15.44618	15.45558
USDCZK	22.37900	22.40300
USDHUF	331.150	331.730
USDPLN	4.19898	4.20263
USDRUR	99.40500	99.55700
AUDCHF	0.66562	0.66574
AUDJPY	83.173	83.181
AUDNZD	1.07209	1.07233
CADCHF	0.72425	0.72433
CADJPY	90.491	90.500
CHFJPY	124.945	124.948
EURAUD	1.55149	1.55168
EURCZK	24.98900	25.01600
EURDKK	7.44121	7.44184
EURHKD	8.72406	8.72587
EURHUF	369.740	370.400

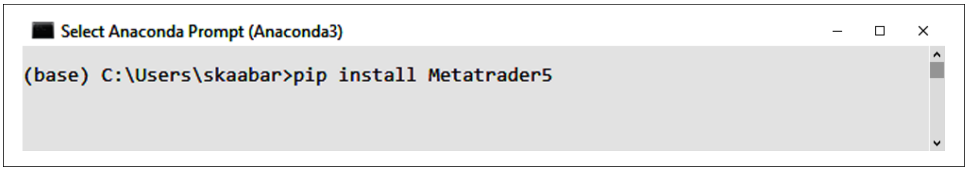
For Help, press F1

Figure 1-2. MetaTrader 5's interface

Creating the Importing Algorithm

Being able to automatically summon historical data of any time frame is a wonderful time-saver as it allows you to focus on research and analysis instead of wasting valuable time acquiring and cleaning the data. Let's create a set of functions that import the historical data of a selected asset almost instantaneously.

Before proceeding to the coding part, you need to install the MetaTrader 5 Python integration library so you can use it later in SPYDER. This is easy and requires one step. Open the Anaconda prompt and type in `pip install Metatrader5`, as shown in Figure 1-3.



```
(base) C:\Users\skaabar>pip install Metatrader5
```

Figure 1-3. Anaconda prompt showing the command to install the MetaTrader 5 library

Installation is the bridge that allows you to use Python libraries designed for MetaTrader 5 in the interpreter.

The following code block uses the `import` built-in statement, which calls for internal (self-created) or external (created by third parties) libraries. A library is a store of functions, and thus, you need to import the libraries that are pertinent to what you want to do. For demonstration purposes, import the following modules, packages, and libraries:

```python
import datetime
import pytz
import pandas as pd
import MetaTrader5 as mt5
import numpy as np
```

The `datetime` module gives tools for manipulating the dates and times, the `pytz` library offers cross-platform time zone calculations that are needed for the import, and the `pandas` and `numpy` libraries are used for data manipulation and analysis.

You mainly use `numpy` for most calculations and data manipulation.

The MetaTrader 5 library imports the functions relating to the software's module and is the key library that will allow you to import the financial historical data.

Note that the three last lines of code contain the `as` statement. This is used to give a custom name to the library when you want to use it frequently and save writing space. In other words, Python recognizes the MetaTrader 5 library as mt5 from now on.

Modules are files that contain functions and variables. A *package* is a collection of modules; it needs to have an *init.py* file. A library is simply a collection of packages.

Executing the `import` statements means that Python now recognizes the functions inside them and will allow you to use them in future code if you decide to call them. You must run them every time you open a new session, which is why the `import` statements are usually found at the beginning of the code.

The next step is to create the universe of the time frames that you will be able to import. Even though I will be showing you how to analyze and back-test hourly data, you can define a wider universe, as shown in the following code snippet:

```
frame_M15 = mt5.TIMEFRAME_M15 # 15-minute time
frameframe_M30 = mt5.TIMEFRAME_M30 # 30-minute time frame
frame_H1 = mt5.TIMEFRAME_H1 # Hourly time frame
frame_H4 = mt5.TIMEFRAME_H4 # 4-hour time frame
frame_D1 = mt5.TIMEFRAME_D1 # Daily time frame
frame_W1 = mt5.TIMEFRAME_W1 # Weekly time frame
frame_M1 = mt5.TIMEFRAME_MN1 # Monthly time frame
```

A *time frame* is the frequency with which you record the prices. With hourly data, you will record the last price printed every hour. This means that in a day, you can have up to 24 hourly prices. This allows you to see the intraday evolution of the price. However, the close price is just one of the things that you want to import. During a time period (whether hourly or daily), you will see the following:

- The first price of the time period, which is called the *open price*.
- The highest price printed during the time period, which is called the *high price*.
- The lowest price printed during the time period, which is called the *low price*.
- The last price seen before starting a new time period, which is referred to as the *close price*.

Altogether these are called the OHLC[1] data, which are generally ordered as written.

The following code defines the current time, which is used so that the algorithm has a reference point when importing the data. Basically, you are creating a variable that stores the current time and date:

```
now = datetime.datetime.now()
```

Let's now proceed to defining the universe of the assets you want to back-test. I do a mix of four asset classes: currencies, cryptocurrencies, commodities, and equity indices:

- *Currencies* (also known as *forex*, an abbreviation for foreign exchange market) form the biggest financial market in terms of daily volume. Currencies are quoted in pairs, meaning that you cannot just buy USD in absolute terms; you have to buy it using another currency. Therefore, the EURUSD pair refers to the price of 1 EUR in terms of USD. The back-testing universe comprises EURUSD, USDCHF, GBPUSD, and USDCAD.

1 An abbreviation for *open*, *high*, *low*, *and close*.

USD is an abbreviation for the United States dollar, EUR is an abbreviation for the euro currency, CHF is an abbreviation for the Swiss franc, GBP is an abbreviation for Great Britain's pound, and CAD is an abbreviation for the Canadian dollar.

- *Cryptocurrencies* (also known as *cryptos*) are a new, disruptive asset class characterized by severe volatility. The most well-known cryptocurrency is Bitcoin, followed by Ethereum. Note that both are expressed in terms of USD; this is why they are labeled as BTCUSD and ETHUSD.

Notice that Bitcoin (BTC) and Ethereum (ETH) are quoted versus USD. They are generally considered the most liquid cryptocurrency pairs.

- *Commodities* are physical assets such as gold, silver, and copper. They are divided into many categories such as energy (crude oil, Brent oil, etc.) and industrial metals (copper, zinc, etc.). In the universe of assets, I stick to gold and silver.
- *Equity indices* are weighted calculations of a select basket of a country's stocks. They are used to analyze the overall stock market health of a nation. In this book, I cover the S&P 500, a proxy for US stocks, and the FTSE 100, a proxy for UK stocks:

```python
assets = ['EURUSD', 'USDCHF', 'GBPUSD', 'USDCAD', 'BTCUSD',
          'ETHUSD', 'XAUUSD', 'XAGUSD', 'SP500m', 'UK100']
```

Now that you have your time and asset variables ready, all you need is to create the structure of the importing algorithm. The `get_quotes()` function does this:

```python
def get_quotes(time_frame, year = 2005, month = 1, day = 1,
               asset = "EURUSD"):

    if not mt5.initialize():

        print("initialize() failed, error code =", mt5.last_error())

        quit()

    timezone = pytz.timezone("Europe/Paris")

    time_from = datetime.datetime(year, month, day, tzinfo = timezone)

    time_to = datetime.datetime.now(timezone) + datetime.timedelta(days=1)

    rates = mt5.copy_rates_range(asset, time_frame, time_from, time_to)
    rates_frame = pd.DataFrame(rates)

    return rates_frame
```

Notice that in the `get_quotes()` function, you finally use the `pytz` and `pandas` libraries. The function starts by defining the Olson time zone, which you can set yourself. Here is a brief, nonexhaustive list of what you can enter depending on your time zone:

```
America/New_York
Europe/London
Europe/Paris
Asia/Tokyo
Australia/Sydney
```

Afterward, I define two variables called `time_from` and `time_to`:

- The `time_from` variable contains the datetime referring to the beginning of the import date (e.g., 01-01-2020).
- The `time_to` variable contains the datetime referring to the end of the import date (e.g., 12-31-2020).

The next step is to create a variable that imports the financial data using the time periods you have specified. This is done through the `rates` variable using the `mt5.copy_rates_range()` function. Finally, using `pandas`, transform the data into a data frame.

Throughout the book you will be dealing with arrays and not data frames; however, the function `get_quotes()` first imports the values as a data frame due to compatibility, and then you will transform it into an array. At any rate, the main difference between a data frame and an array is the type of data you can keep inside and the structure of the axes.

The final function required for the importing process is the `mass_import()` function. It lets you choose the time frame using the variable and then uses the `get_quotes()` function to import the data and format it to an array. The following code snippet defines the `mass_import()` function:

```
def mass_import(asset, time_frame):

    if time_frame == 'H1':
        data = get_quotes(frame_H1, 2013, 1, 1, asset = assets[asset])
        data = data.iloc[:, 1:5].values
        data = data.round(decimals = 5)
    if time_frame == 'D1':
        data = get_quotes(frame_D1, 2000, 1, 1, asset = assets[asset])
        data = data.iloc[:, 1:5].values
        data = data.round(decimals = 5)

    return data
```

The `mass_import()` function automatically converts the data frame into an array, so you do not have to worry about conversion when using the automatic import.

> The algorithm imports a number of historical data limited by MetaTrader 5. Although that number is high, in time you may need to adjust the year argument higher in order to get the data. For instance, if you get an empty array using the `mass_import()` function, try putting a more recent year in the `get_quotes()` function ("2014" instead of "2013," as shown in the preceding example).

Even though there is a MAC version of MetaTrader 5, the Python library only works on Windows. It requires an emulator on a Mac. For Mac users, you may also try the manual import method shown later in the chapter.

Putting It All Together

Let's now see a full example of data import. Remember that the full importing code can be found in the GitHub repository of the book. Normally I am dealing only with hourly data in the book because traders use it heavily, which creates interesting signals; however, let's try applying a couple of examples of importing after having defined the functions seen in this chapter:

- To import the daily ETHUSD data, type the following code:

  ```
  my_data = mass_import(5, 'D1')
  ```

- To import the hourly GBPUSD data, type the following code:

  ```
  my_data = mass_import(2, 'H1')
  ```

There are many ways to import data into Python; some are automatic, and some are manual. You have just seen the first way of using code to communicate with a charting platform and to download the data. The manual way is to have an Excel file with OHLC data that you have downloaded from a third party. In this case, you can use the `pandas` library to import it and transform it into an array.

Suppose that the Excel filename is *my_data* and the file is stored on your desktop. You have to make sure that the SPYDER directory is in the same place as the file. In layperson's terms, SPYDER must search the desktop for the Excel file. To choose the right directory, you must click the folder button next to the arrow, as shown in Figure 1-4.

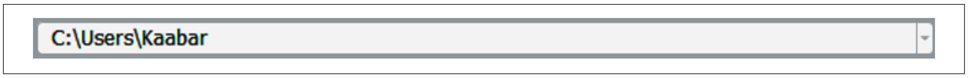

C:\Users\Kaabar

Figure 1-4. Directory tab

You should get a separate window where you can choose the desktop location and then validate the choice. Having done this, the tab should look like Figure 1-5.

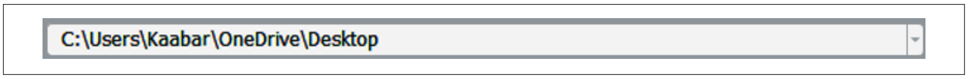

C:\Users\Kaabar\OneDrive\Desktop

Figure 1-5. Directory tab

You must use the `read_excel()` function to get the values inside the Excel file. Follow this syntax:

```python
# Importing the excel file into the Python interpreter
my_data = pd.read_excel('my_data.xlsx')
```

Right about now, you have a data frame called `my_data` with four different columns representing open, high, low, and close prices. You generally have to enter the library's name before using a function that belongs to it; this is why `read_excel()` is preceded by `pd`.

Remember that `pd` is the shortcut used to refer to `pandas`. In parallel, `np` is the shortcut used to refer to `numpy`.

The following syntax shows how to convert structured elements from a data frame to an array to facilitate manipulation. The array library I use is called `numpy`, which is the main library used in this book.

I recommend using the automatic way for Windows users and the manual way for macOS users due to compatibility issues.

In your opinion, what should you do before using the functions of `numpy`? If your answer is import the library, then you are correct. The following code snippet imports `numpy` and converts `my_data` into an array so that it is ready for analysis:

```python
# Importing the library
import numpy as np

# Converting from data frame to array
my_data = np.array(my_data)
```

Alternatively, you can do all of this in one line by just adding `.values` to `pd.read_excel('my_data.xlsx')`, thus becoming `pd.read_excel('my_data.xlsx').values` and resulting in an array instead of a data frame.

Summary

The research and trading framework is composed of four different algorithms that are discussed in detail in the next chapter. They can be summarized as follows:

Import algorithm
: This is the algorithm shown in this chapter that deals with importing and preparing the historical OHLC data to be analyzed or back-tested. I believe that at this stage you can easily do this automatically and manually.

Signal algorithm
: This algorithm, which you will see in Chapter 2, will be responsible for generating the buy and sell orders. In essence, it is the set of conditions that gives the green light that a pattern has appeared and a trade can take place.

Charting algorithm
: This is the simplest algorithm. You will use it to chart the signals on the price chart. Its purpose is to visually analyze the buy and sell signals on the chart. You will also learn about this in Chapter 2.

Performance algorithm
: This algorithm is used to calculate and analyze the results acquired from the signal algorithm. Basically, it calculates performance metrics on the signals generated following the patterns' conditions. You will also learn about this in Chapter 2.

It is important that you know how to automatically import and prepare financial data that you will later analyze. Python offers strong and rapid tools to do so; hence, make sure you master this technique so that you supersize your research capabilities.

CHAPTER 2

Algorithmic Mindset and Functions

An *algorithm* is a set of rules that a computer applies given the realization of certain conditions. You generally use an algorithm to solve a specific problem or to simply follow a repetitive sequence of tasks. You can also use algorithms to find patterns by scanning for the conditions that you set.

The main purpose of this book is to show you how to scan, find, and evaluate candlestick patterns and strategies. The main benefits of using an algorithm as opposed to manually performing the tasks are as follows:

Speed
: Algorithms can run at extreme speed compared to humans. While a simple algorithm can scan hundreds of thousands of data pieces in a few seconds, a human may spend weeks and months doing the same task.

Discipline
: Algorithms follow a distinct set of rules and do not have feelings or emotions that make them ignore the rules occasionally. In addition, algorithms do not fall into the trap of subjective interpretation. This is important for the evaluation process as you need objective and clear measures to judge your trading system.

Percentage of error
: Algorithms are generally error-free when there are no bugs in the code. Humans can make a lot of mistakes due to inattention and fatigue.

A *trading system* is composed of multiple algorithms such as trading and risk management algorithms. A robust trading system relies on clear and stable rules-based algorithms to give reliable measures.

This chapter is divided into four sections. First it covers primal functions, which deal with the basic data manipulation that I use throughout the book. Then it shows how to code the signals of the patterns and strategies. It then moves on to visualizing the signals on the price charts so that you have an aesthetic and interpretable representation. Finally, you'll learn the key performance evaluation metrics and how to code them. Make sure you master the concepts of this chapter as they are crucial for the rest of the book; they are helpful not only in detecting the patterns but also in creating strategies.

Coding the Primal Functions

The *primal functions* are of a small collection of custom functions that help you better manipulate data arrays. The idea of primal functions began as an exercise for me to practice coding in my early years, but over time, they became regular code snippets that I use all the time in my research.

Analyzing and back-testing are repetitive tasks that require some functions to work everywhere. Let's start with the most basic primal functions (remember, you will be using `numpy` throughout the book).

The Function to Add Columns to an Array

Occasionally, you need to add columns to populate them with either indicators or signals. For example, assume that you have an OHLC array composed of four columns. You can use a function to add an extra column that you may use to harbor the following:

- An indicator calculated from the close price, such as a simple moving average[1]
- A proxy for buy and sell signals using predetermined binary values (such as 1 for buy signals and –1 for sell signals)

The rows in the arrays represent time steps. I use the hourly time frame in the book, which means that every row contains the hourly OHLC values and any indicator value or a buy and sell proxy.

1 A *moving average* is a mean calculated on a rolling window. It is typically used to understand the trend of the market. Moving averages are discussed in greater depth in Chapter 3.

The first primal function is therefore `add_column()`, as shown in the following code snippet:

```
def add_column(data, times):

    for i in range(1, times + 1):

        new = np.zeros((len(data), 1), dtype = float)

        data = np.append(data, new, axis = 1)

    return data
```

The function loops a predetermined number of times chosen by the variable `times`, and for every loop, a new array is created containing zero values and having the same length as the original data, as shown by `len(data)`. This new array is born out of the `np.zeros()` prebuilt `numpy` function.

The final step is to take this freshly created array and paste it right next to the original four columns using `np.append()`, thus giving an array with five columns.

As the loop continues, the number of columns increases. The argument `axis` specifies a binary choice between rows and columns. When it is equal to 0, it refers to rows, and when it is equal to 1, it refers to columns. Remember that the variable `times` is the number of columns that you want to add, and this is specified when calling the function. Let's look at an example to make things clearer.

You have an array called `my_data` consisting of four columns, and you want to add five new columns to update your original array so that it has nine columns. What would you write to accomplish this assuming you have already defined the `add_column()` function? The answer is as follows:

```
# Adding five columns to an existing array
my_data = add_column(my_data, 5)
```

The Function to Delete Columns from an Array

Extra columns may occur when trying to calculate complex indicators because you may have to populate them by intermediary calculations that are not needed after the indicator is ready. An example of this would to be to use a column to calculate weights and to use the next column to calculate a weight-based indicator. You would retain the indicator column but not the weight column.

At the end, you want to retain the final result of the indicator and remove the previous columns so that you have a clean array composed of OHLC data with the indicator's readings in the next column.

A quick and easy way to do this is the `delete_column()` function. The following code snippet shows how to define it:

```
def delete_column(data, index, times):

    for i in range(1, times + 1):

        data = np.delete(data, index, axis = 1)

    return data
```

Remember that to use a function, you must define it first. This means that after writing the syntax of the function, you must execute it so that Python stores it in its memory.

The function loops around the specified range, which is the number of columns you want to delete as of a selected index. For example, the function deletes the three columns as of column number 4, which is included.

The function states that the newly transformed array has a column deleted using the built-in `numpy` function `np.delete()` starting from the variable `index`. Finally, the variable `times` is the number of columns to delete, and this is specified when calling the function. Here are two examples to make things clearer:

- You have an array consisting of 10 columns, and you want to erase 4 of them starting from the column indexed at 4. Here is how to do so:

  ```
  my_data = delete_column(my_data, 4, 4)
  ```

Remember that a column indexed at 4 is not the fourth column but the fifth. This is because Python starts indexing at zero.

- You have an array consisting of eight columns, and you want to erase two of them starting from the column indexed at 1. Here is how to do so:

  ```
  my_data = delete_column(my_data, 1, 2)
  ```

You would use `delete_column()` mostly with technical indicators as opposed to patterns, as the latter are direct rules instead of sequential complex calculations like some indicators.

The Function to Add Rows to an Array

Sometimes you want to add rows to the end of an array due to lag or the addition of manual values from the coder. You can add empty rows to the end of an array using the following function:

```
def add_row(data, times):
    
    for i in range(1, times + 1):
        
        columns = np.shape(data)[1]
        
        new = np.zeros((1, columns), dtype = float)
        
        data = np.append(data, new, axis = 0)
        
    return data
```

The function loops around the data, and in each loop it finds the shape of the array using the `np.shape()` built-in function, which gives out the number or rows and columns in a matrix. Since you are interested only in the number of columns, you would add `[1]` to tell the algorithm that you want only the second value, which is the number of columns. Having stored the number in the variable `columns`, the algorithm proceeds to the next line and creates a whole new array made up of zeros with only one row and a column number equal to the variable `columns`. Finally, the algorithm appends this freshly created array with one row to the end of the original array, thus giving out a new row with the same size as the other rows.

The Function to Remove Rows from an Array

When calculating indicators, some calculations may require a minimum number of data in the past; otherwise, you would see invalid values, which appear as NaN[2] in Python. The following function deletes rows from the start:

```
def delete_row(data, number):
    
    data = data[number:, ]
    
    return data
```

This simple function states that the data actually starts from the `number` index and continues to the end of the data. Basically, it ignores a select number of rows from the beginning.

2 An abbreviation for Not a Number. The cell that has NaN is considered invalid and will yield only other NaNs if used in calculations.

Basic Array Syntax or Array Manipulation

Referring to data in Python requires practice. Here is a quick summary to use as rules of thumb:

```
# An array has the following structure: array[row, column]

# Referring to the whole my_data array
my_data
# Referring to the first 100 rows inside the array
my_data[:100, ]
# Referring to the first 100 rows of the first two columns
my_data[:100, 0:2]
# Referring to all the rows of the seventh column
my_data[:, 6]
# Referring to the last 500 rows of the array
my_data[-500:, ]
# Referring to the first row of the first column
my_data[0, 0]
# Referring to the last row of the last column
my_data[-1, -1]
```

The Function to Round Numbers

Why discuss such a simple concept here that does not seem to be important enough to be a function on its own? The answer is that there are certain patterns that are extremely rare unless you round the numbers of the OHLC data.

Let's look at an example of the currency pair EURUSD. In recent years, for accuracy reasons, retail market dealers started quoting most currency pairs in five decimals instead of four. While traders were used to the concept of a *pip*, which is the fourth decimal (i.e., the 4 in 1.0964), they now have a *pipette,* which is the fifth decimal (the 5 in 1.09645).

The change can be confusing for market participants who have been used to the conventional four-decimals system, but it is necessary and favors a tighter spread quoted by dealers. The *spread* is the difference between the ask (buy) and bid (sell) prices. It is the compensation of the market dealer for the risk taken. It is considered a transaction cost to the trader because it entails buying slightly more expensive and selling slightly cheaper.

Let's take this opportunity to discuss how currency pairs work and then build up to the need to round the numbers eventually.

A currency pair is made up of two currencies; the one on the left is the base currency, and the one on its right is the price currency. When you see the pair EURUSD quoting at 1.0500, it means that to buy 1 EUR, you need to spend 1.0500 USD.

Now, let's assume that you bought 10,000 EUR for 10,500 USD and several weeks later find that EURUSD is quoting at 1.0750. What does this mean exactly? It means that the value of EUR has gone up relative to the value of USD because now you need 10,750 USD to buy 10,000 EUR; therefore, theoretically, you have made 250 USD (which you can also refer to as 250 pips gain, as the value per 1 EUR has gone up by 0.0250). This is realized only if you exchange your EUR back to USD, which gives you back 10,750 USD instead of 10,500 USD. With the development of markets and the introduction of the five-decimals system, you may now see EURUSD with five numbers after the decimals (e.g., EURUSD at 1.07516).

There are certain candlestick patterns that require a close price similar to the open price. This is why I include rounding in some of the patterns presented in subsequent chapters. The optimal rounding number for currency pairs must be four decimals—no more, no less. The following code block shows the rounding function:

```
def rounding(data, how_far):

    data = data.round(decimals = how_far)

    return data
```

The function uses a built-in function called `round()` to simplify the process. Make sure to understand the necessity of rounding the OHLC data when searching for specific patterns as you will need it in subsequent chapters.

Before moving on, let's briefly practice the new functions you have learned in this section. Suppose you have an OHLC array and you want to:

- Add two columns to the array:

  ```
  my_data = add_column(my_data, 2)
  ```

- Delete three columns starting from the second column:

  ```
  my_data = delete_column(my_data, 1, 3)
  ```

- Add 11 rows to the end of the array:

  ```
  my_data = add_row(my_data, 11)
  ```

- Delete the first four rows in the array:

  ```
  my_data = delete_row(my_data, 4)
  ```

- Round all the values in the array to four decimals:

  ```
  my_data = rounding(my_data, 4)
  ```

Remember that the variable `data` is generally used in the functions and refers to any data you want to refer to, whereas the variable `my_data` is my default name of all the OHLC arrays I import.

Coding Signals

This marks the beginning of the signal algorithm. Remember, the aim of this section is to see how to create a set of conditions in order to find valid signals. Up to this point, you have imported an OHLC array of an asset or a currency pair.

As I have not yet started the discussion of any candlestick patterns, I will create a hypothetical pattern and then code its conditions and create the signal algorithm. Let's call this hypothetical configuration the Alpha pattern:

- A long (buy) signal is generated on the next open whenever the current low is lower than the low price 5 periods ago and the low price 13 periods ago but higher than the low price 21 periods ago. Simultaneously, the close price of the current bar must be higher than the close price 3 periods ago.
- A short (sell) signal is generated on the next open whenever the current high price is higher than the high price 5 periods ago and the high price 13 periods ago but lower than the high price 21 periods ago. Simultaneously, the close price of the current bar must be lower than the close price 3 periods ago.

A *short sell* position is a complex action where you profit from a price decrease. In layperson's terms, you borrow an asset from a third party and then sell it to a buyer. Finally, you buy it back and return it to the original owner. If the price has decreased, you would pocket the difference; otherwise, you would buy it back at a higher price than when you sold it.

The previous bullet points give everything you need to generate buy and sell signals, and hence, you can start coding the `signal()` function.

Think of the `signal()` function as the one that scans each row and leaves a trace if all the conditions you set are made. These traces are buy and sell proxy orders.

The first thing you need to tell the algorithm is to have a sufficient number of columns that you want to populate with buy and sell orders. As previously seen, you can do this with `add_column()`. The next step is to tell the algorithm to loop around the totality of data, and this is done using the `for` statement, which performs a finite loop between a predefined range, which in your case is the length of the OHLC data array.

As you are initiating the position on the next open price after validating the pattern at the close price, you can use the `try()` and `except()` functions, which simply bypass errors caused by having a signal at the last data of the array that gives a trigger on the next nonexistent row. To illustrate this error, imagine having 500 rows and getting a buy signal on the 500th row. Since you are buying on the next open, the signal would appear on the next row, which does not exist. This would cause an index error.

By now, you are ready to code the Alpha pattern's conditions. I generally code the bullish conditions first. Therefore, after telling the algorithm to add two columns and to loop across the data while ignoring the `IndexError()` issue, I will simply add the conditions using the `if` statement.

The conditions are intuitive: all you need to know is that the variable `i` refers to the current row index in the ongoing loop and therefore `i-5` refers to the variable five rows (time steps) before the current one. The `i` variable is used with the `for` loop so that the conditions can be applied to every row.

Indexing Refresher

Remember that with Python indexing columns and rows at zero, the following rules of thumb must be memorized for any OHLC data I use in the book:

- The index 0 refers to the open price column.
- The index 1 refers to the high price column.
- The index 2 refers to the low price column.
- The index 3 refers to the close price column.

Additionally, you may have the following if they are not preceded by another calculation (indicator):

- The index 4 refers to the buy signals column.
- The index 5 refers to the sell signals column.

However, sometimes you may need to calculate indicators or volatility measures after the OHLC data, which pushes the buy and sell columns to the next indices. This is covered in "Indicator Analysis" on page 52. The following code snippet defines the function of the Alpha pattern:

```
def signal(data):

    data = add_column(data, 5)

    for i in range(len(data)):
       try:

           # Bullish Alpha
           if data[i, 2] < data[i - 5, 2] and data[i, 2] < data[i - 13, 2]
              and data[i, 2] > data[i - 21, 2] and \
              data[i, 3] > data[i - 1, 3] and data[i, 4] == 0:

                    data[i + 1, 4] = 1

           # Bearish Alpha
           elif data[i, 1] > data[i - 5, 1] and data[i, 1] > data[i - 13,
                1] and data[i, 1] < data[i - 21, 1] and \
                data[i, 3] < data[i - 1, 3] and data[i, 5] == 0:

                    data[i + 1, 5] = -1

       except IndexError:

            pass

    return data
```

The statement `data[i, 4] == 0` is the condition that to have a buy signal, you must not have had a buy signal in the previous row.[3] This is to avoid successive signals in case the pattern is repetitive in nature.

Writing if Statements the Right Way

With conditions, the equal sign is doubled; otherwise, you get a syntax error. See the following example:

```
# Wrong syntax
if x = 0:

    y = 1

# Correct syntax
if x == 0:

    y = 1
```

The proxy for a long signal is the number 1 inputted in the column indexed at 4 (therefore, the fifth column). The reason I write `data[i + 1, 4] = 1` after validating the pattern is to force the algorithm to buy on the next open after finishing with the

3 Since I use an hourly time frame throughout the book, the previous row refers to the previous hourly candlestick.

current close price. In parallel, the proxy for a short signal is the number –1 in the column indexed at 5 (therefore, the sixth column).

And voilà, you have coded the necessary conditions to create signals based on the open of the row. How would you code the following conditions?

- A long signal is generated on the next open price whenever the current close price is higher than the close price two periods ago.
- A short signal is generated on the next open price whenever the current close price is lower than the close price two periods ago.

The answer is in the following code snippet:

```
def signal(data):

    data = add_column(data, 2)

    for i in range(len(data)):

        try:

            # Bullish signal
            if data[i, 3] > data[i - 2, 3]:

                data[i + 1, 4] = 1

            # Bearish signal
            elif data[i, 3] < data[i - 2, 3]:

                data[i + 1, 5] = -1

        except IndexError:

            pass

    return data
```

There you have it! The way you code signals is by using functions and then calling them. Calling a function is synonymous to applying it and seeing it realized.

To call a signal function, use the following syntax:

```
my_data = signal(my_data)
```

The `return` statement obliges you to redefine your array to be what you want it to be (as defined by the signal function). This obligation comes from the fact that you are adding columns and, therefore, you must re-create the array.

Now, let's proceed to the third algorithm: visual representation of the signals. It may seem that this is an optional step, and it is, but it is helpful to look at a sample of the signals so that you get an idea of what to expect. You must also visually check the signals of the patterns to make sure the algorithm is detecting the right configurations.

Creating the Signal Charts

When you create and apply long and short signals, you have the core of your strategy ready, so you need to evaluate it in two different ways:

Subjectively
: Visually inspect multiple signal charts to detect anomalies that can give hints to coding bugs (e.g., a wrong pattern detected by the algorithm). This can also help you understand better the frequency and the succession of signals. This is what you will see in this section.

Objectively
: Objective evaluation is the more important method. It is about calculating performance metrics such as the hit ratio. You will see and code all the metrics needed in the next section as I discuss them in depth.

The simple diagram in Figure 2-1 can help you understand where you stand at this point. Visualization is the third algorithm within the framework after importing the historical data and coding the signals.

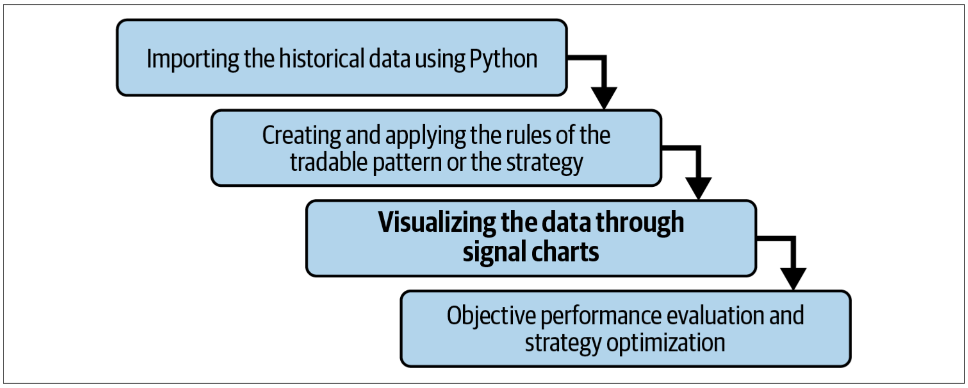

Importing the historical data using Python
Creating and applying the rules of the tradable pattern or the strategy
Visualizing the data through signal charts
Objective performance evaluation and strategy optimization

Figure 2-1. Algorithmic diagram

Chapter 1 covered how to import historical OHLC data from the MetaTrader 5 platform, and you have also seen how to load your own data in case it is not included on the MetaTrader 5 platform. This chapter has already discussed how to code and apply the signals generated from a set of conditions called the Alpha pattern. Let's continue with that same example.

Here is a refresher of the trading conditions leading to a valid Alpha pattern:

- A long signal is generated on the next open whenever the current low is lower than the low price 5 periods ago and the low price 13 periods ago but higher than the low price 21 periods ago. Simultaneously, the close price of the current bar must be higher than the close price 3 periods ago.
- A short signal is generated on the next open whenever the current high price is higher than the high price 5 periods ago and the high price 13 periods ago but lower than the high price 21 periods ago. Simultaneously, the close price of the current bar must be lower than the close price 3 periods ago.

The next step is to visualize these signals using simple bar charts. I then apply upward-pointing arrows where long signals are generated and downward-pointing arrows where short signals are generated. Later in the book, I create these signal charts using candlesticks, but since I have not introduced those yet, I will use *simple bar charts*, which are black vertical lines joining the highs and the lows of every bar. Figure 2-2 shows a simple bar.

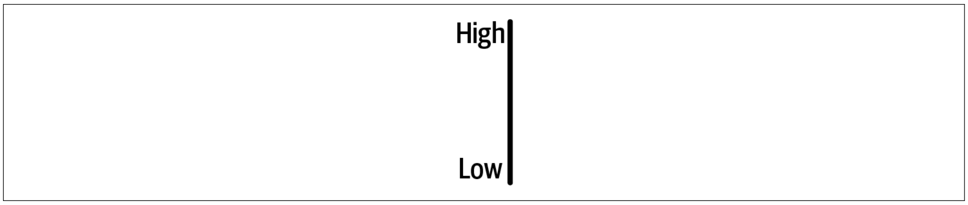


Figure 2-2. A simple high to low bar

As a reminder, I have not yet discussed technical analysis, so don't worry if you feel overwhelmed with information, as everything should be clearer by the next chapter.

Take a look at Figure 2-3. It is an example of a simple bar chart where a vertical line has been drawn between the high and the low of every hour.

Let's apply the signals from the Alpha pattern on the bars so that you know the positioning of your trades relative to the price action. This allows you to see what has happened to the price every time you have gotten an Alpha signal.

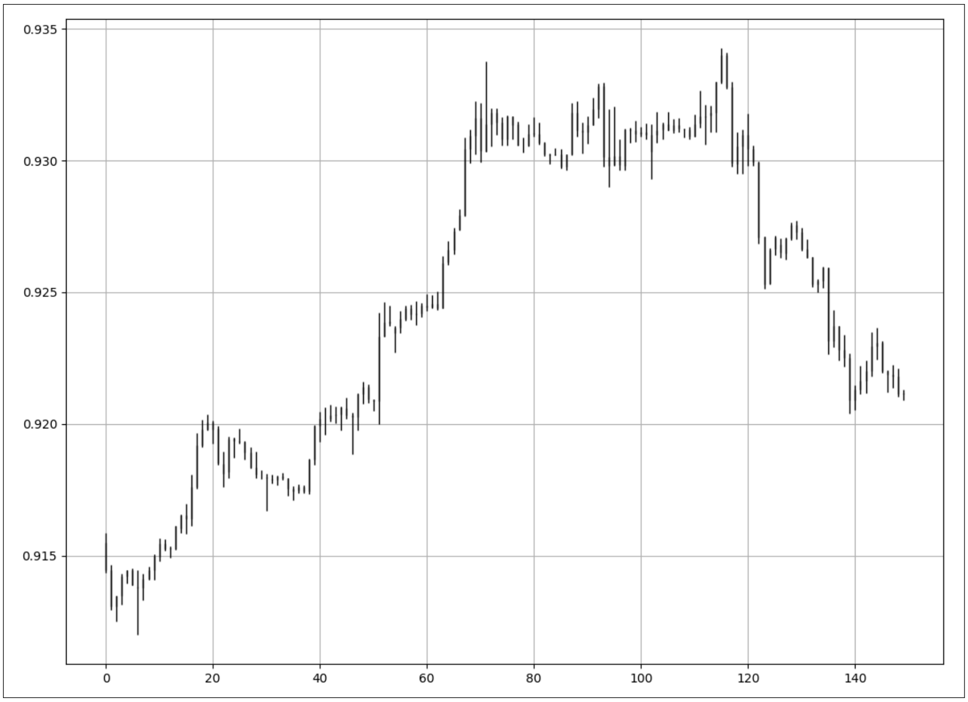


Figure 2-3. Simple bar chart on USDCHF

Use the following function, which I call `signal_chart()`. Before I define the signal chart, here is how to create the simple bar chart by using the `ohlc_plot_bars()` function, defined as follows:

```
def ohlc_plot_bars(data, window):

    sample = data[-window:, ]

    for i in range(len(sample)):

        plt.vlines(x = i, ymin = sample[i, 2], ymax = sample[i, 1],
        color = 'black', linewidth = 1)

        if sample[i, 3] > sample[i, 0]:

            plt.vlines(x = i, ymin = sample[i, 0], ymax = sample[i, 3],
            color = 'black', linewidth = 1)

        if sample[i, 3] < sample[i, 0]:

            plt.vlines(x = i, ymin = sample[i, 3], ymax = sample[i, 0],
            color = 'black', linewidth = 1)

        if sample[i, 3] == sample[i, 0]:
            plt.vlines(x = i, ymin = sample[i, 3], ymax = sample[i, 0] +
            0.00003, color = 'black', linewidth = 1.00)

        plt.grid()
```

The function allows you to plot the simple bar chart. It first starts by defining the *lookback period*, which is the number of recent observations to include in the visualization. The variable for the lookback period is `window`; thus, a window of 500 shows you the last 500 observations. It later loops around the observations that are within the variable `window` and plots vertical lines using `plt.vlines()`, as shown in the previous code.

To chart the last 500 bars, use the following syntax:

```
ohlc_plot_bars(my_data, 500)
```

Now, let's add the signals represented by arrows to the simple bar chart. Take a look at the following code block that defines the `signal_chart()` function and notice how it uses the previous function I showed:

```
def signal_chart(data, position, buy_column, sell_column, window = 500):

    sample = data[-window:, ]

    fig, ax = plt.subplots(figsize = (10, 5))

    ohlc_plot_bars(data, window)

    for i in range(len(sample)):

        if sample[i, buy_column] == 1:

            x = i
            y = sample[i, position]

            ax.annotate(' ', xy = (x, y),
                        arrowprops = dict(width = 9, headlength = 11,
                        headwidth = 11, facecolor = 'green', color =
                        'green'))

        elif sample[i, sell_column] == -1:

            x = i
            y = sample[i, position]

            ax.annotate(' ', xy = (x, y),
                        arrowprops = dict(width = 9, headlength = -11,
                        headwidth = -11, facecolor = 'red', color =
                        'red'))
```

The variable `position` should be set to zero because the signals refer to open prices, which makes the arrows well placed. The function is therefore called like this:

```
signal_chart(my_data, 0, 4, 5, window = 250)
```

If you apply this function on the OHLC array where you have already applied the signals from the Alpha pattern, you would get the signal chart in Figure 2-4. Note that upward-pointing arrows represent exactly where long signals have been generated, whereas downward-pointing arrows represent where short signals have been generated. Figure 2-4 shows a signal chart on the hourly USDCHF values.

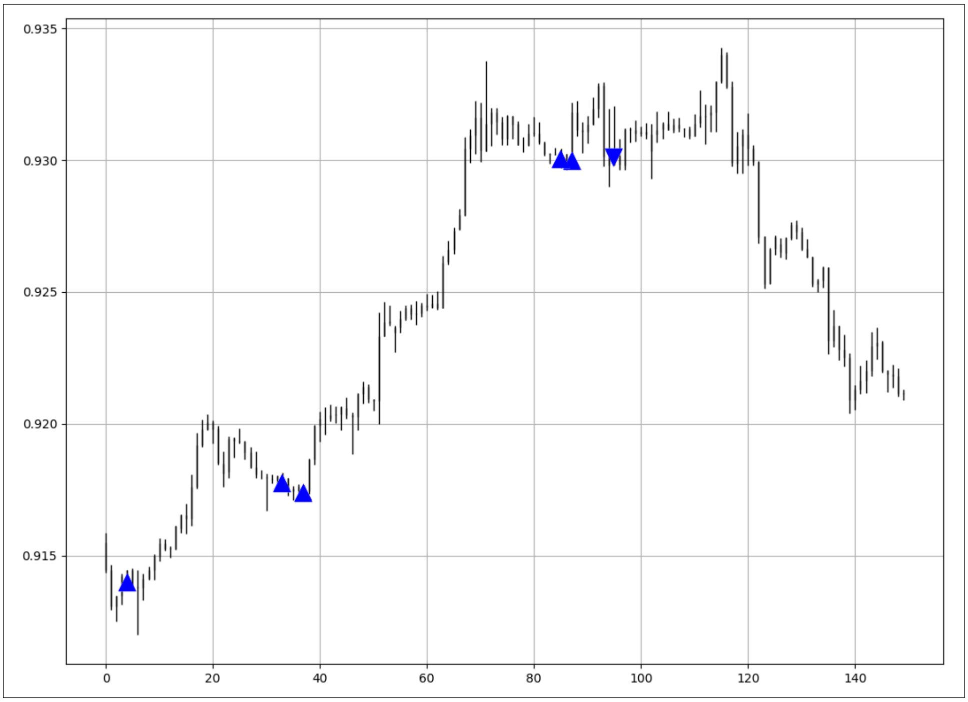


Figure 2-4. Signal chart on USDCHF

Let's recap what you have done so far. You started by defining and coding the signals so that you have your proxies for buy and sell orders. Then you coded a chart function that shows these signals superimposed on the price chart, which gives an idea of where you bought and sold in the past.

Note that even though the simple bar chart shows only the highs and lows, the signals are put where the open price is, as it takes into account the totality of the OHLC data.

The following code shows the chronological order of doing this process on the hourly values of USDCHF, as shown in the previous chart:

```
# Choosing the asset
pair = 1

# Time frame
horizon = 'H1'

# Importing the asset as an array
my_data = mass_import(pair, horizon)

# Creating the signal function
def signal(data):

    data = add_column(data, 2)

    for i in range(len(data)):

        try:

            # Bullish Alpha
            if data[i, 2] < data[i - 5, 2] and data[i, 2] <
               data[i - 13, 2] and data[i, 2] > data[i - 21, 2] and
               data[i, 3] > data[i - 1, 3] and data[i, 4] == 0:

                    data[i + 1, 4] = 1

            # Bearish Alpha
            elif data[i, 1] > data[i - 5, 1] and data[i, 1] >
                 data[i - 13, 1] and data[i, 1] < data[i - 21, 1] and
                 data[i, 3] < data[i - 1, 3] and data[i, 5] == 0:

                    data[i + 1, 5] = -1

        except IndexError:

            pass

    return data

# Calling the signal function
my_data = signal(my_data)

# Charting the latest 150 signals
signal_chart(my_data, 0, 4, 5, window = 150)
```

Figure 2-5 shows the same process applied on the hourly values of EURUSD. Note that the pattern is hypothetical and has no scientific backing.

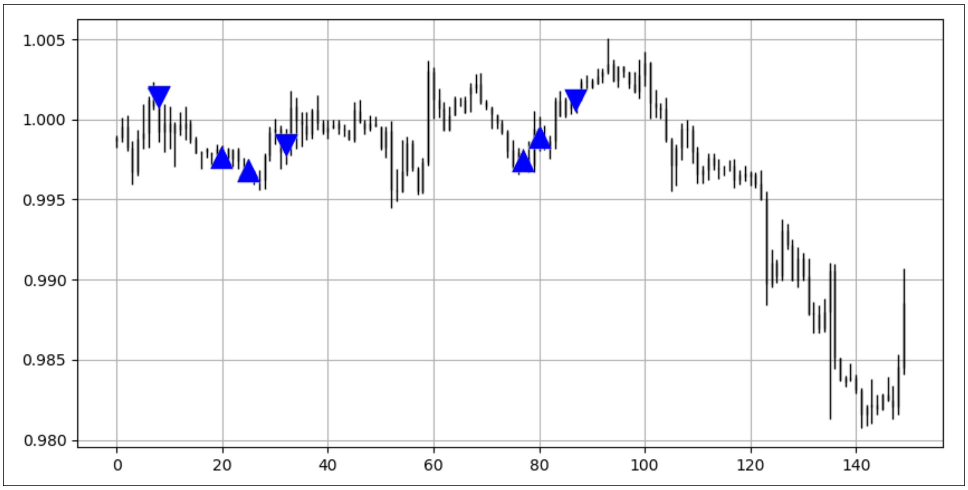


Figure 2-5. Signal chart on the EURUSD

Coding Performance Evaluation Functions

Visualizing signals is only a first step. You need objective data that tells you whether you have a winning or losing trading system.

For this, you need *performance evaluation*, which is simply the calculation of different metrics and ratios that give you hints about past performance. Your job afterward is to interpret these metrics and try to tweak the use of the pattern to improve the metrics.

Remember, a trading strategy can easily be a few conditions that lead to a pattern; therefore, even though this book discusses the details of candlestick patterns, I will briefly review trading strategies. Chapters 10 and 11 discuss a few strategies that naturally involve candlestick patterns.

Performance evaluation also involves the expectation that past performance indicates future performance. This kind of persistence is unlikely, but back-testing and evaluating past performance is the best you can do to validate and take your strategy live.

The Hit Ratio

We humans like to be right more often than we are wrong. Typically, we are proud of our achievements and ashamed of our failures, which is why we are comforted when we find out that most of our decisions led to the expected and desired results. Being right most of the time provides an ego boost that everyone welcomes. After all, who doesn't want to be seen as successful?

The *hit ratio* in trading jargon is the number of past profitable trades divided by the total number of realized past trades. This means that the hit ratio measures the percentage of when you were right about the future direction. A 70% hit ratio means that, on average, you tend to make money on 70 trades out of every 100, which is not bad, but you must absolutely be careful because this is a double-edged sword, and you will see why later. Now, let's take a look at the mathematical representation of the hit ratio:

$$\text{Hit ratio} = \frac{\text{Number of profitable trades}}{\text{Number of total trades}}$$

Evidently, the number of total trades includes both the number of profitable and losing trades. The hit ratio is one of the most observed and analyzed performance metrics due to its psychological effect on the receiver. Make sure you calculate the ratio on the total number of realized trades excluding the pending trades.

The Rate of Return

When you invest \$100 and one year later it becomes \$105, what can you say about your rate of return? Instead of stating that you have gained \$5, you can present your profit as a return over the initial investment calculated this way:

$$\text{Rate of return} = \left(\frac{\text{New balance}}{\text{Initial balance}}\right) - 1$$

This means that your rate of return is 5%. Expressing profitability in percentage terms gives better information regarding the magnitude of the gains and losses on the portfolio. Take, for example, these two hypotheticals:

- Portfolio A made \$125,000.
- Portfolio B returned 10%.

Portfolio A does seem more impressive, but what if I told you that both portfolios had a starting (initial) balance of \$2,000,000? This sure makes portfolio A the underperformer, as it made only 6.25%, whereas portfolio B made 10% (\$200,000).

You can also calculate several types of returns, namely, the *gross* rate of return and the *net* rate of return. Surely what should interest you is the latter as it is net of fees. Let's see the difference between the two in the following example:

- Initial balance on 01/01/2021 = \$1,000,000
- Final gross balance on 12/31/2021 = \$1,175,000
- Total unpaid commissions and fees incurred during 2021 = \$35,000
- Unpaid research provider fees during 2021 = \$10,000

In this example, the gross rate of return is the one that does not consider the fees paid to the broker and other third-party vendors:

$$\text{Gross rate of return} = \left(\frac{1{,}175{,}000}{1{,}000{,}000}\right) - 1 = 17.5\%$$

The net return is closer to what you actually gain and is found through this calculation:

$$\text{Net rate of return} = \left(\frac{1{,}175{,}000 - 35{,}000 - 10{,}000}{1{,}000{,}000}\right) - 1 = 13\%$$

The portfolio is still profitable but has taken a hit after subtracting the costs. Some portfolios switch from winners to losers after removing the costs, which is why it is important to choose a broker with an acceptable fee structure so that it does not eat away your profits over time. Even small differences in commissions can have a huge impact on active traders.

The Profit Factor

The *profit factor* is a quick measure to see how much you are winning for every 1 currency unit of loss. It is calculated as the ratio between gross overall profit and gross overall loss, meaning that you divide the sum of the profits from all profitable trades by the sum of the losses from all losing trades. Mathematically, it is expressed as follows:

$$\text{Profit factor} = \frac{\text{Gross total profit}}{|\text{Gross total loss}|}$$

It is important to divide by the absolute value of the gross total loss. Let's consider an example of a portfolio that had \$127,398 profits in 2020 and \$88,318 losses. What would be the profit ratio in this case?

The answer is 1.44, which is interpreted as gaining on average $1.44 for every $1.00 lost. Whenever the profitability is positive, the profit factor is bigger than 1, and whenever the profitability is negative, the profit factor is less than 1. Some traders like to optimize their strategies by tweaking them until they find the highest profit factor.

The Risk-Reward Ratio

A good trading system is measured by the amount of reward acquired per amount of risk taken to achieve the reward. When you risk $1.00 to gain $1.00, the risk-reward ratio is equal to 1.00, and you have 50% chance of either winning or losing the same amount, unless you have a statistical edge that over time makes you win more often than you lose.

This statistical edge is the notion of the hit ratio. When you enter a trade and set your *target* (the level at which you close out profitably) and your *stop* (the level at which you close out at a loss to avoid a worse outcome), you can actually calculate your theoretical (expected) risk-reward ratio.

Here is a simple example based on buying the cryptocurrency Cardano (ADA) relative to the USDT (a proxy for the USD):

- Buy ADAUSDT at $1.00.
- Set stop at $0.95.
- Set target at $1.10.

What is the risk-reward ratio of this trade, and how can you interpret it? Simply put, you are risking $0.05 to gain $0.10, which means your reward is double the risk; hence, the risk-reward ratio is 2.00.

$$\text{Risk reward ratio} = \frac{|\text{Entry price} - \text{Target price}|}{|\text{Entry price} - \text{Stop price}|}$$

The preceding formula shows how to calculate the theoretical or expected risk-reward ratio, which is set before the trade.

A rule of thumb is to generally try to target strategies that offer a risk-reward ratio close to 2.00, as this gives you enough margin of predictability error to stay in the green. Assuming same quantity trading, with a 2.00 risk-reward ratio, you only need a hit ratio of 33.33% to break even.

The break-even hit ratio is the minimum hit ratio needed to achieve zero profit and loss, excluding costs and fees. As it is just an indicative measure, the break-even hit ratio is rarely presented in performance reports. However, you can easily calculate it through the risk-reward ratio like this:

$$\text{Break-even hit ratio} = \frac{1}{1 + \text{Risk reward ratio}}$$

That gives you a negative relationship between the hit ratio and the risk-reward ratio, as the less you risk and increase your reward, the less you are likely to actually hit the reward (target) because you are closer to the risk (stop). Consider the following example:

- Hit ratio since 01-01-2021 = 43.67%
- Realized risk-reward ratio since 01-01-2021 = 2.11

What would be the break-even hit ratio in this case? By using the formula, you can find the following result:

$$\text{Break-even hit ratio} = \frac{1}{1 + 2.11} = 32.15\%$$

This means that you have a winning strategy because you have on average 43–44 profitable trades per 100 trades. Furthermore, for each of these trades, you win on average 2.11 times the amount you lose, and this is what makes the difference. Proper risk management is what makes profitable trading systems. A first look on the hit ratio leaves the eye unimpressed, but by glancing at the risk-reward, a whole new picture emerges.

In reality, some trades can be closed before getting stopped out or before seeing their targets, and this is due to various reasons, like getting another signal in the same direction. Therefore, you have two different risk-reward ratios:

The theoretical risk-reward ratio
: This is generally set before the trade and is a forecast.

The realized risk-reward ratio
: This is the average profit per trade divided by the average loss per trade, which gives an idea of how close you are to your theoretical risk-reward ratio.

Let's take a look at another example to interpret the two ratios:

- Theoretical risk-reward ratio set on 01-01-2021 = 2.00
- Average gain per trade during 2021 = $241,597
- Average loss per trade during 2021 = $127,222

The realized risk-reward ratio is therefore 1.90, which is slightly less than the theoretical one. This is acceptable as sometimes you exit some trades before reaching the stop or target level. The ratios presented in the back-tests in this book are the realized ones.

The Number of Trades

The frequency of trades is important for performance evaluation. A rule of thumb to keep in mind is to have at least 30 trades in order to meet the minimum threshold for reliability. Of course, over the years, the frequency of trades must be much higher. Some patterns are so rare that they are unlikely to give many signals, thus preventing you from properly judging them.

Unfortunately, some patterns can be quite rare, but, in any case, the book is not about blemishing and glorifying them, as some are actually very bad and nonpredictive yet are used by many retail traders to analyze the market. This brings us to the second utility of the book: demystification. It is time to see how to code these metrics in Python.

Creating a Performance Evaluation Function

You have finally ended the theoretical part of the discussion of performance metrics. Even though in reality there are more metrics that dig into the details of performance, you do not need to calculate them all to get a quick idea of what works and what does not.

For example, the rate of return is not very useful in the back-tests later in the book because it is a function of position size and transaction costs, so it does not deal with predictability directly. This is why I use only the other four metrics:

- The hit ratio gives you a preliminary idea of the predictability of the pattern or the strategy.
- The profit factor allows you to see whether the generated profits are greater than the generated losses regardless of position sizing.
- The realized risk-reward ratio shows you how much you are being rewarded compared to the risk you are taking.
- The frequency of signals reveals whether the results are meaningful or not and whether you expect frequent signals or uncommon ones.

The performance metrics are bundled into a Python function that I call `performance()`, shown in the following code block. I will explain after the code how it works and what it outputs:

```
def performance(data,
                open_price,
                buy_column,
                sell_column,
                long_result_col,
                short_result_col,
                total_result_col):

    # Variable holding period
    for i in range(len(data)):

        try:

            if data[i, buy_column] == 1:

                for a in range(i + 1, i + 1000):

                    if data[a, buy_column] == 1 or data[a, sell_column] \
                                                   == -1:

                        data[a, long_result_col] = data[a, open_price] - \
                                                   data[i, open_price]

                        break

                    else:

                        continue

            else:

                continue

        except IndexError:

            pass

    for i in range(len(data)):

        try:

            if data[i, sell_column] == -1:

                for a in range(i + 1, i + 1000):

                    if data[a, buy_column] == 1 or data[a, sell_column] \
                                                   == -1:
```

```
                        data[a, short_result_col] = data[i, open_price] -\
                                                    data[a, open_price]

                        break

                    else:

                        continue

            else:
                continue

        except IndexError:

            pass

    # Aggregating the long & short results into one column
    data[:, total_result_col] = data[:, long_result_col] + \
                                data[:, short_result_col]

    # Profit factor
    total_net_profits = data[data[:, total_result_col] > 0, \
                        total_result_col]
    total_net_losses  = data[data[:, total_result_col] < 0, \
                        total_result_col]
    total_net_losses  = abs(total_net_losses)
    profit_factor     = round(np.sum(total_net_profits) / \
                        np.sum(total_net_losses), 2)

    # Hit ratio
    hit_ratio         = len(total_net_profits) / (len(total_net_losses) \
                        + len(total_net_profits))
    hit_ratio         = hit_ratio * 100

    # Risk-reward ratio
    average_gain            = total_net_profits.mean()
    average_loss            = total_net_losses.mean()
    realized_risk_reward    = average_gain / average_loss

    # Number of trades
    trades = len(total_net_losses) + len(total_net_profits)

    print('Hit Ratio         = ', hit_ratio)
    print('Profit factor     = ', profit_factor)
    print('Realized RR       = ', round(realized_risk_reward, 3))
    print('Number of trades  = ', trades)
```

The function is not as complex as it seems. For instance, it takes seven variables that refer to the array and its column indices. The variable `data` represents the OHLC array that contains the historical data and the signals generated by the pattern. Remember, the signals are in the form of 1s in case of a long trigger and in the form of −1s in case of a short trigger. The variable `open_price` is simply the first column of the array, which represents the open price for each time step. In your case, it would be each hour as you are using the hourly time frame.

Do not forget that the first column of an array is indexed at zero in Python, so the variable `open_price` always has a value of 0. The following two variables represent the positions where the buy and sell signals are stored, respectively. With simple patterns, it is generally 4 and 5, while with more complex patterns, it can be more. The `long_result_col` and the `short_result_col` variables are the indices of the results found from the `buy_column` and `sell_column`.

This means that whenever you exit a long position, the result of that position is stored in the column described by the variable `long_result_col`. Finally, `total_result_col` is always the next column to come after `short_result_col` as it is the sum of the two result columns. The `total_result_col` column is created to facilitate the calculation of the performance metrics. Now, to call the function, use the following syntax:

```
my_data = performance(my_data, 0, 4, 5, 6, 7, 8)
```

The statement calls the performance function on an array called `my_data`, which you have seen how to import, then it replaces the variables with the necessary values. If you are unsure of how to find the variables, carefully reread the previous paragraph.

A Hypothetical Example: Appraising Performance

Before I close this chapter, let's take a look at a full example and interpret its results. After all, back-testing is about understanding what the strategy or pattern has given so that you improve it. The following details are for a portfolio that uses a single strategy between 2017 and 2021:

- Total number of trades = 2,348
- Profitable trades = 1,236
- Losing trades = 1,112
- Theoretical risk-reward ratio = 2.00
- Total net gain from trades = $457,995
- Total net loss from trades = $321,589
- Average gain per trade = $370.54
- Average loss per trade = $289.19

What is the hit ratio of the period between 2017 and 2021?

The hit ratio is simply the number of profitable trades divided by the total number of realized trades. In this example, it is 52.64%.

What is the net profit factor between 2017 and 2021? How would you interpret it?

The profit factor is the total profits divided by the total losses. In this example, it is 1.42, which is well above 1.00. The portfolio is generating $1.42 for every $1.00 it loses.

What is the realized risk-reward ratio between 2017 and 2021? And how does it compare to the theoretical risk-reward ratio?

The realized risk-reward ratio is the ratio of the average gain per trade to the average loss per trade. In this example, it is 1.28. It is well below the theoretical risk-reward ratio, which stands at 2.00, and therefore the portfolio is experiencing suboptimal risk management, probably due to early closing of positions.

How should we interpret the frequency of signals?

With 2,348 trades over the course of five years, the portfolio has a relatively high trading activity, about 469 trades per year. Transaction costs are a cause of concern with such a high number of trades. Statistically, the performance metrics should properly describe the portfolio's situation since there are many signals.

Is this a well-managed portfolio?

Even though profitability as shown by the profit factor is high, the manager should consider improving the realized risk-reward ratio while maintaining a hit ratio above 50%. This can be done through tweaks, optimizations, and even reviewing the entry/exit techniques. The manager should also investigate filtering the trades in order to decrease brokerage fees. However, the portfolio is in the green and seems to have a predictive strategy.

You have now finished building the core of the algorithms that you will use to analyze and back-test the candlestick patterns. Before presenting those patterns, you have one more stop to make, which is to understand technical analysis, the field of research where candlestick pattern recognition lies.

CHAPTER 3

Introducing Technical Analysis

Technical analysis relies on the visual interpretation of the price action's history to determine the likely aggregate direction of the market. It relies on the idea that the past is the best predictor of the future. There are several types of techniques within the vast field that is technical analysis, notably the following:

Charting analysis
: This is where you apply subjective visual interpretation techniques onto charts. You generally use methods like drawing support and resistance lines as well as retracements to find inflection levels that aim to determine the next move.

Indicator analysis
: This is where you use mathematical formulas to create objective indicators that can be either trend following or contrarian. Among known indicators are moving averages and the relative strength index (RSI), both of which are discussed in greater detail in this chapter.

Pattern recognition
: This is where you monitor certain recurring configurations and act on them. A *pattern* is generally an event that emerges from time to time and presents a certain theoretical or empirical outcome. In finance, it is more complicated, but certain patterns have been shown to add value across time, and this may partly be due to a phenomenon called *self-fulfilling prophecy* (a process by which an initial expectation leads to its confirmation). Among known patterns are candlestick patterns, which are the protagonists of this book.

Let's take a quick tour of the history of technical analysis so that you have a better idea of what to expect. Technical analysis relies on three principles:

History repeats itself.
: You are likely to see clusters during trends and ranges. Also, certain configurations and patterns are likely to have a similar outcome most of the time.[1]

The market discounts everything.
: It is assumed that everything (all fundamental, technical, and quantitative information) is included in the current price.

The market moves in waves.
: Due to different time frames and needs, traders buy and sell at different frequencies, therefore creating trends and waves as opposed to a straight line.

Unfortunately, technical analysis is overhyped and misused by the retail trading community, which gives it a somewhat less than savory reputation in the professional industry. Every type of analysis has its strengths and weaknesses, and there are successful fundamental, technical, and quantitative investors, but there are also failed investors from the three fields.

Fundamental analysis relies on economic and financial data to deliver a judgment on a specific security or currency with a long-term investment horizon, whereas quantitative analysis is more versatile and is more often applied to short-term data. It uses mathematical and statistical concepts to deliver a forecast.

Among other assumptions, technical analysis suggests that markets are not efficient, but what does that mean? *Market efficiency* states that information is already embedded in the current price and that price and value are the same thing. When you buy an asset, you are hoping that it is *undervalued* (in fundamental analysis jargon) or *oversold* (in technical analysis jargon), which is why you believe the price should go up to meet the value. Therefore, you are assuming that the value is greater than the price. Market efficiency rebuffs any claims that the price does not equal the value and therefore suggests that any alpha trading must not result in above average returns (*alpha trading* is the act of engaging in speculative operations to perform better than a benchmark, which is typically an index or a weighted measure).

The market efficiency hypothesis is the technical analyst's greatest enemy, as one of its principles is that in the weak form of efficiency, you cannot earn excess returns from technical analysis. Hence, technical analysis gets shot down right at the beginning, and then fundamental analysis gets its share of the beating.

1 This assumes a nonrandom probability that over the long term shows deterministic characteristics.

It is fair to assume that at some point in the future, markets will have no choice but to be efficient due to the number of participants and the ease of access to information. However, as political and abnormal events show us, markets tend to be anything but efficient.

An example of a political event that triggers panic and irrationality in the markets is the Russian invasion of Ukraine. Similarly, an example of an abnormal economic event is an unexpected interest rate hike from a central bank.

Charting Analysis

Before you can understand what charting analysis is, you need to know what you see when opening a chart, more specifically a candlestick chart.

Let's assume that the market opens at $100. Some trading activity occurs. Let's also record the high price ($102) and the low price ($98) printed during the hourly period. Also, record the hourly close price ($101). Recall that these four pieces of data are referred to as *open*, *high*, *low*, and *close* (OHLC). They represent the four basic prices that are necessary to create candlestick charts.

Candlesticks are extremely simple and intuitive. They are box-shaped chronological elements across the timeline that contain the OHLC data. Figure 3-1 shows everything you need to know about how a candlestick works.

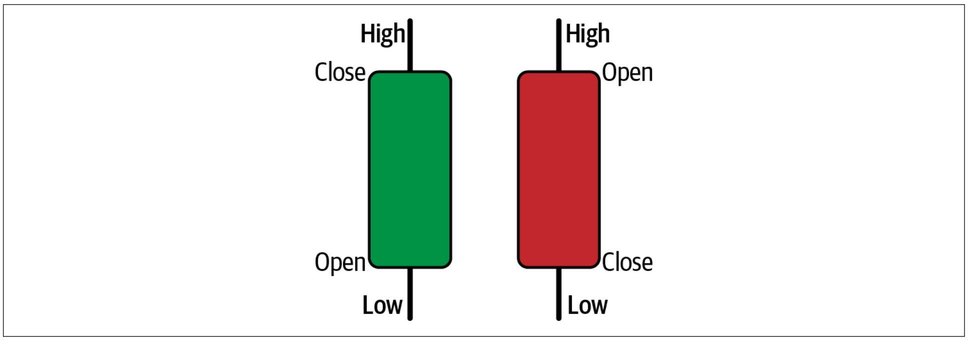


Figure 3-1. On the left, a bullish candlestick; on the right, a bearish candlestick

A *bullish* candlestick has a close price higher than its open price, whereas a *bearish* candlestick has a close price lower than its open price.

Candlestick charts are among the most famous ways to analyze financial time series. They contain more information than simple line charts and have more visual interpretability than bar charts. Many libraries in Python offer charting functions, but in my opinion it is always better to do it yourself. Let's start from the beginning, with line charts.

A *line chart* is created by joining the close prices chronologically. It is the simplest way to chart an asset. It contains the least information among the three chart types since it shows only the close price.

Plotting basic line plots is extremely easy in Python and requires only one line of code. You have to make sure that you have imported a library called `matplotlib`, which handles the plotting for you. The following code snippet shows how to plot a line chart of a one-dimensional array that contains close prices for the EURUSD in an hourly time frame:

```python
# Importing the necessary charting library
import matplotlib.pyplot as plt

# The syntax to plot a line chart
plt.plot(my_data, color = 'black', label = 'EURUSD')

# The syntax to add the label created above
plt.legend()

# The syntax to add a grid
plt.grid()
```

Figure 3-2 shows a line chart on EURUSD. The chart provides information only about the closing price and the aggregate direction.

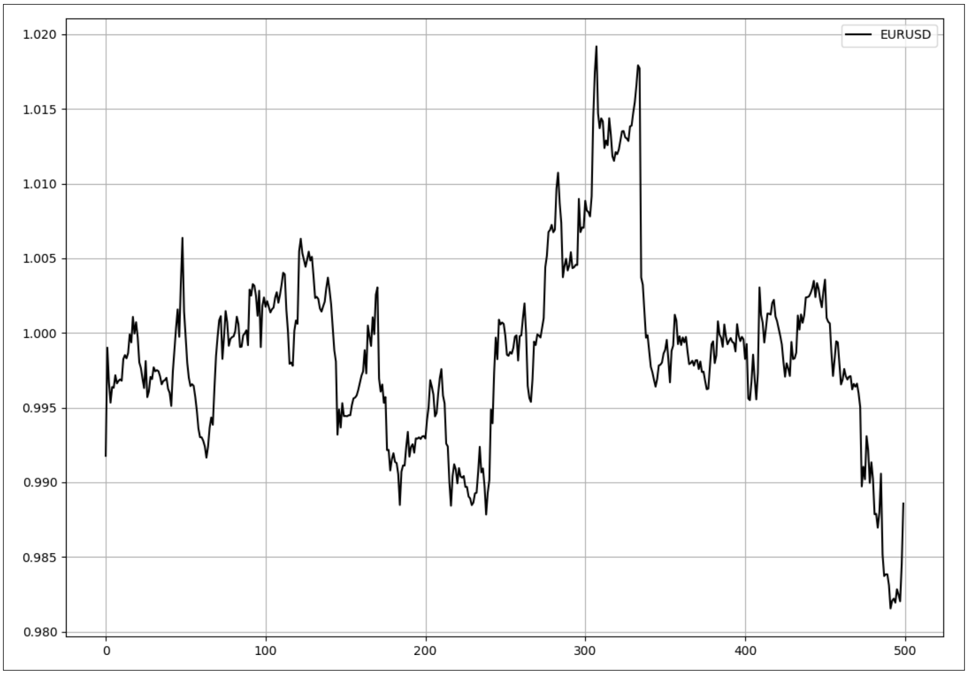


Figure 3-2. Line chart on EURUSD

Now let's take it to the next level with candlestick charts.

The easiest way to do this is to think about vertical lines at each time step (as with the simple bar charts shown in Chapter 2). Follow these steps:

1. Plot vertical lines for each row representing the highs and lows. For example, on OHLC data, you use a `matplotlib` function called `vlines()`, which plots a vertical line on the chart using a minimum (low price) value and a maximum (high price). Therefore, this function plots bars that span between each high and low. Figure 3-3 shows a simple bar chart (like the one from Chapter 2) where each bar is the range between the high and low of the EURUSD.
2. Repeat the first step, only this time apply the new vertical line on the open and close prices. This gives you a vertical line inside a vertical line, but that's not visible. How do you fix this? Impose a color condition and a greater width that represents the candlestick's body.

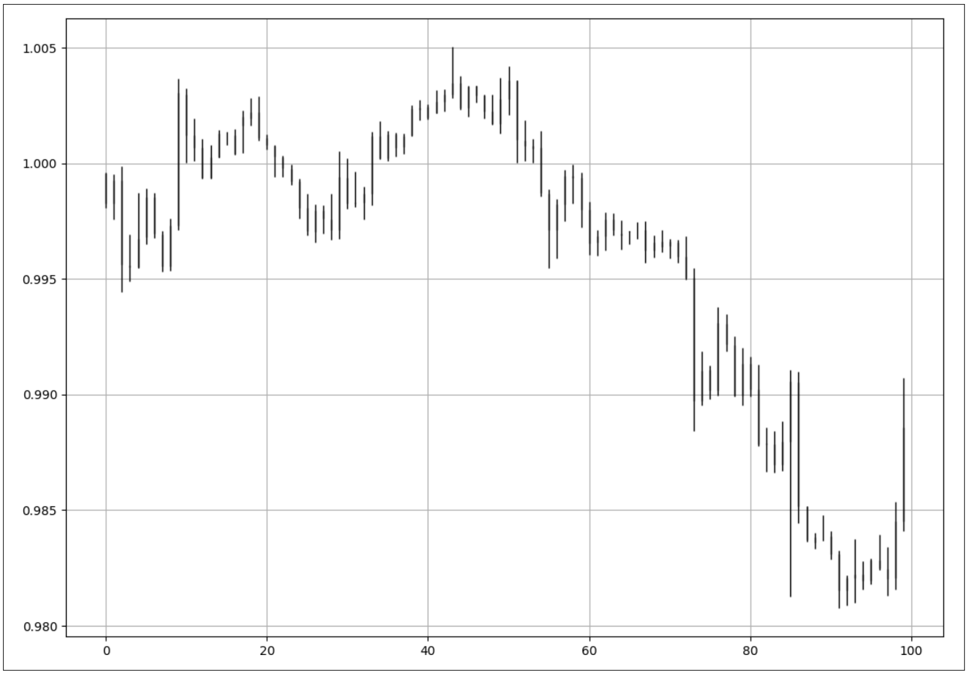


Figure 3-3. Simple bar chart on EURUSD

Figure 3-4 shows a complete candlestick chart on EURUSD. You can see that with the color-coding and the visibility of the totality of the OHLC data, you are able to get more information such as volatility and the general tendency of the close price to the open price. This type of information is not available in line charts.

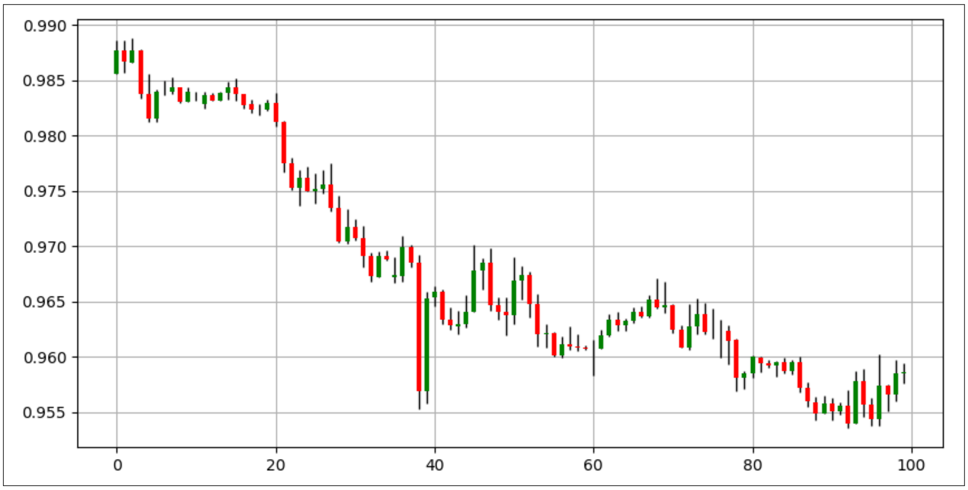


Figure 3-4. Candlestick chart on EURUSD

The full code to do this is as follows:

```
def ohlc_plot_candles(data, window):

    sample = data[-window:, ]

    for i in range(len(sample)):

        plt.vlines(x = i, ymin = sample[i, 2], ymax = sample[i, 1],
                   color = 'black', linewidth = 1)

        if sample[i, 3] > sample[i, 0]:

            plt.vlines(x = i, ymin = sample[i, 0], ymax = sample[i, 3],
                       color = 'green', linewidth = 3)

        if sample[i, 3] < sample[i, 0]:

            plt.vlines(x = i, ymin = sample[i, 3], ymax = sample[i, 0],
                       color = 'red', linewidth = 3)

        if sample[i, 3] == sample[i, 0]:

            plt.vlines(x = i, ymin = sample[i, 3], ymax = sample[i, 0] +
                       0.00003, color = 'black', linewidth = 1.00)

    plt.grid()
```

To call the function and show the last 100 candlesticks, use the following syntax:

```
ohlc_plot_candles(my_data, window = 100)
```

Some people prefer plotting candlesticks in other colors. To do that, you have to tweak the `color` parameter. For example, the following code plots a gray (bullish) and black (bearish) candlestick chart:

```
def ohlc_plot_candles(data, window):

    sample = data[-window:, ]

    for i in range(len(sample)):

        plt.vlines(x = i, ymin = sample[i, 2], ymax = sample[i, 1],
                   color = 'black', linewidth = 1)

        if sample[i, 3] > sample[i, 0]:

            plt.vlines(x = i, ymin = sample[i, 0], ymax = sample[i, 3],
                       color = 'grey', linewidth = 3)

        if sample[i, 3] < sample[i, 0]:

            plt.vlines(x = i, ymin = sample[i, 3], ymax = sample[i, 0],
                       color = 'black', linewidth = 3)

        if sample[i, 3] == sample[i, 0]:

            plt.vlines(x = i, ymin = sample[i, 3], ymax = sample[i, 0] +
                       0.00003, color = 'black', linewidth = 1.00)

    plt.grid()
```

Charting analysis is the task of finding support and resistance lines through subjective drawing. *Lines*, whether horizontal or diagonal, are the essence of finding levels to predict the market's reaction:

- *A support level* is a level from where the market should bounce, as it is implied that demand should be higher than the supply around it.
- *A resistance level* is a level from where the market should retreat, as it is implied that supply should be higher than the demand around it.

The asset's direction on a timeline axis can be threefold: *uptrend* where prices are making higher highs, *downtrend* where prices are making lower lows, and *sideways (or ranging)* where prices fluctuate around the same level for extended periods of time.

Figure 3-5 shows a support level on EURUSD close to 0.9850. Generally, traders start thinking about buying when a price is close to support. This is in anticipation of a reaction to the upside since the balance of power should shift more to the demand (positive) side, where traders accept to pay a higher price as they expect an even higher price in the future (remember the price-to-value argument discussed earlier). The implication here is that most traders see a price that is lower than the value.

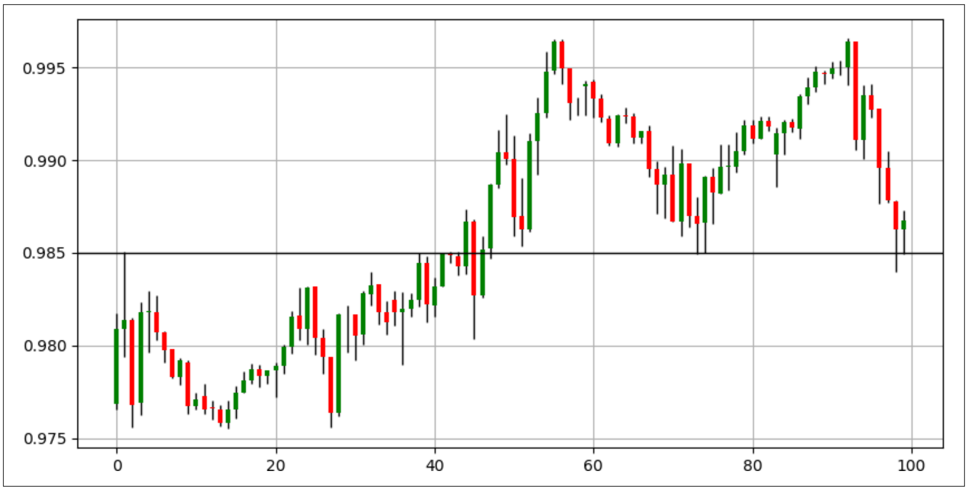


Figure 3-5. Candlestick chart on EURUSD showing support at 0.9850

Figure 3-6 shows a resistance level on EURUSD close to 0.9960. Generally, traders start thinking about shorting the market when it is close to resistance. This is in anticipation that a reaction to the downside should occur since the balance of power should shift more to the supply side. The implication here is that most traders see a price that is higher than the value.

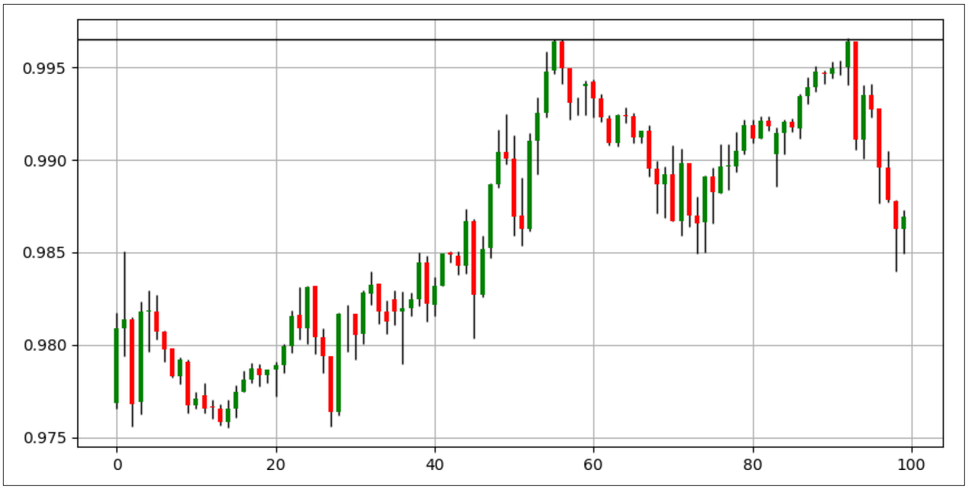


Figure 3-6. Candlestick chart on EURUSD showing resistance at 0.9960

Ranging (sideways) markets give more confidence that horizontal support and resistance lines will work. This is because of the already implied general balance between supply and demand. Therefore, if there is excess supply, the market would adjust quickly, as demand should rise enough to stabilize the price.

Figure 3-7 shows a ranging market trapped between two horizontal levels; this is the case of EURUSD. Whenever the market approaches the resistance line in a ranging market, you should have more confidence that a drop will occur than you would in a rising market, and whenever it approaches support, you should have more confidence that a bounce will occur than you would in a falling market.

Furthermore, charting analysis is also applied on trending markets. This comes in the form of ascending and descending channels. They share the same inclination as horizontal levels but with a bias (discussed later).

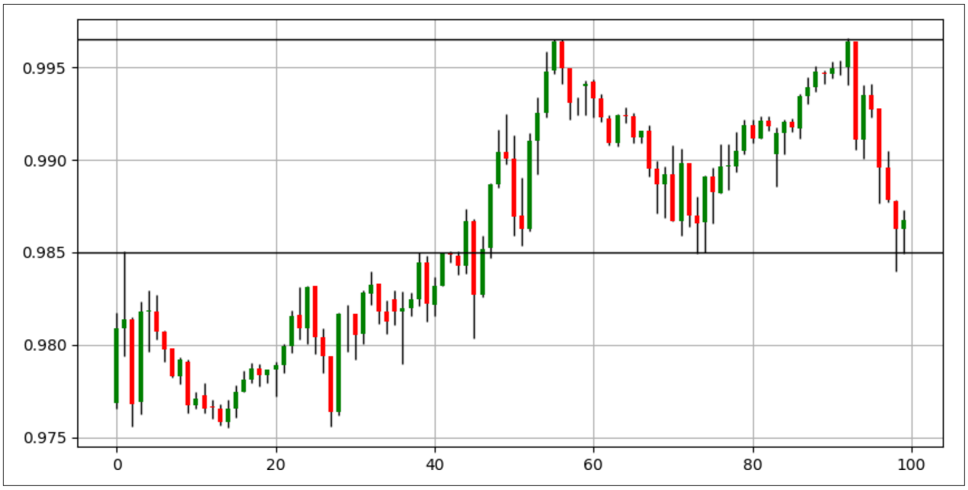


Figure 3-7. Candlestick chart on EURUSD showing support at 0.9850 and resistance at 0.9960

Figure 3-8 shows an *ascending channel* where support and resistance points rise over time to reflect the bullish pressure stemming from a steadily rising demand force.

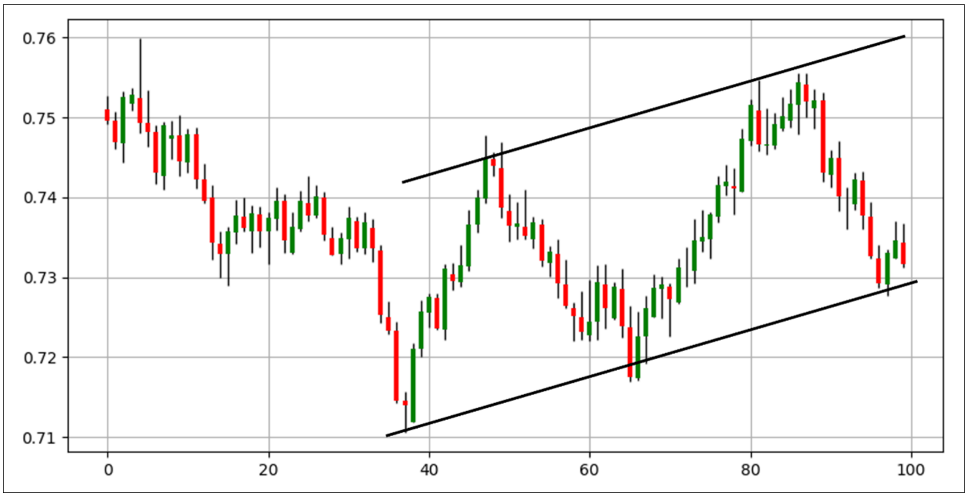


Figure 3-8. Candlestick chart on AUDUSD showing an ascending channel

Traders seeing this would anticipate a bullish reaction whenever the market approaches the lower part of the ascending channel and would expect a bearish reaction whenever the market approaches the upper part of the channel.

This has no sound scientific backing because nothing says that the market must move in parallel, but the self-fulfilling prophecy may be why such channels are considered predictive in nature.

Figure 3-9 shows a descending channel where support and resistance points fall with time to reflect the bearish pressure coming from a steadily rising supply force. Generally, bearish channels tend to be more aggressive as fear dominates greed and sellers are more panicky than buyers are greedy.

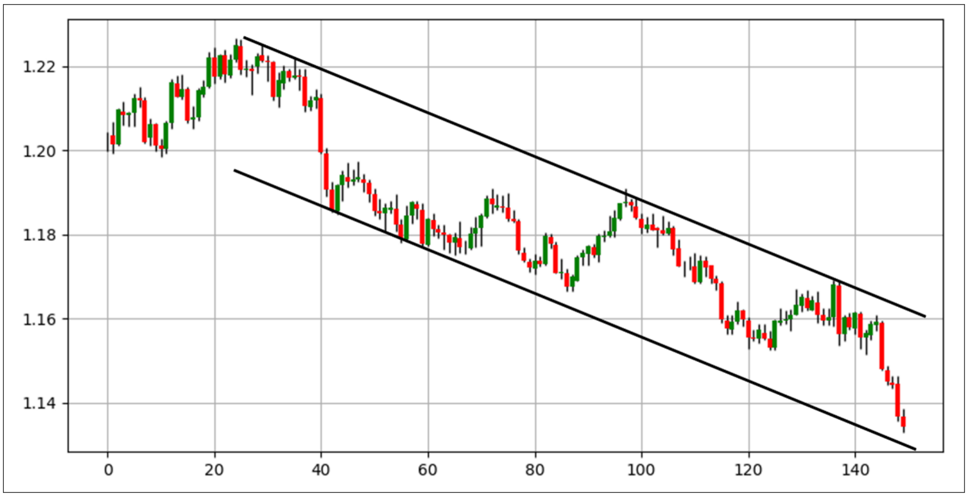


Figure 3-9. Candlestick chart on EURUSD showing a descending channel

I mentioned a bias when dealing with ascending and descending channels. I refer to this bias as the *invisible hand*. Here's why:

"The trend is your friend." This saying, coined by Martin Zweig, means that with ascending channels, you need to be focusing more on buying whenever the market reverts to the support zone. That's because you want the invisible hand of the bullish pressure to increase your probability of a winning trade. Similarly, in the case of a descending channel, focus more on short selling whenever the market reaches the upper limit. The full version of Zweig's axiom goes as follows: "The trend is your friend, until the end when it bends." This means that at any point in time, the market may change its regime, and any friendship with the trend gets terminated. In the end, charting analysis is subjective in nature and relies more on the experience of the trader or analyst.

It is worth mentioning also that there are many ways of finding support and resistance levels other than drawing them through visual estimation:

Fibonacci retracements
: This is where you use Fibonacci ratios to give out reactionary levels. Fibonacci retracements are usually calculated on up or down legs so that you know where the market will reverse if it touches one of these levels. The problem with this method is that it is very subjective and, as with any other technique, not perfect. The advantage is that it gives many interesting levels.

Pivot points
: With pivot points you use simple mathematical formulas to find levels. Based on yesterday's trading activity, you use formulas to project today's future support and resistance levels. Then whenever the market approaches the levels, you try to fade the move by trading in the opposite direction.

Moving averages
: These are discussed in the next section. They are dynamic in nature and follow the price. You can also use them to detect the current market regime.

The best way to find support and resistance levels is to combine as many techniques as possible so that you have a certain confluence of methods, which in turn will increase your conviction for the initial idea. Trading is a numbers game, and stacking as much odds as possible on your side should eventually increase your chances for a better-performing system.

Indicator Analysis

Indicator analysis is the second-most used technical analysis tool. It generally accompanies charting to confirm your initial idea. You can consider *indicators* as assistants. They can be divided into two types:

Trend-following indicators
: Used to detect and trade a trending market where the current move is expected to continue. Therefore, they are related to the persistence of the move.

Contrarian indicators
: Used to fade the move and best used in sideways markets as they generally signal the end of the initial move. Therefore, they are related to the expected reversal of the move (and therefore to the anti-persistence of the move).

The following sections present two pillars of technical analysis, moving averages (trend following) and the relative strength index (contrarian).

Moving Averages

The most famous trend-following overlay indicator is the *moving average*. Its simplicity makes it without a doubt one of the most sought-after tools. Moving averages help confirm and ride the trend. You can also use them to find support and resistance levels, stops, and targets, as well as to understand the underlying trend.

There are many types of moving averages, but the most common is the simple moving average where you take a rolling mean of the close price, as shown in the following formula:

$$\text{Moving average}_i = \frac{Price_i + Price_{i-1} + \ldots + Price_{i-n}}{n}$$

This is the simple mean that you use in statistics and basically any other part in life. It is simply the total values of the observations divided by the number of observations.[2]

Figure 3-10 shows the 30-hour simple moving average applied on USDCAD. The term *30-hour* means that I calculate the moving average of the latest 30 periods in case of hourly bars.

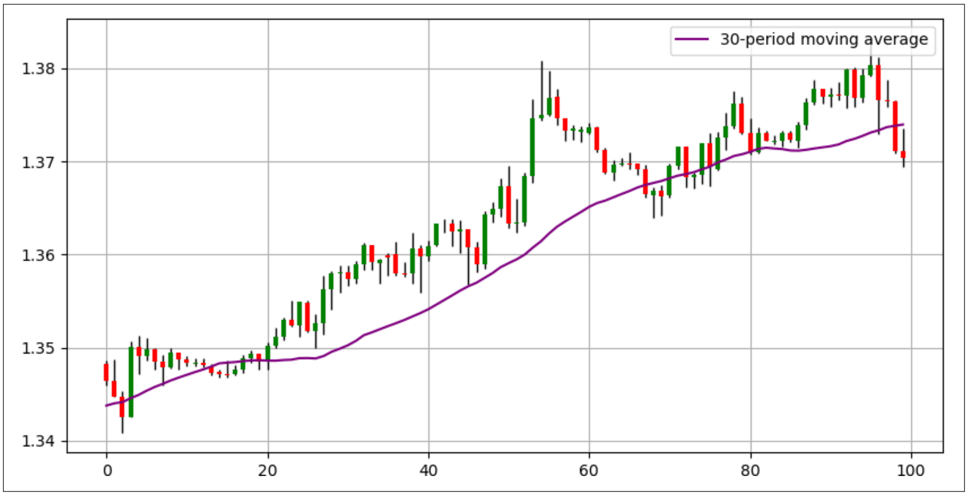


Figure 3-10. Candlestick chart on USDCAD with a 30-hour simple moving average

2 You can also think of the mean as the sum divided by the quantity.

Rules of thumb with moving averages include the following:

- Whenever the market is above its moving average, a bullish momentum is in progress, and you are better off looking for long opportunities.
- Whenever the market is below its moving average, a bearish momentum is in progress, and you are better off looking for short opportunities.
- Whenever the market crosses over or under its moving average, you can say that the momentum has changed and that the market may be entering a new regime (trend).

You can also combine moving averages so that they give out signals. For example, whenever a short-term moving average crosses over a long-term moving average, a bullish crossover occurs, and the market may continue to rise. This is also referred to as a *golden cross*.

In contrast, whenever a short-term moving average crosses under a long-term moving average, a bearish crossover has occurred, and the market may continue to drop. This is also referred to as a *death cross*.

Figure 3-11 shows USDCAD with a 10-hour (closer to the market price) and a 30-hour moving average (further from the market price). You can notice that, at the beginning, a golden cross appeared and signaled a beginning of a bullish trend.

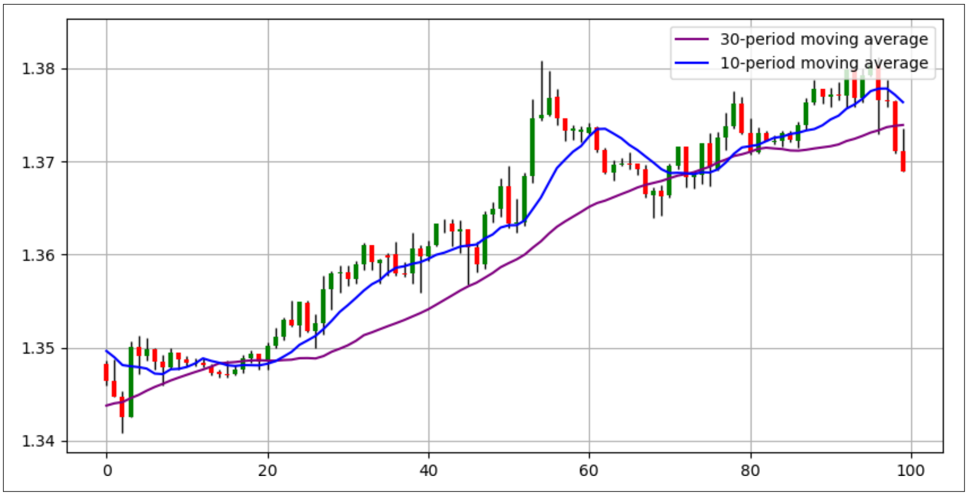


Figure 3-11. Candlestick chart on USDCAD with a 30-hour and a 10-hour simple moving average

To code a moving average, use this function:

```
def ma(data, lookback, close, position):

    data = add_column(data, 1)

    for i in range(len(data)):

            try:

                data[i, position] = (data[i - lookback + 1:i + 1,
                                     close].mean())

            except IndexError:

                pass

    data = delete_row(data, lookback)

    return data
```

To calculate a 30-period (or hour, depending on your time frame) moving average on the close prices, you need to define the `ma()` function and call it as follows:

```
# Setting the lookback period
lookback = 30

# Setting the index of the close price column
close_column = 3

# Setting the index of the moving average column
ma_column = 4

# Calling the moving average function
my_data = ma(my_data, lookback, close_column, ma_column)
```

The Relative Strength Index

Let's now discuss the contrarian indicator. First introduced by J. Welles Wilder Jr.,[3] the *relative strength index* (RSI) is one of the most popular and versatile bounded indicators. It is mainly used as a contrarian indicator where extreme values signal a reaction that can be exploited.

3 See *New Concepts in Technical Trading Systems* by J. Welles Wilder Jr. (1978), published by Trend Research.

Use the following steps to calculate the default 14-period RSI:

1. Calculate the change in the closing prices from the previous ones.
2. Separate the positive net changes from the negative net changes.
3. Calculate a smoothed moving average on the positive net changes and on the absolute values of the negative net changes.
4. Divide the smoothed positive changes by the smoothed absolute negative changes. Refer to this calculation as the relative strength (RS).
5. Apply this normalization formula for every time step to get the RSI:

$$RSI_i = 100 - \frac{100}{1 + RS_i}$$

The *smoothed* moving average is a special type of moving average developed by the creator of the RSI. It is smoother and more stable than the simple moving average.

Generally, the RSI uses a lookback period of 14 by default, although each trader may have their own preferences on this. Here's how to use this indicator:

- Whenever the RSI is showing a reading of 30 or less, the market is considered to be oversold, and a correction to the upside might occur.
- Whenever the RSI is showing a reading of 70 or more, the market is considered to be overbought, and a correction to the downside might occur.
- Whenever the RSI surpasses or breaks the 50 level, a new trend might be emerging, but this is generally a weak assumption and more theoretical than practical in nature.

Figure 3-12 shows EURUSD versus its 14-period RSI in the second panel. Indicators should be used to confirm long or short bias and are very helpful in timing and analyzing the current market state.

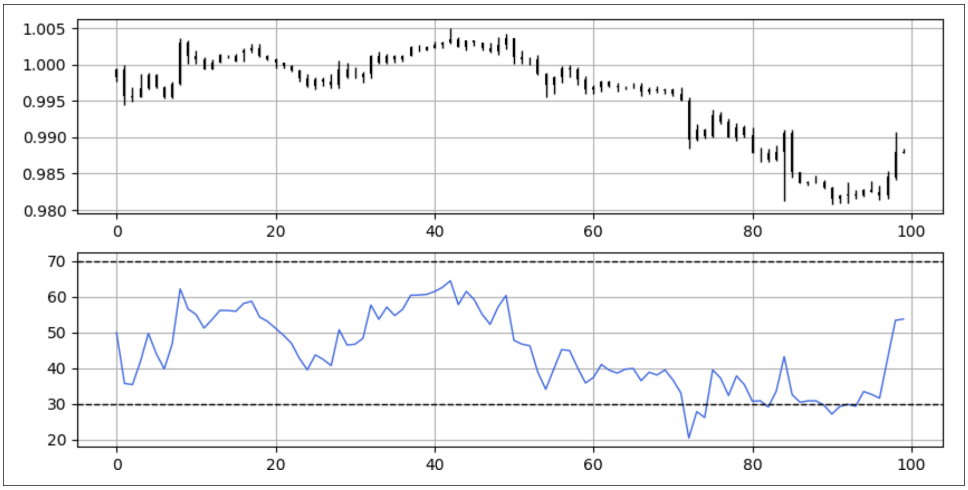


Figure 3-12. Hourly EURUSD values in the top panel with the 14-period RSI in the bottom panel

To create the RSI using the usual array method, define the smoothed moving average first as follows:

```python
def smoothed_ma(data, alpha, lookback, close, position):

    lookback = (2 * lookback) - 1

    alpha = alpha / (lookback + 1.0)

    beta  = 1 - alpha

    data = ma(data, lookback, close, position)

    data[lookback + 1, position] = (data[lookback + 1, close] * alpha) +
                                    (data[lookback, position] * beta)

    for i in range(lookback + 2, len(data)):

            try:

                data[i, position] = (data[i, close] * alpha) +
                                     (data[i - 1, position] * beta)

            except IndexError:

                pass

    return data
```

Now, let's code the RSI using the following function:

```
def rsi(data, lookback, close, position):

    data = add_column(data, 5)

    for i in range(len(data)):

        data[i, position] = data[i, close] - data[i - 1, close]

    for i in range(len(data)):

        if data[i, position] > 0:

            data[i, position + 1] = data[i, position]

        elif data[i, position] < 0:

            data[i, position + 2] = abs(data[i, position])

    data = smoothed_ma(data, 2, lookback, position + 1, position + 3)
    data = smoothed_ma(data, 2, lookback, position + 2, position + 4)

    data[:, position + 5] = data[:, position + 3] / data[:, position + 4]

    data[:, position + 6] = (100 - (100 / (1 + data[:, position + 5])))

    data = delete_column(data, position, 6)
    data = delete_row(data, lookback)

    return data
```

To summarize, indicators can be calculated in many ways. The two most commonly used ones are moving averages and the RSI. I come back to them later in Chapters 10 and 11. At the moment, make sure to grasp the concepts of technical analysis. Let's move on to pattern recognition.

Pattern Recognition

Patterns are recurring configurations that show a specific prediction of the ensuing move. Patterns can be divided into the following types:

Classic price patterns
: These are known technical reversal price patterns, which are extremely subjective and can be considered unreliable due to the difficulty of back-testing them without taking subjective conditions. However, they are still used by many traders and analysts.

Timing patterns
: Based on a combination of timing and price. These patterns are less well-known but can be powerful and predictive when used correctly.

Candlestick patterns
: This is where OHLC data is used to predict the future reaction of the market. Candlesticks are one of the best ways to visualize a chart as they harbor many patterns that could signal reversals or confirm the move. They are discussed in greater depth in coming chapters.

Classic price patterns refer to theoretical configurations such as double tops and rectangles. They are usually either reversal or continuation patterns:

Continuation price patterns
: These are configurations that confirm the aggregate ongoing move. Examples include rectangles and triangles.

Reversal price patterns
: These are configurations that fade the aggregate ongoing move. Examples include head and shoulders and double bottoms.

Old-school chartists are familiar with double tops and bottoms, which signal reversals and give the potential of said reversals. Despite their simplicity, they are subjective, and some are not visible like others.

This hinders the ability to know whether they add value or not. Figure 3-13 shows an illustration of a double top where a bearish bias is given right after the validation of the pattern, which is usually breaking the line linking the lows of the bottom between the two tops. This line is referred to as the *neckline*.

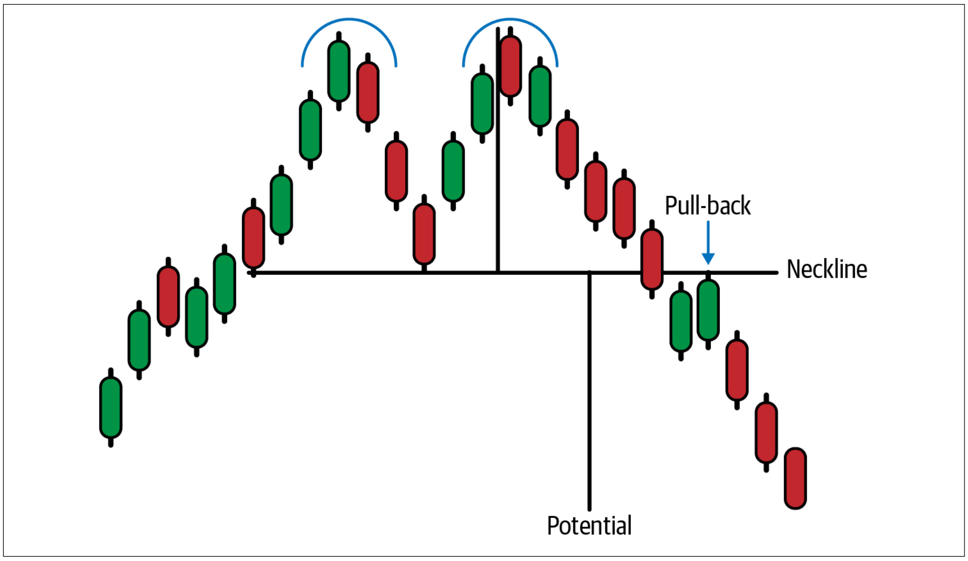


Figure 3-13. Double top illustration

Notice these three important elements in a double top:

The neckline
: This is the line linking the lowest low between the two peaks and the beginning/end of the pattern. It serves to determine the pull-back level, which we'll define next.

The pull-back
: Having broken the neckline, the market should shape a desperate attempt toward the neckline but fails to continue higher as the sellers use this level as re-entry to continue shorting. Therefore, the pull-back level is the theoretical optimal selling point after validating a double top.

The potential
: This is the target of the double top. It is measured as the midpoint between the top of the pattern and the neckline projected lower and starting from the same neckline point.

The double top or bottom can have any size, but preferably it should be visible to most market participants so that its impact is bigger. Theoretically, the pattern's psychological explanation is that with the second top or bottom, the market has failed to push the prices beyond the first peak and therefore is showing weakness, which might be exploited by the seller.

There are other patterns that are more objective in nature; i.e., they have clear rules of detection and initiation. These are all based on clear objective conditions and are not subject to the analyst's discretion. This facilitates their back-testing and evaluation.

Before ending this introductory chapter, I want to point out some misconceptions and best practices of technical analysis so that you start the right way.

Common Pitfalls of Technical Analysis

As simple as technical analysis is, it can be misused, and this unfortunately fuels an everlasting debate about its utility and place relative to fundamental analysis. The important thing is to set expectations right and to remain within the means of logical thinking. This section talks about known pitfalls of technical analysis that you must make sure to avoid in order to maximize your survival rate in the financial jungle.

Wanting to Get Rich Quickly

This pitfall is mostly psychological, as it deals with a lack of discipline and poor management. The need to make money as soon as possible to impress society pushes a trader to make emotional and bad decisions with regard to trades and trading-related activities.

This is also related to the fact that with the need to make money, you are likely to keep changing your trading plan because you believe that the new plan is a quicker way to wealth.

When you do not have enough faith in yourself and think that other people are better than you at making money, you are more likely to follow them, especially because of the abundance of information they provide. No one except you can change your future.

Forcing the Patterns

A common psychological deficiency known as *confirmation bias* prevents traders from seeing signals that contradict their already established views.

Sometimes, you have an initial view on some market and therefore start looking for anything that agrees with the view, and this can also force patterns into existence even though there is no validity to them.

You have to always be neutral in your analysis and approach elements with caution while retaining a maximum of objectivity. Of course, this is easier said than done, and the best alternative for absolute neutrality is an algorithmic approach that comes at the expense of the human intelligence factor.

Hindsight Bias, the Dream Smasher

Technical analysis looks good in the past. Everything looked obvious and easy to predict, even with very basic strategies; however, when you apply the latter, you find deceiving results due to the harsh reality that your brain is wired to trick you into believing the past was perfectly predictable.

This is also why back-testing results almost always outperform forward-testing. When you look at past patterns and believe that they should have been easy to spot, you are exhibiting *hindsight bias*. To remedy this problem, you have to back-test using unbiased algorithms. Only then can you know for sure if the pattern adds value or not. Many retail traders fall into the trap of taking a general and simplistic look at the past to determine the validity of their strategies, which then fail to perform. I discuss these biases in greater depth in Chapter 12.

Assuming That Past Events Have the Same Future Outcome

You've heard the saying "History doesn't repeat itself, but it does often rhyme." This saying is key to understanding how the markets function. When you apply technical analysis, you must not expect exact outcomes from past signals and patterns. Rather, you must use them as guidelines and probabilistic elements that suggest markets may have a similar reaction that can be correlated with past reactions.

Trading is a numbers game, and you have to stack the odds in your favor. Humans sometimes do behave the same way when they are confronted with similar events. However, as different participants enter and leave the activity of buying and selling, with their motives changing all the time, you can be certain that the future reaction of the market after encountering a pattern similar to one in the past will not be exactly the same, although it might *rhyme* with the past, meaning it might have a correlated reaction on average.

This being said, do not expect to time the market perfectly every time you see a distinct pattern.

Making Things More Complicated Than They Need to Be

Another saying is "Everything should be made as simple as possible, but no simpler." This is a perfect description of how you should do research and trade.

Financial markets are highly complex, semi-random environments that require more than simple strategies, yet strategies must not be so sophisticated that you fall into the trap of *overfitting*, a common issue where traders perfectly predict the past and assume it will behave exactly the same in the future, thus resulting in huge paper gains in the past but huge real losses in the future.

Technical Analysis Best Practices

When you misuse something, you tend to shy away from blaming yourself, instead placating your ego and relieving yourself of any responsibility. Let's list some best practices that optimize the way you analyze the markets technically.

Harness the Power of Different Time Frames

Many analysts and traders analyze the markets by looking only at the short-term horizon. An example of this is to look for opportunities on the hourly chart while disregarding the market's configuration on the daily chart. It is a strict rule to know that the longer time frame (weekly and monthly) is always more important than the shorter time frame, meaning that if you have a solid resistance on the higher time frame, you should not really look to buy close to that level even if you see a bullish configuration on the shorter time frame.

The reason you should do this is to respect the *principle of the alignment of the planets*, which in this case dictates the fact that the more elements that confirm each other, the better your confidence in the probability of a successful trade. A good framework to follow is to analyze the market in at least three different time frames, starting with the longer time horizon and going down.

Use More Than One Strategy or Indicator

Some analysts argue that charting analysis works better than indicators or patterns, while others disagree. The best solution is to use all three and create a strategy that combines the best formations from each type of analysis. Why use only one when you have the choice of using more? Remember, it is a numbers game, and you need to have as many opinions as possible.

Choose the Right Type of Strategy for Current Market Conditions

As stated, the market tends to trend or move sideways. The two main types of strategies can be categorized into contrarian or trend following, with the former relying on the concept of mean reversion (fading the aggregate move) and the latter relying on the momentum principle (following the aggregate move).

Before you choose the strategy to deploy, you have to know whether the market is trending or ranging, and this can be done by using moving averages or other trend identification techniques.

The main point is not to use contrarian strategies in a strongly trending market and not to use trend-following strategies in ranging markets. This is easier said than done because it is extremely difficult, if not impossible, to estimate the current and next market regime.

Don’t Underestimate Default Parameters

Default parameters are actually not as horrible as people portray them, for one obvious reason: they have the most visibility to traders and analysts.

To look at an example, consider the relative strength index. You can say that the default version that uses 14 periods as a lookback window is probably more reliable than a version that uses 55 periods, assuming the same strategy, and this is because more people look at the default version and base their decisions on it than use the 55 periods version.

A key point in trading is to make sure that the configuration you are seeing can be seen by other participants so that you maximize your chances of a winning trade. This is not to say that you must use exactly the same techniques as others, but it can be helpful sometimes not to tweak the indicators too much.

CHAPTER 4

Classic Trend-Following Patterns

Trend following, a concept so simple yet so complicated. It may be easy not to worry about timing the market and just riding it, but the reality is that doing so can be challenging due to a plethora of random variables impacting the price. A steadily rising market may get destabilized by one economic or political event but then continue to move smoothly, discombobulating the traders who got stopped out during the volatility tsunami.

This chapter covers the classic (already known and established in the world of technical analysis) trend-following candlestick patterns. These are simple and complex configurations that have been around since the dawn of technical analysis and are taught in many introductory and advanced technical analysis courses. The purpose of this chapter is to create the patterns' objective conditions and back-test them so that you can form a basic opinion about their frequency and predictability. Some patterns may occur quite often, while others occur once in a blue moon, thus hindering the ability to properly judge them.

As previously mentioned, the way I back-test the patterns is to assume that I initiate the buy or sell position upon the next open after having validated the signal on the previous close. The time frame chosen for the back-tests in this book is hourly.

The Marubozu Pattern

The first classic trend-following pattern is the *Marubozu*. The word refers to baldness or close-cropped head in Japanese, and you should soon understand why.

It is relatively easy to distinguish between Marubozu candlesticks and normal candlesticks because Marubozu candlesticks do not have any wicks. This means that a bullish Marubozu candlestick has the same open and low price as well as the same close and high price. In contrast, a bearish Marubozu candlestick has the same open and high price as well as the same close and low price. Figure 4-1 shows these two versions of a Marubozu candlestick.

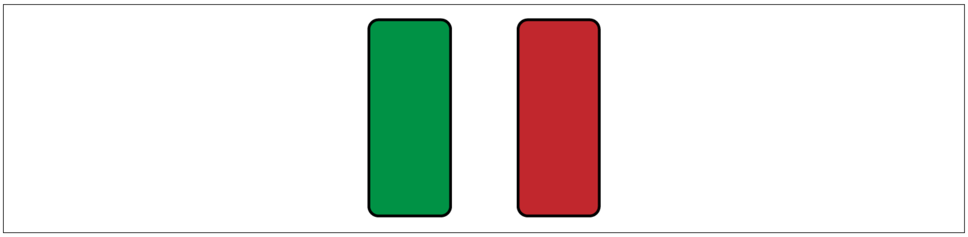


Figure 4-1. On the left, a bullish Marubozu candlestick; on the right, a bearish Marubozu candlestick

The Marubozu pattern typically occurs during shorter time frames, as candles have less time to fluctuate and go out of their bounds. This can be seen when analyzing 1-minute and 5-minute charts and comparing them to daily charts.

A candlestick has more probability to assume a Marubozu pattern when it has less time between the open and close prices.

Before you use any pattern, you must understand the reason for its existence. After all, patterns are not discovered randomly; they must have a rationale behind them. More important, you need to know why you should expect a certain reaction after the validation of the pattern.

The answer lies in market psychology, which leaves its footprints everywhere so that focused and knowledgeable traders pick up on any hints and ride (or fade) the move.

When the market is in a strong uptrend, it rarely makes lower lows and generally closes around the highs due to the sheer force of demand for the underlying security. The maximum force of demand in one candlestick is demonstrated by the absence of a low and by a close at the high, and this is precisely what the bullish Marubozu is all about. When an asset closes at the high price, you get the signal that the buyers are hungry for more, and when no low is lower than the open price, you should have further conviction that there is no interest in selling since no one has managed to push the price lower than its opening.

Similarly, when the market is in a strong downtrend, it rarely makes higher highs and generally closes around the lows due to the overwhelming force of supply for the underlying security. The maximum force of supply in one candlestick is demonstrated by the absence of a high and a close at the low, as demonstrated by the bearish Marubozu.

The bullish Marubozu candlestick is the most powerful representation of a bullish activity, while the bearish Marubozu candlestick is the most powerful representation of a bearish activity.

Detecting the Marubozu pattern can be tricky in some markets that use more than a few decimal points because of the probability of occurrence. For instance, in the currencies market, if you use five decimals on EURUSD, you are likely to not find any Marubozu patterns anytime soon, but when you stick to the usual, less-accurate convention of four decimals (see Chapter 2), you start seeing the pattern more often. This holds true for all markets; therefore, to be able to analyze this pattern, you need to make a small tweak with regard to decimals.

Figure 4-2 shows a clean chart on USDCHF with the rounding function applied as follows:

```
my_data = rounding(my_data, 4)
```

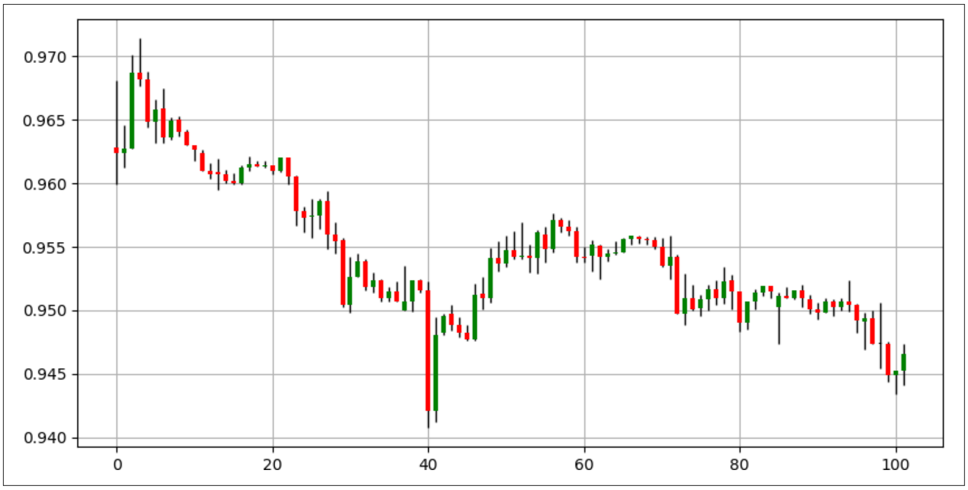


Figure 4-2. Candlestick chart on USDCHF

The next step is to code the signal function.

The *signal function* is the core of the algorithm, as it is in charge of finding the patterns and pinpointing them chronologically and, therefore, outputting buy and sell signals according to specific conditions.

Algorithmically, the conditions need to be as follows:

- If the close price is greater than the open price, the high price equals the close price, and the low price equals the open price, then print 1 in the next row of the column reserved for buy signals.
- If the close price is lower than the open price, the high price equals the open price, and the low price equals the close price, then print –1 in the next row of the column reserved for sell signals.

In the Python language, use the following syntax to code the Marubozu signal function:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] > data[i, open_column] and
              data[i, high_column] == data[i, close_column] and \
              data[i, low_column] == data[i, open_column] and
              data[i, buy_column] == 0:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] < data[i, open_column] and
                data[i, high_column] == data[i, open_column] and \
                data[i, low_column] == data[i, close_column] and
                data[i, sell_column] == 0:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 4-3 shows the trades generated on EURUSD following the signals given by the function.

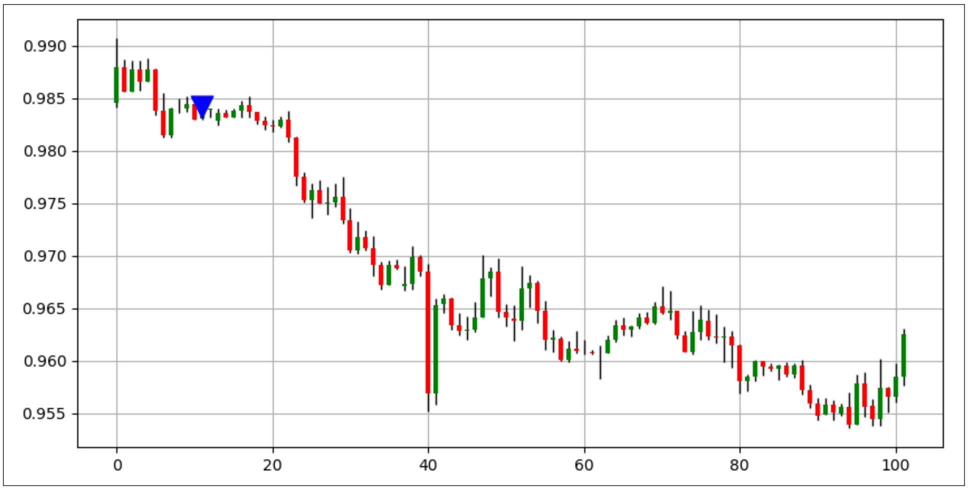


Figure 4-3. Signal chart on EURUSD

Upward-pointing arrows refer to bullish signals, while downward-pointing arrows refer to bearish signals. Figure 4-4 shows another example of the signals on USDCHF.

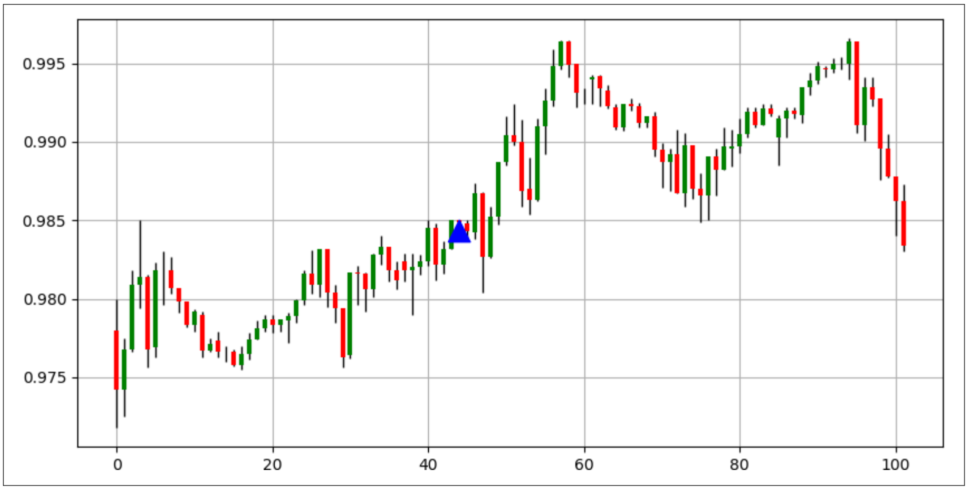


Figure 4-4. Signal chart on USDCHF

Let's now back-test the pattern using the performance evaluation metrics discussed in Chapter 2.

Table 4-1 shows the performance summary on the Marubozu pattern. The trading conditions are as follows:

- The entry is done on the next open price after the validation of the signal on the current close price.
- The exit is done upon getting another signal in either direction (bullish or bearish).
- The transaction costs are omitted from the performance evaluation metrics.
- No risk management system is used.

A *risk management* system entails the use of simple and complex order management techniques that regulate the maximum loss and expected targets.

Table 4-1. Marubozu pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	47.14%	1.06	1.19	963
USDCHF	47.01%	0.95	1.07	1355
GBPUSD	47.27%	0.79	0.89	605
USDCAD	49.06%	0.99	1.03	958
BTCUSD	47.81%	1.38	1.51	433
ETHUSD	39.76%	0.93	1.41	3687
GOLD	43.33%	0.96	1.25	8443
S&P500	41.92%	0.66	0.91	260
FTSE100	50.00%	1.09	1.09	90

The back-tested assets show that the Marubozu pattern is relatively rare and does not seem to add great value. All of the predictions are random as evidenced by the hit ratio. A hit ratio around 50% means that you are as likely to be right as you are to be wrong.

Similarly, the profit factor shows that you are barely generating gross profits to cover gross losses. This is amplified when accounting for transaction costs. A profit factor close to 1.00 is insignificant and is unlikely to give more information as to whether the pattern performs well or not.

If you analyze the risks taken by trading the pattern, you would find that it confirms the randomness of the strategy. By having an average of around 1.00, you are basically risking $1 to earn $1, which can go either way. Finally, the signals are not very frequent but are above the 30 threshold, which is needed to make any statistical interpretations.

The main conclusion is that the Marubozu pattern is unlikely to deliver a profitable strategy on its own. Nevertheless, keep in mind the following points for the back-tests:

- The positions are taken solely based on one pattern and are not combined with other confirming indicators. Most of the time, patterns work better when combined with other techniques and indicators.
- The back-tested universe is limited; therefore, it is possible that the pattern works well on other markets.
- The back-tested time frame is limited since the results reflect only the hourly time frame. Remember, other time frames exist such as five-minute, daily, and weekly.
- The exit technique is variable and depends on the next signal, which may not come before long. Other exit techniques may be more suited such as a fixed one or one that depends on volatility.

Hence, even though the back-test gives a general approximate idea on the ability of the patterns, you must always understand its limitations.

The Three Candles Pattern

The *Three Candles* pattern is a trend-confirmation configuration where the signal is given after printing three same-color candles that have a minimum specified size. A bullish Three Candles pattern is also called the Three White Soldiers and has three big bullish candles with each having a close higher than the previous one. In contrast, a bearish Three Candles pattern is called the Three Black Crows and has three big bearish candles, with each having a close lower than the previous one.

The names Three White Soldiers and Three Black Crows come from the fact that candlesticks were previously charted in black and white with the former representing bearish candles and the latter representing bullish candles.

Figure 4-5 illustrates the Three White Soldiers. Notice that every close price is greater than the one preceding it.

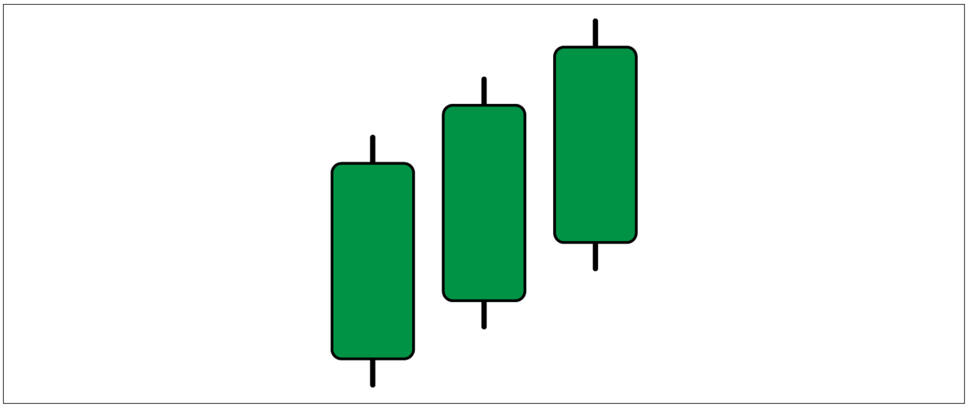


Figure 4-5. A Three White Soldiers pattern

Figure 4-6 illustrates the Three Black Crows. Notice that every close price is lower than the one preceding it.

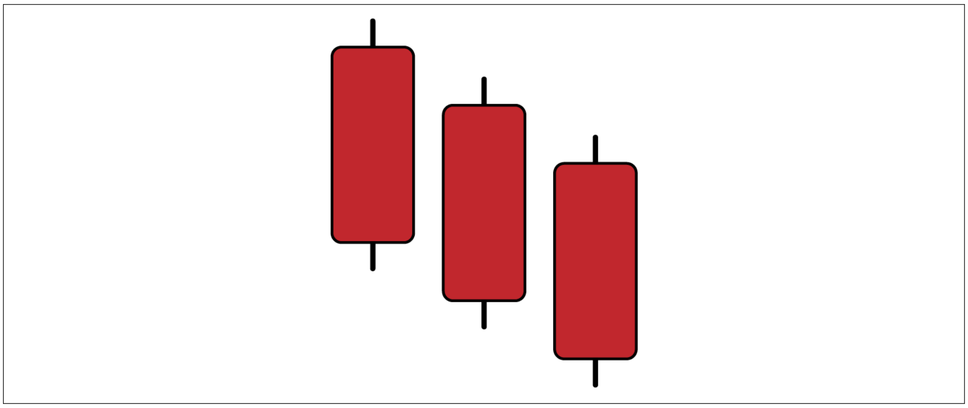


Figure 4-6. A Three Black Crows pattern

The intuition of the pattern is quite simple. It stems from a type of psychological bias called *herding*, where market participants follow the trend because others are doing so. This does not mean that the pattern is based on human deficiency and lack of effort; it simply refers to a common behavior observed in humans where they tend to follow the aggregate trend.

People follow the latest fashion trend because it delivers a psychological reward. Trend-following traders follow the latest trend because they believe it delivers a financial reward or because they have a form of fear of missing out (FOMO).

Financial herding can be defined as the act of following the majority in an attempt to profit as much as possible from the current persistent move. Hence, in essence, herding is actually trend following. When you see three big candles within the same direction, you typically interpret it as a healthy move filled with motivation, conviction, and desire to continue in the same direction. Compare these three big candles to a timid move of small candles where their type alternates between bullish and bearish.

Humans are attracted to confidence and strength and are more likely to follow it. A pattern of three healthy candlesticks is the embodiment of strength and confidence.

Detecting this pattern is easy but must have strict algorithmic rules that dictate more than just three big candles with the same colors; otherwise, you would have plenty of signals. Algorithmically, the conditions need to be as follows:

- If the latest three close prices are each greater than the close prices preceding them and each candlestick respects a minimum body size,[1] then print 1 in the next row of the column reserved for buy signals.
- If the latest three close prices are each lower than the close prices preceding them and each candlestick respects a minimum body size, then print –1 in the next row of the column reserved for sell signals.

The body of the candlestick is the difference between the close and open prices in absolute terms. Therefore, to find the body, follow this mathematical formula:

$$\text{Candlestick body}_i = \left|\text{Close price}_i - \text{Open price}_i\right|$$

```python
def signal(data, open_column, close_column, buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] - data[i, open_column] > body and \
              data[i - 1, close_column] - data[i - 1, open_column] > \
              body and data[i - 2, close_column] - \
              data[i - 2, open_column] > body and data[i, close_column] > \
              data[i - 1, close_column] and data[i - 1, close_column] > \
              data[i - 2, close_column] and data[i - 2, close_column] > \
              data[i - 3, close_column] and data[i, buy_column] == 0:
```

1 Remember, the Three Candles pattern is composed of three big candles. Therefore, you need to code the condition that they must have a minimum size.

```
                data[i + 1, buy_column] = 1

            # Bearish pattern
            elif data[i, close_column] - data[i, open_column] > body and \
                 data[i - 1, close_column] - data[i - 1, open_column] > \
                 body and data[i - 2, close_column] - \
                 data[i - 2, open_column] > body and data[i, close_column] \
                 < data[i - 1, close_column] and data[i - 1, close_column] \
                 < data[i - 2, close_column] and data[i - 2, close_column] \
                 < data[i - 3, close_column] and data[i, sell_column] == 0:

                data[i + 1, sell_column] = -1

        except IndexError:

            pass
```

The function applies the conditions shown previously in order to generate buy and sell signals. The bodies of the candlestick may actually hinder the objectivity of this pattern to some extent since it differs according to volatility. Theoretically, the Three Candles pattern merely refers to the candles as "big" and does not indicate how size should be adjusted; therefore, I use fixed values depending on the asset and the time frame (which is hourly). Table 4-2 shows a summary.

Table 4-2. Three Candles pattern: candle size choice

Asset	Body	Type
EURUSD	0.0005	USD
USDCHF	0.0005	CHF
GBPUSD	0.0005	USD
USDCAD	0.0005	CAD
BTCUSD	50	USD
ETHUSD	10	USD
GOLD	2	USD
S&P500	10	Points
FTSE100	10	Points

You can also adjust the variable `body` depending on volatility, but I will show you another pattern that adjusts to volatility. For now, I will keep the theoretical conditions of the classic patterns untouched and leave the creativity to the modern patterns in Chapters 5 and 7.

Figure 4-7 shows a signal chart on EURUSD.

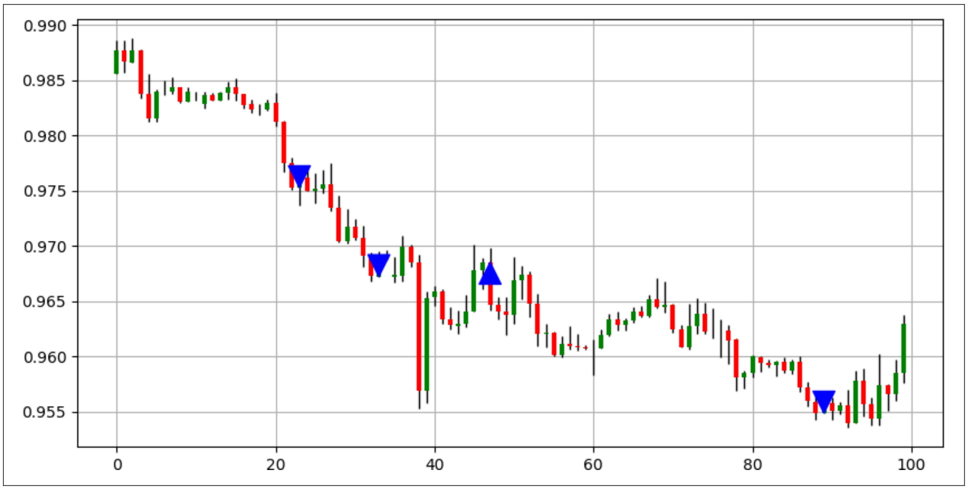


Figure 4-7. Signal chart on EURUSD

Table 4-3 shows the performance summary of the Three Candles pattern.

Table 4-3. Three Candles pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	61.45%	1.05	0.66	2672
USDCHF	62.04%	0.98	0.60	2005
GBPUSD	61.53%	0.96	0.60	3611
USDCAD	60.97%	0.97	0.62	2844
BTCUSD	62.30%	1.00	0.61	1085
ETHUSD	61.22%	0.96	0.61	392
GOLD	61.11%	1.04	0.66	828
S&P500	65.99%	1.10	0.57	741
FTSE100	64.54%	0.97	0.53	1018

Even though the results show a relatively higher hit ratio than with the Marubozu pattern, you have to be careful as the hit ratio as a standalone metric is not useful. You must look at it in parallel to the risk-reward ratio to understand if the predictive ability can be profitable or not.

The first thing you notice is the crucially low risk-reward values, which range between 0.53 and 0.66. This is the first red light that the hit ratio is not impressive at all. In fact, it may not even be enough to deliver positive results.

The Tasuki Pattern

The *Tasuki* pattern is a trend-confirmation configuration where the market gives a continuation signal after an unfilled gap. Before I go any further, you need to understand what gaps are.

Gaps form an important part of price action. They vary in rareness from market to market. For instance, in the currencies market, they usually happen on the open of the retail market following the weekend or whenever there is a big announcement. In stocks, gaps are fairly common from one day to another.

A gap is a discontinuation or a hole between two successive close prices mainly caused by low liquidity. When a market is trading at $100 and suddenly trades at $102 without ever quoting at $101, it has formed a bullish gap. This can be seen in the charts where it appears to have a missing piece between candlesticks. Figure 4-8 illustrates a bullish gap. Notice the empty space between candlesticks.

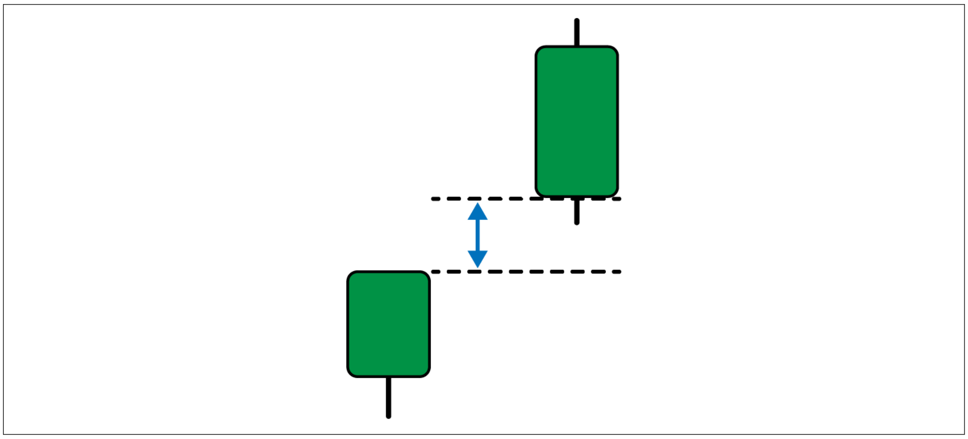


Figure 4-8. A bullish gap

Gaps can occur due to fundamental and technical reasons, but you should be interested in identifying and trading on them regardless of the reason for their appearance. In the currencies market, the visible gaps are the ones that occur over the weekend. Figure 4-9 illustrates a bearish gap.

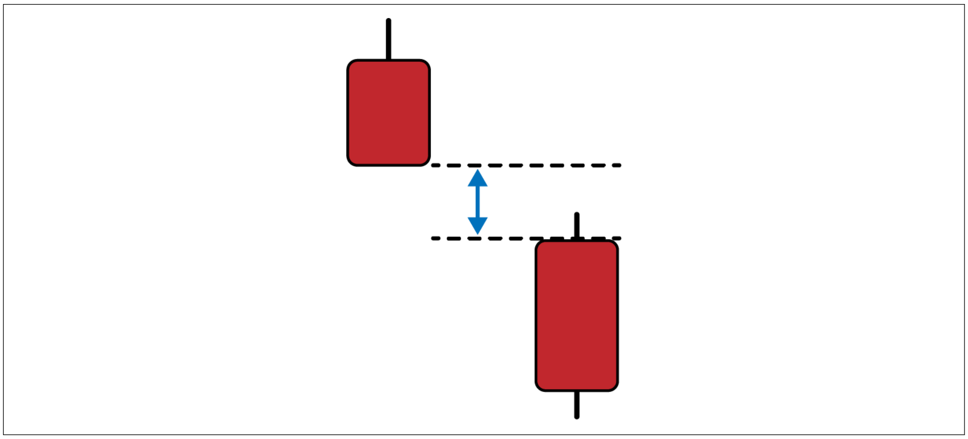


Figure 4-9. A bearish gap

There are different types of gaps, and categorizing them can be difficult when they appear due to hindsight bias (a cognitive bias that causes the analyst to overestimate the predictive power of a technique due to already knowing the outcome):

Common gaps
: These generally occur in a sideways market. They are likely to be filled because of the market's mean-reversion dynamic.

Breakaway gaps
: These generally resemble a common gap, but the gap occurs above a graphical resistance or below a graphical support. They signal acceleration in the new trend.

Runaway gaps
: These generally occur within the trend, but confirm it more; therefore, this is a continuation pattern.

Exhaustion gaps
: These generally occur at the end of a trend and close to a support or resistance level. It is a reversal pattern.

There is no sure way to determine the type of gap the moment it occurs, but that is not essential in detecting the Tasuki pattern. The pattern is named after an accessory used to hold the Japanese kimono dress, but why is unclear.

A bullish Tasuki pattern is composed of three candlesticks where the first one is a bullish candlestick, the second one is another bullish candlestick that gaps over the first candlestick, and the third candlestick is bearish but does not close below the close of the first candlestick.

Figure 4-10 illustrates a bullish Tasuki.

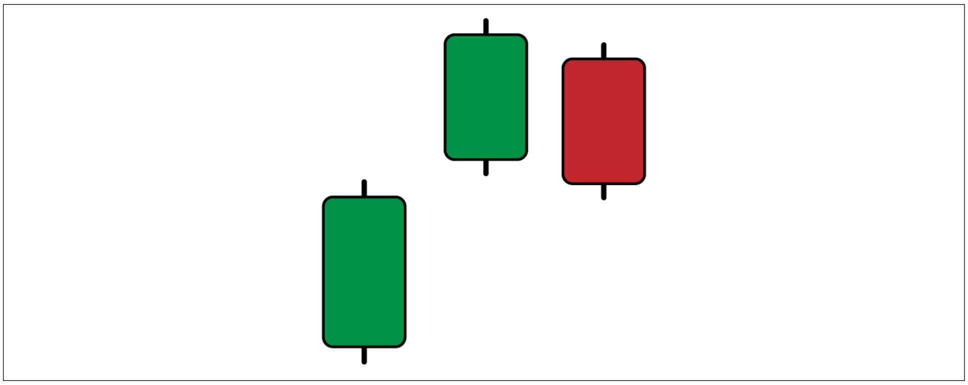


Figure 4-10. A bullish Tasuki

A bearish Tasuki pattern (see Figure 4-11) is the mirror image of the bullish Tasuki. It is composed of three candlesticks where the first one is a bearish candlestick, the second one is another bearish candlestick that gaps under the first candlestick, and the third candlestick is bullish but does not close above the close of the first candlestick.

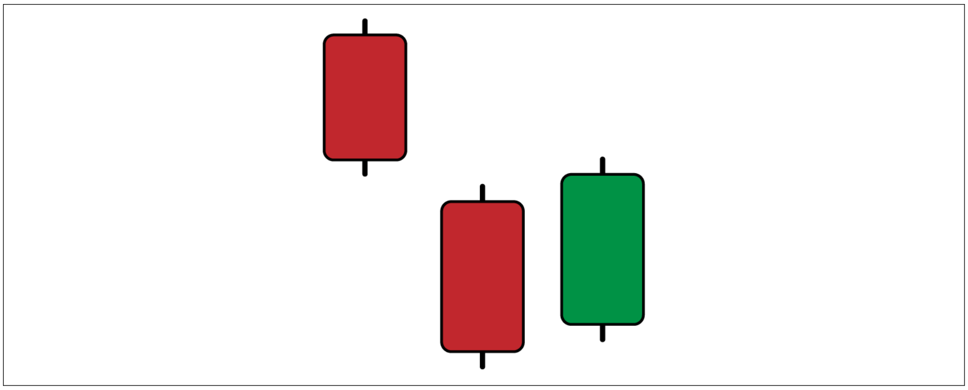


Figure 4-11. A bearish Tasuki

The intuition of the pattern relates to the breakout principle where breaching a certain threshold, be it support or resistance, should have a gravitational pull toward the threshold before releasing it to the initial direction.

Whenever you see a market gapping up followed by a bearish candlestick that does not quite close the gap, it may be a signal that the sellers are not strong enough to take the lead, giving a bullish bias. Similarly, whenever you see a market gapping down followed by a bullish candlestick that does not quite close the gap, it may be a signal that the buyers are not strong enough to take the lead, giving a bearish bias.

Algorithmically, the conditions need to be as follows:

- If the close price from two periods ago is greater than the open price from two periods ago, the open price from one period ago is greater than the close two periods ago, the close price from one period ago is greater than the open price from one period ago, and the current close price is greater than the close price two periods ago, print 1 in the next row of the column reserved for buy signals.
- If the close price from two periods ago is lower than the open price from two periods ago, the open price from one period ago is lower than the close two periods ago, the close price from one period ago is lower than the open price from one period ago, and the current close price is lower than the close price two periods ago, then print –1 in the next row of the column reserved for sell signals.

```python
def signal(data, open_column, close_column, buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] < data[i, open_column] and \
              data[i, close_column] < data[i - 1, open_column] and \
              data[i, close_column] > data[i - 2, close_column] and \
              data[i - 1, close_column] > data[i - 1, open_column] and \
              data[i - 1, open_column] > data[i - 2, close_column] and \
              data[i - 2, close_column] > data[i - 2, open_column]:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] > data[i, open_column] and \
                data[i, close_column] > data[i - 1, open_column] and \
                data[i, close_column] < data[i - 2, close_column] and \
                data[i - 1, close_column] < data[i - 1, open_column] and \
                data[i - 1, open_column] < data[i - 2, close_column] and \
                data[i - 2, close_column] < data[i - 2, open_column]:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

It is time to discuss the back-testing results (see Table 4-4). Keeping in mind that the pattern is quite rare, you should approach the results with a form of statistical doubt.

Table 4-4. Tasuki pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	58.06%	0.97	0.70	31
USDCHF	59.46%	1.92	1.31	37
GBPUSD	48.72%	0.93	0.98	39
USDCAD	40.00%	0.61	0.91	40
BTCUSD	53.57%	0.89	0.77	56
ETHUSD	50.00%	1.03	1.03	30
GOLD	51.49%	1.01	0.95	101
S&P500	36.36%	0.33	0.58	11
FTSE100	50.00%	1.56	1.56	14

The results confirm the initial hypothesis: the pattern is quite rare. For it to appear only 31 times over the course of around 10 years on the hourly values of EURUSD is extremely low.

The issue with rare patterns is that it is unclear how they made it into the literature, as there was not enough (and is not enough) data to analyze the results and the predictability of trades taken on the pattern.

The results are mostly mixed to negative with statistical robustness unlikely due to the low number of trades. The Tasuki pattern may have a logical intuition but lacks in terms of hard data.

The Three Methods Pattern

The *Three Methods* pattern is a complex configuration mainly composed of five candlesticks. The rising Three Methods pattern should occur in a bullish trend with the first candlestick being a big-bodied bullish one followed by three small-bodied bearish candlesticks typically contained within the range of the first candlestick. To confirm the pattern, one last big bullish candlestick must be printed with a close higher than the first candlestick's high. This is just like a bullish breakout of a small consolidation.

Figure 4-12 illustrates a rising Three Methods.

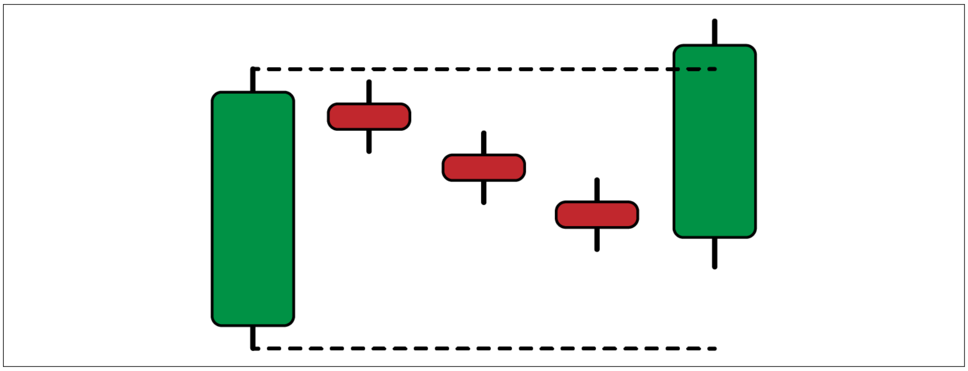


Figure 4-12. A rising Three Methods pattern

The falling Three Methods pattern should occur in a bearish trend with the first candlestick being a big-bodied bearish candlestick followed by three small-bodied bullish candlesticks typically contained within the range of the first candlestick. To confirm the pattern, one last big bearish candlestick must be printed with a close lower than the first candlestick's low. This is just like a bearish breakout of a small consolidation. Figure 4-13 illustrates a falling Three Methods.

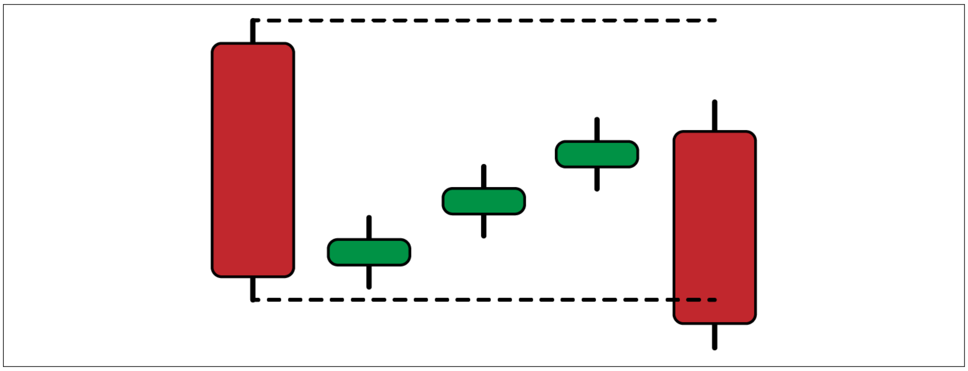


Figure 4-13. A falling Three Methods pattern

Psychologically speaking, the pattern relates to the concept of surpass or break as a confirmation of the initial move. Traders generally push the prices up until they begin to book their profits and thus close out of the positions. This activity has a smoothing effect on the price and should stabilize it through a phase called *correction* or *consolidation*.

In case the initial traders resume their buying or selling activity and manage to go beyond the range of consolidation, you can have a certain confidence that the move should continue.

The signal function for the Three Methods candlestick pattern can be written as follows:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] > data[i, open_column] and\
              data[i, close_column] > data[i - 4, high_column] and\
              data[i, low_column] < data[i - 1, low_column] and\
              data[i - 1, close_column] < data[i - 4, close_column] and\
              data[i - 1, low_column] > data[i - 4, low_column] and\
              data[i - 2, close_column] < data[i - 4, close_column] and\
              data[i - 2, low_column] > data[i - 4, low_column] and\
              data[i - 3, close_column] < data[i - 4, close_column] and\
              data[i - 3, low_column] > data[i - 4, low_column] and\
              data[i - 4, close_column] > data[i - 4, open_column]:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] < data[i, open_column] and\
                data[i, close_column] < data[i - 4, low_column] and\
                data[i, high_column] > data[i - 1, high_column] and\
                data[i - 1, close_column] > data[i - 4, close_column] and\
                data[i - 1, high_column] < data[i - 4, high_column] and\
                data[i - 2, close_column] > data[i - 4, close_column] and\
                data[i - 2, high_column] < data[i - 4, high_column] and\
                data[i - 3, close_column] > data[i - 4, close_column] and\
                data[i - 3, high_column] < data[i - 4, high_column] and\
                data[i - 4, close_column] < data[i - 4, open_column]:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 4-14 shows a signal chart on USDCAD.

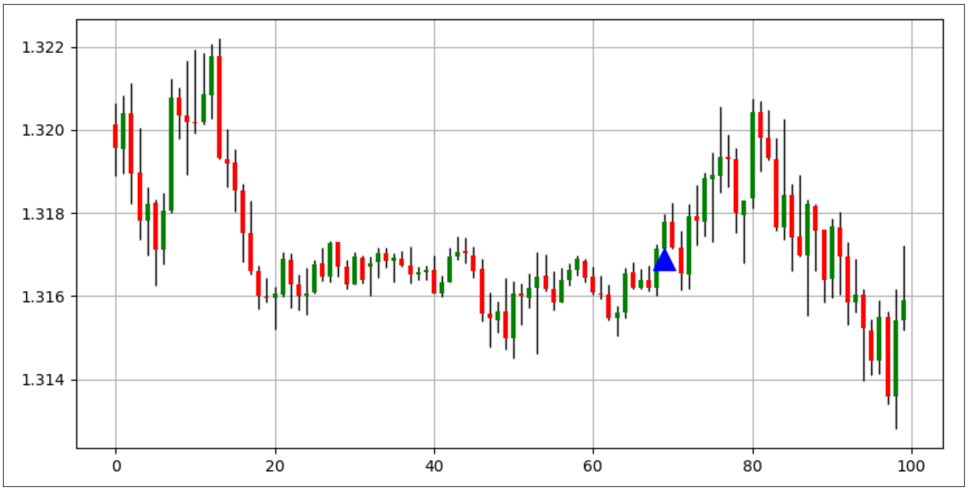


Figure 4-14. Signal chart on the USDCAD

The pattern is extremely rare and does not show much added value on its own. The results from the back-tests show very few signals generated even when making the conditions more flexible; therefore, this pattern remains mysterious.

Depending on your data provider, some historical values may not exactly be the same and thus, you can see past patterns validated on data coming from a certain provider but not on data coming from another provider. This typically explains the slight differences in back-testing results when switching providers and brokers.

The Hikkake Pattern

Hikkake is a Japanese verb that means "to trick" or "to trap," and this pattern refers to an actual trap. The pattern is complex, being composed of around five candlesticks depending on the literature (some research shows that it could be a combination of multiple candlesticks but does not specify the number of candlesticks).

The bullish Hikkake (shown in Figure 4-15) starts with a bullish candlestick followed by a bearish candlestick completely embedded inside the first one. Then, two candlesticks must appear with a high that does not surpass the second candle's high. Finally, a big bullish candlestick appears with a close that surpasses the high of the second candlestick. This serves as the validation of the pattern and the upside confirmation.

The bearish Hikkake starts with a bearish candlestick followed by a bullish candlestick completely embedded inside the first one. Then, two candlesticks must be printed with a low that does not surpass the second candle's low. Finally, a big bearish candlestick must be printed with a close that breaks the low of the second candlestick. This serves as the validation of the pattern and the downside confirmation.

The psychology of the pattern is not hard to grasp even though it is subjective in nature. The term *entrapment* comes from the fact that a bullish Hikkake is historically a bearish trap for traders thinking that the market has seen a resistance and should continue lower. Therefore, whenever you see the final candlestick breaching the highs and validating the configuration, you can have a certain confidence that more strength is to come. This is because stops are getting taken out and because traders are looking at a market that is surpassing its resistance, which is theoretically a strong bullish signal.

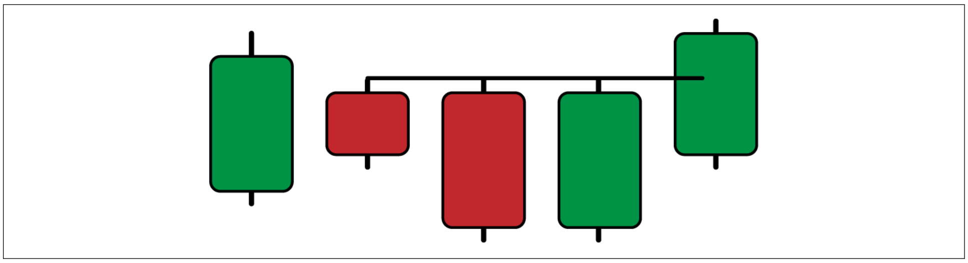


Figure 4-15. A bullish Hikkake

A bearish Hikkake (shown in Figure 4-16) can be compared to a bullish trap for traders who think that the market has seen a support and should now bounce. Therefore, whenever you see the final candlestick breaking the lows and validating the configuration, you can start changing the bias.

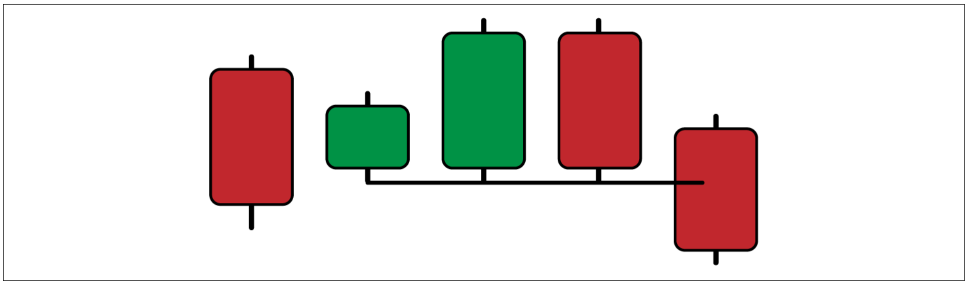


Figure 4-16. A bearish Hikkake

The signal function can be written as follows:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_signal, sell_signal):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

            # Bullish pattern
            if data[i, close_column] > data[i - 3, high_column] and \
               data[i, close_column] > data[i - 4, close_column] and \
               data[i - 1, low_column] < data[i, open_column] and \
               data[i - 1, close_column] < data[i, close_column] and \
               data[i - 1, high_column] <= data[i - 3, high_column] and \
               data[i - 2, low_column] < data[i, open_column] and \
               data[i - 2, close_column] < data[i, close_column] and \
               data[i - 2, high_column] <= data[i - 3, high_column] and \
               data[i - 3, high_column] < data[i - 4, high_column] and \
               data[i - 3, low_column] > data[i - 4, low_column] and \
               data[i - 4, close_column] > data[i - 4, open_column]:

                   data[i + 1, buy_signal] = 1

            # Bearish pattern
            elif data[i, close_column] < data[i - 3, low_column] and \
                 data[i, close_column] < data[i - 4, close_column] and \
                 data[i - 1, high_column] > data[i, open_column] and \
                 data[i - 1, close_column] > data[i, close_column] and \
                 data[i - 1, low_column] >= data[i - 3, low_column] and \
                 data[i - 2, high_column] > data[i, open_column] and \
                 data[i - 2, close_column] > data[i, close_column] and \
                 data[i - 2, low_column] >= data[i - 3, low_column] and \
                 data[i - 3, low_column] > data[i - 4, low_column] and \
                 data[i - 3, high_column] < data[i - 4, high_column] and \
                 data[i - 4, close_column] < data[i - 4, open_column]:

                     data[i + 1, sell_signal] = -1

       except IndexError:

            pass

    return data
```

Figure 4-17 shows a generated signal on EURGBP. It is clear that this pattern is rare and does not have meaningful back-testing results.

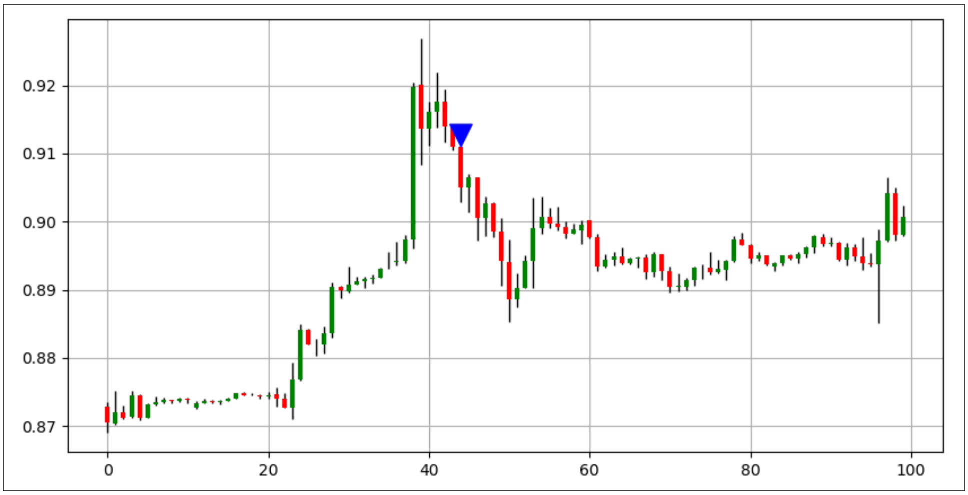


Figure 4-17. Signal chart on EURGBP

Because the pattern is quite rare, it does not have enough sufficient data to be evaluated properly. Surely, the conditions can be relaxed so that more signals appear, but theoretically, you would have to call it something else if you were to do that. Table 4-5 summarizes the performance.

Table 4-5. Hikkake pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	44.83%	0.94	1.16	116
USDCHF	47.66%	0.74	0.81	107
GBPUSD	49.57%	1.55	1.57	115
USDCAD	55.14%	0.84	0.68	107
BTCUSD	56.96%	1.18	0.89	79
ETHUSD	60.00%	1.24	0.82	50
GOLD	51.81%	1.14	1.06	166
S&P500	83.33%	6.01	1.20	12
FTSE100	53.33%	2.06	1.80	15

The results show random performance, which likely means that the pattern is not predictive. The hit ratio of 83.33% on the S&P 500 is unlikely to be statistically correct since the algorithm detected only 12 signals in the past few years.

CHAPTER 5

Modern Trend-Following Patterns

Continuing with the trend-following concept, this chapter reveals new patterns that are not part of the classic ones. I refer to these new patterns as *modern* since they are personal discoveries that I use in my analyses.

The aim remains the same, which is to create objective conditions and back-test them so that you form an opinion about their frequency and predictability. The important thing to remember is that a pattern's predictability is quite random from one market to the other, meaning that what works on GBPUSD may not work on EURGBP as every market has different statistical and technical properties.

Therefore, the main assumption I am making is that modern patterns are neither better nor worse than classic patterns; they are simply a diversification tool that allows you to get more confirmation in the analyses. This means that if you see at least two or three patterns (classic or modern) emerging around the same time, you should have a better conviction to take the trade. Let's now get started with these new patterns and explore their intuition and code.

The Quintuplets Pattern

The *Quintuplets* pattern is a multicandlestick configuration that confirms the underlying trend. This pattern was born out of the psychology of herding as well as the failure of reaction,[1] which gives it the extra push to continue in the same direction. It is characterized by having five successive small candles of the same type. The pattern relies on the gradual upward drift event where the market seems to be overvalued or overbought but remains unscathed and continues upward.

1 This event occurs when market participants are expecting a reversal due to extreme conditions but the market continues its initial state.

This type of pattern is typically seen in markets after an impactful event where the price action remains in a low-volatility environment but a clear trend can be seen. An example of this would be USDTRY[2] at the end of December 2021. Turkish president Recep Tayyip Erdoğan shook the markets with his statements on inflation and interest rates, which caused huge volatility, but then a deterministic form of indecision occurred where the USD steadily rose versus the TRY.

Figure 5-1 illustrates a bullish Quintuplets pattern.

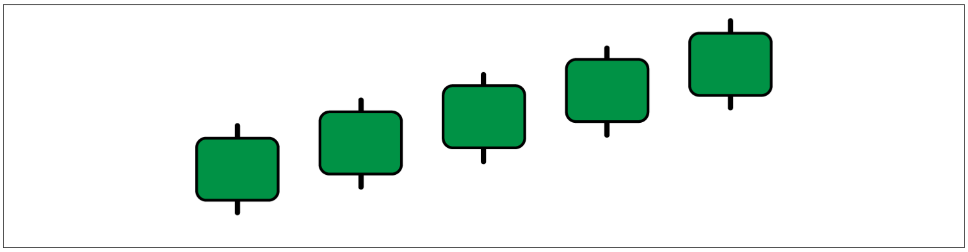


Figure 5-1. A bullish Quintuplets pattern

The Quintuplets pattern occurs when volatility slows down, as evidenced by the small-bodied bullish candles. Figure 5-2 shows a bearish Quintuplets pattern.

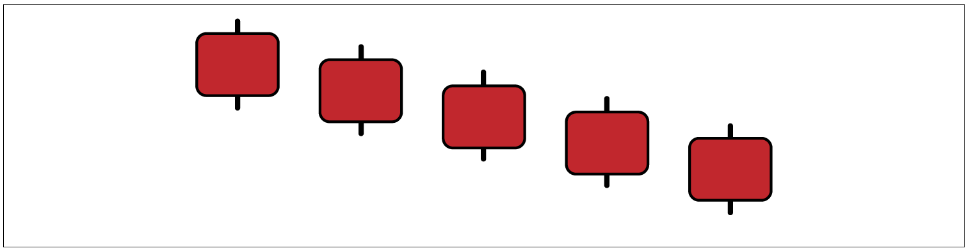


Figure 5-2. A bearish Quintuplets pattern

The difference between the Three Candles pattern discussed in Chapter 4 and the Quintuplets pattern is the number of candlesticks and their size. The former assumes that big moves are followed by big moves, while the latter assumes that slow and gradual moves are followed by slow and gradual moves as well. (This is in tandem with the volatility-clustering event in the financial markets. *Volatility clustering* is the persistence of volatility through time where low volatility is generally followed by low volatility while high volatility is followed by high volatility.)

2 USDTRY is the value of 1 USD in terms of the Turkish lira.

Hence, both patterns point to the same direction but deal with different market properties.

Detecting the Quintuplets pattern is easy, but the pattern can be uncommon since you have many conditions to fulfill. Algorithmically, the conditions need to be as follows:

- If the latest five close prices are each greater than their open prices, as well as the close prices preceding them, and each candlestick respects a maximum body size, then print 1 in the next buy row.
- If the latest five close prices are each lower than their open prices, as well as the close prices preceding them, and each candlestick respects a maximum body size, then print –1 in the next sell row.

Remember, the Quintuplets pattern is composed of five small candles. Therefore, you need to code the condition that they must have a maximum size.

The job now is to code the signal function as follows:

```python
def signal(data, open_column, close_column, buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column]>data[i, open_column] and\
              data[i, close_column]>data[i - 1, close_column] and\
              data[i, close_column]-data[i, open_column]<body and\
              data[i-1, close_column]>data[i-1, open_column] and\
              data[i-1, close_column]>data[i-2, close_column] and\
              data[i-1, close_column]-data[i-1, open_column]<body and\
              data[i-2, close_column]>data[i-2, open_column] and\
              data[i-2, close_column]>data[i-3, close_column] and\
              data[i-2, close_column]-data[i-2, open_column]<body and\
              data[i-3, close_column]>data[i-3, open_column] and\
              data[i-3, close_column]>data[i-4, close_column] and\
              data[i-3, close_column]-data[i-3, open_column]<body and\
              data[i-4, close_column]>data[i-4, open_column] and\
              data[i-4, close_column]-data[i-4, open_column]<body and\
              data[i, buy_column] == 0:

                    data[i + 1, 4] = 1
```

```
                # Bearish pattern
                elif  data[i, close_column]<data[i, open_column] and\
                      data[i, close_column]<data[i-1, close_column] and\
                      data[i, open_column]-data[i, close_column]<body and\
                      data[i-1, close_column]<data[i-1, open_column] and\
                      data[i-1, close_column]<data[i-2, close_column] and\
                      data[i-1, open_column]-data[i-1, close_column]<body and\
                      data[i-2, close_column]<data[i-2, open_column] and\
                      data[i-2, close_column]<data[i-3, close_column] and\
                      data[i-2, open_column]-data[i-2, close_column]<body and\
                      data[i-3, close_column]<data[i-3, open_column] and\
                      data[i-3, close_column]<data[i-4, close_column] and\
                      data[i-3, open_column]-data[i-3, close_column]<body and\
                      data[i-4, close_column]<data[i-4, open_column] and\
                      data[i-4, open_column]-data[i-4, close_column]<body and\
                      data[i, sell_column] == 0:

                        data[i + 1, 5] = -1

            except IndexError:

                    pass

        return data
```

Similar to the Three Candles pattern, Table 5-1 shows the default values for the variable body when using the hourly time frame.

Table 5-1. Quintuplets pattern: candle size choice

Asset	Body	Type
EURUSD	0.0005	Pip
USDCHF	0.0005	Pip
GBPUSD	0.0005	Pip
USDCAD	0.0005	Pip
BTCUSD	50	USD
ETHUSD	10	USD
GOLD	2	USD
S&P500	10	Points
FTSE100	10	Points

Figure 5-3 shows the trades generated on ETHUSD following the signals given by the function. The Quintuplets are validated at the close, which is why the signals are given on the open of the next candlestick.

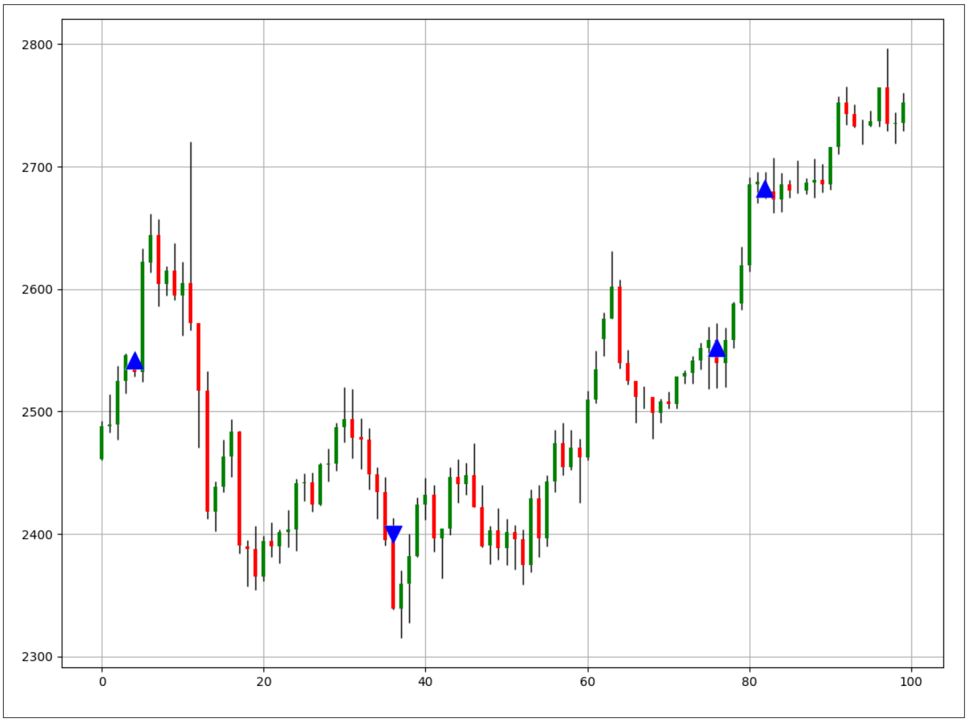


Figure 5-3. Signal chart on ETHUSD

The upward-pointing arrows refer to bullish signals generated on the open of the candlestick, whereas the downward-pointing arrows refer to bearish signals also generated on the open of the candlestick.

Figure 5-4 shows another example of the signals silver. You can see that whenever the conditions remain valid, you may get successive Quintuplets, but it does not necessarily say anything about the reinforcement of the conviction for a trend continuation.

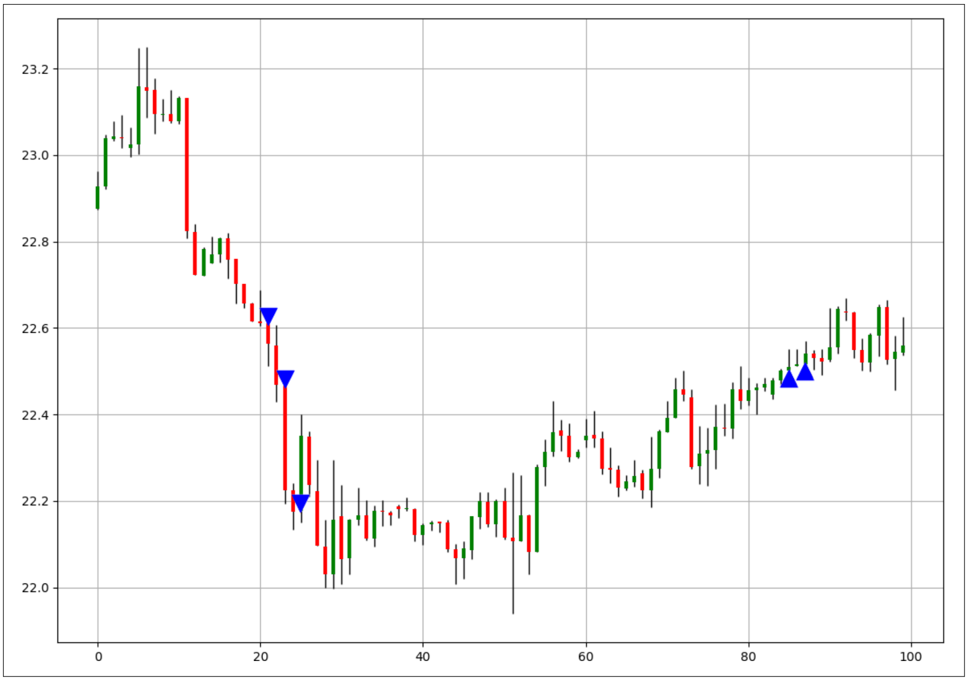


Figure 5-4. Signal chart on silver

Table 5-2 summarizes the performance of the Quintuplets pattern.

Table 5-2. Quintuplets pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	55.21%	0.93	0.76	2032
USDCHF	57.28%	0.98	0.73	2004
GBPUSD	58.08%	1.06	0.76	2052
USDCAD	59.04%	1.04	0.72	2146
BTCUSD	57.49%	1.09	0.81	854
ETHUSD	58.25%	0.93	0.67	800
GOLD	57.93%	1.16	0.84	1883
S&P500	59.44%	1.12	0.77	217
FTSE100	58.96%	1.05	0.73	212

The results are mixed and close to breakeven on multiple markets when accounting for the realized risk-reward ratio. Remember from "Coding Performance Evaluation Functions" on page 30, the hit ratio does not mean anything without the realized risk-reward ratio. This is why a 57.28% hit ratio on the USDCHF is not impressive since you are risking $1 to gain $0.76 in every trade. This is also seen through a profit factor that is less than 1.00.

The main conclusion is that the Quintuplets pattern should be used in combination with other techniques while optimizing entries and exits so as to obtain better metrics.

The Double Trouble Pattern

The *Double Trouble* pattern uses exogenous variables to be validated, meaning that you borrow information from a volatility indicator called the *average true range* (ATR) to validate the signal on the pattern.

Let's first define the concept of volatility and understand the ATR before introducing the Double Trouble pattern.

Volatility is one of the key concepts in trading and investing as it is directly related to risk and indirectly related to reward. A *volatile* asset or variable is one that has its returns fluctuate greatly around its mean value.

Figure 5-5 shows a low-volatility line (the less widely varying one) and a high-volatility line (the wildly swinging one).

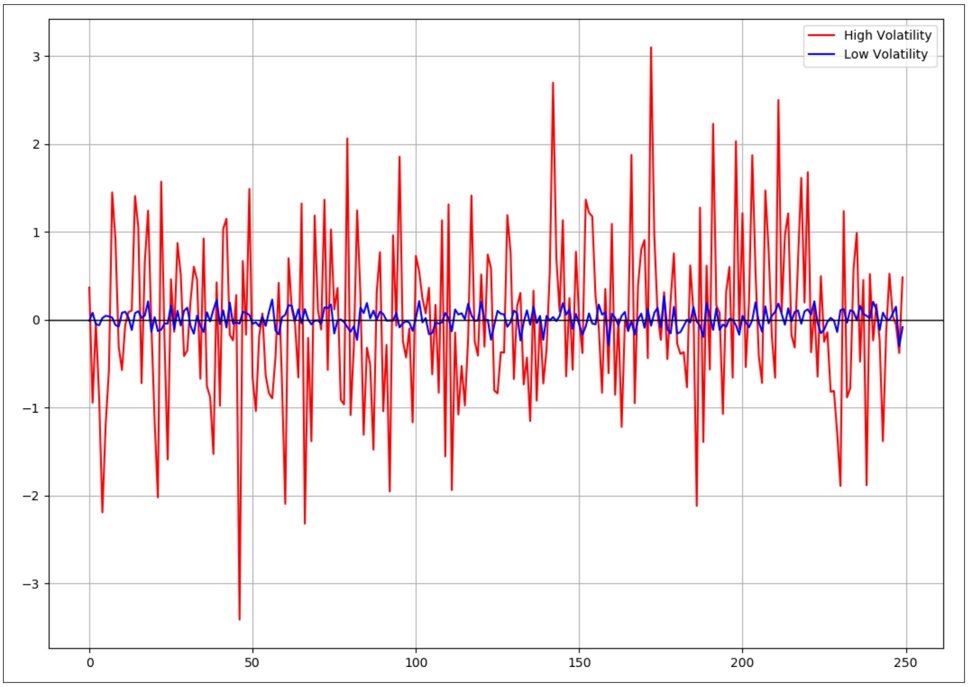


Figure 5-5. Comparison between high- and low-volatility variables

You can code it yourself with the following snippet:

```
# Importing the necessary libraries
import numpy as np
import matplotlib.pyplot as plt

# Creating high volatility noise
hv_noise = np.random.normal(0, 1, 250)

# Creating low volatility noise
lv_noise = np.random.normal(0, 0.1, 250)

# Plotting
plt.plot(hv_noise, color = 'red', label = 'High Volatility')
plt.plot(lv_noise, color = 'blue', label = 'Low Volatility')
plt.axhline(y = 0, color = 'black', linewidth = 1)
plt.grid()
plt.legend()
```

I have used a `numpy` function called `random.normal()`, which outputs random samples from a normal distribution. A normal distribution is a probabilistic continuous function with a mean that splits the data in a symmetric way. Data close to the mean tends to occur more frequently than data far from the mean. Visually, normally distributed data looks like a smooth bell with the middle value being the mean.

The different types of volatility can be summed up like so:

Historical volatility

This is the realized volatility over a certain period of time. Even though it is backward looking, historical volatility is used more often than not as an expectation of future volatility. One example of a historical measure is the standard deviation.

Before moving on, I want to mention a type of financial instruments called derivatives. *Derivatives* are products that traders use to trade markets in certain ways. For example, a *forward* contract is a derivative contract where a buyer locks in a price for an asset to buy it at a later time. A forward contract is an obligation. Another type of derivative is an option. An *option* is the right but not the obligation to buy a certain asset at a specific price in the future by paying a premium now (the option's price). When a buyer wants to buy the underlying stock, they exercise their option to do so; otherwise, they may let the option expire.

Implied volatility
: In its simplest definition, implied volatility is the measure that, when inputted into the Black-Scholes equation, gives out the option's market price. (The Black-Scholes equation is a mathematical formula used to price options. The option's price is the premium a buyer pays to buy the option, which gives them the right to buy a certain asset at a predetermined price before a certain expiration date.) It is the expected future actual volatility. It has one time scale: the option's expiration date.

Forward volatility
: The volatility over a specific period in the future.

Actual volatility
: The amount of volatility at any given time, also known as *local volatility*. This measure is hard to calculate and has no time scale.

The most basic type of volatility is the standard deviation. It is one of the pillars of descriptive statistics. First, I must explain variance.

Variance is a dispersion measure calculated as the squared deviations from the mean. You take the squared deviations, to force the distance from the mean to be non-negative, and then you take the square root of variance, thus turning it into the standard deviation and making the measure have the same units as the mean.

By squaring the deviations, you are avoiding a negative distance measure, and by taking the square root, you are comparing apples (standard deviation) to apples (mean).

Variance is therefore calculated through this mathematical formula:

$$\sigma^2 = \frac{1}{n}\Sigma_{i=1}^{n}\left(x_i - \chi\right)^2$$

Following this logic, standard deviation is therefore as follows:

$$\sigma = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(x_i - \chi\right)^2}$$

In layperson's terms, standard deviation is the average distance away from the mean that you expect to find when you analyze every value in the data set.

With the concept of volatility in mind, let's go over the volatility indicator that you must use to find the Double Trouble pattern, the ATR.

The ATR is another way to measure volatility. Similar to the standard deviation, the ATR takes into account the highs and the lows in its calculation, thus making it more complete than the standard deviation.

The ATR is used as a gauge of historical volatility. It was developed by Wilder Welles Jr., the creator of the RSI, an indicator discussed in Chapter 3. The first building block of the ATR is the *true range*. Let's see how to calculate the true range.

Suppose you have an OHLC array. For each hour, the true range is simply the greatest of the three price differences:

- High – Low
- High – Previous close
- Previous close – Low

Once you have gotten the maximum out of those, you simply take a smoothed average of a certain lookback period of the true ranges to get the ATR.

Since in periods of panic and price depreciation you see volatility go up, the ATR will most likely trend higher during these periods. Similarly, in times of steady uptrends or downtrends, the ATR will tend to go lower. Figure 5-6 shows an example of EURUSD with the 10-period ATR measure. What can you say by looking at the graph?

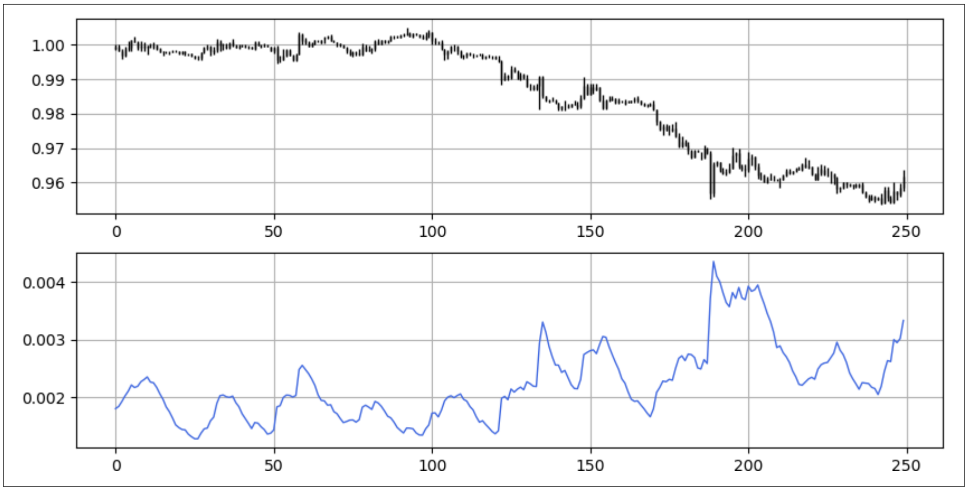


Figure 5-6. EURUSD with the 10-period ATR

Obviously, there is a negative correlation with the market and its ATR measure. This is quite common, as high volatility is often associated with fear (a falling market), whereas low volatility is often associated with greed (a rising market). The correlation is not perfect, but over the long term it is statistically negative.

The code for the ATR can be found in the following code snippet.

Make sure you have already defined the `smoothed_ma()` function from Chapter 3. The ATR is a smoothed average of the true ranges for a certain lookback period. Therefore, it is not a simple average.

```
def atr(data, lookback, high_column, low_column, close_column, position):

    data = add_column(data, 1)

    for i in range(len(data)):

        try:

            data[i, position] = max(data[i, high_column] - \
                                    data[i, low_column], abs(data[i, \
                                    high_column] - data[i - 1, close_column]),\
                                    abs(data[i, low_column]  - \
                                    data[i - 1, close_column]))

        except ValueError:

            pass

    data[0, position] = 0

    data = smoothed_ma(data, 2, lookback, position, position + 1)

    data = delete_column(data, position, 1)

    data = delete_row(data, lookback)

    return data
```

Now you are ready to learn about the *Double Trouble* pattern, a two-candle trend-following configuration that often signals a continuation in the aggregate direction.

The bullish Double Trouble pattern is composed of two bullish candles with the first close price lower than the second close price. The second candlestick must be at least double the size (from high to low) of the previous candlestick's 10-period ATR.

The bullish Double Trouble is based on the market psychology of euphoria. Another explanation is that it can be a form of a short squeeze, which can further increase the probability of a bullish exit.

A *short squeeze* is an abnormal situation characterized by rapidly rising prices in a market. The market must have a huge number of short sellers before the short squeeze. The rapid price increase is magnified by the trigger of short sellers' stop-loss orders.

Figure 5-7 illustrates a bullish Double Trouble pattern.

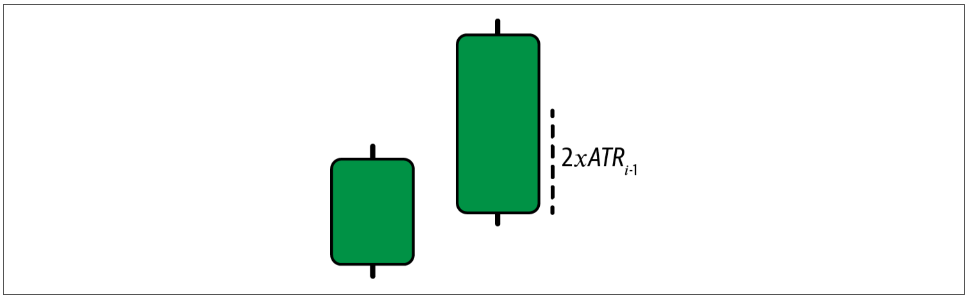


Figure 5-7. A bullish Double Trouble pattern

The bearish Double Trouble pattern is composed of two bearish candles with the first close price higher than the second close price. The second candlestick must also be at least double the size (from high to low) of the previous candlestick 10-period ATR.

Figure 5-8 illustrates a bearish Double Trouble pattern.

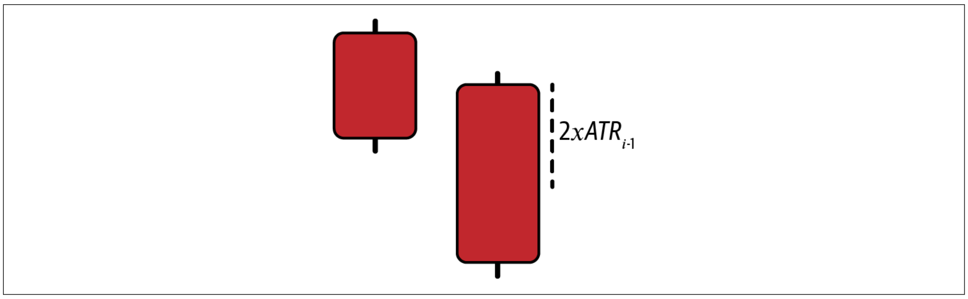


Figure 5-8. A bearish Double Trouble pattern

It is clear now that the Double Trouble pattern highly depends on volatility to appear. The validation of the pattern can occur only when the distance between the current high and the current low is two times greater than the 10-period ATR of the previous hour.

Algorithmically speaking, the code for the Double Trouble pattern is as follows:

```python
def signal(data, open_column, high_column, low_column, close_column,
           atr_column, buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] > data[i, open_column] and \
              data[i, close_column] > data[i - 1, close_column] and \
              data[i - 1, close_column] > data[i - 1, open_column] and \
              data[i, high_column] - data[i, low_column] > (2 * data[i - \
              1, atr_column]) and data[i, close_column] - data[i, \
              open_column] > data[i - 1, close_column] - data[i - 1, \
              open_column] and data[i, buy_column] == 0:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] < data[i, open_column] and \
              data[i, close_column] < data[i - 1, close_column] and \
              data[i - 1, close_column] < data[i - 1, open_column] and \
              data[i, high_column] - data[i, low_column] > (2 * data[i - \
              1, atr_column]) and data[i, open_column] - data[i, \
              close_column] > data[i - 1, open_column] - data[i - 1, \
              close_column] and data[i, sell_column] == 0:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

The signal function uses the `abs()` function, which takes the absolute values of the numbers inside the brackets. This is used to measure the size of a candle.

By calculating the ATR, you are using the fifth column (index = 4). Therefore, the buy and sell signals of the Double Trouble pattern must be in the sixth and seventh columns (index = 5 and index = 6) as opposed to the previous patterns where no intermediary indicator has been calculated. Make sure to shift the columns before you call the functions.

Figures 5-9 and 5-10 show a signal chart on the FTSE 100 and on BTCUSD.

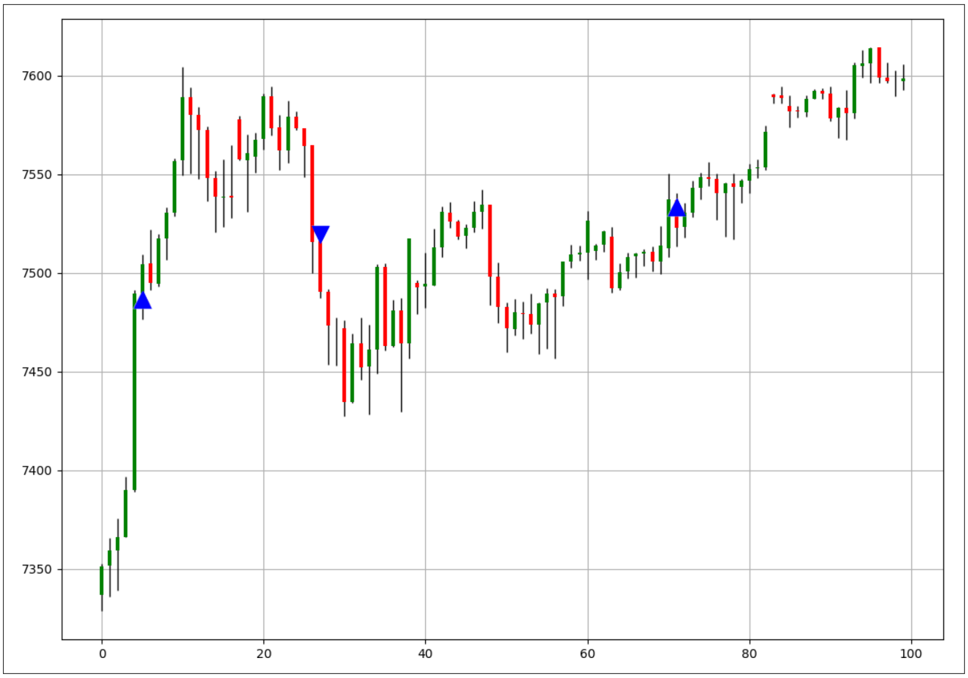


Figure 5-9. Signal chart on the FTSE 100

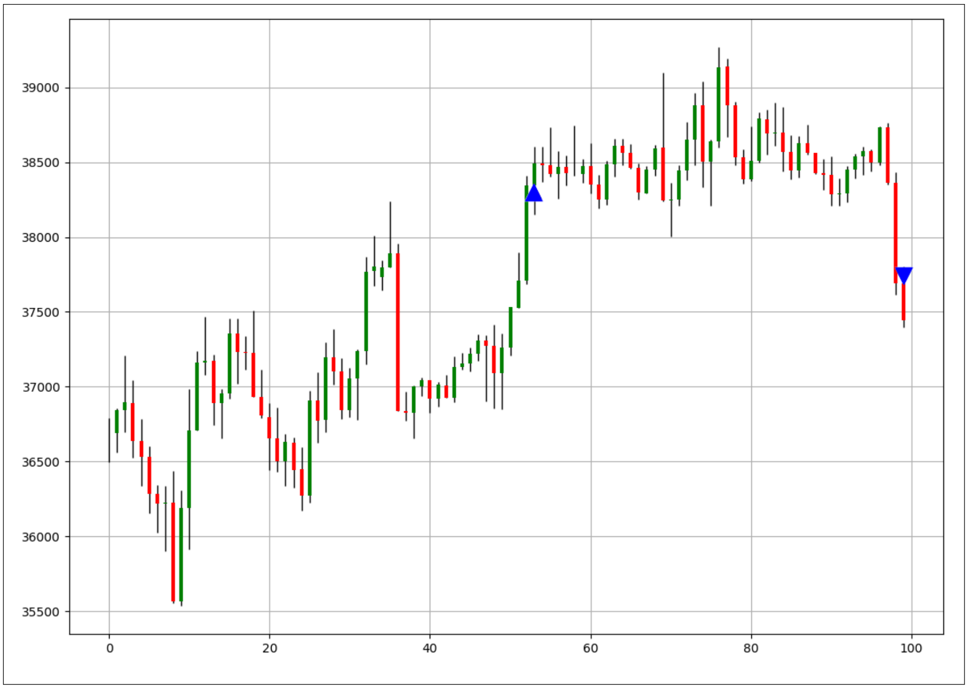


Figure 5-10. Signal chart on BTCUSD

Table 5-3 summarizes the performance of the Double Trouble pattern.

Table 5-3. Double Trouble pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	51.54%	1.03	0.97	2103
USDCHF	50.47%	1.03	1.01	2106
GBPUSD	50.06%	0.98	0.98	2155
USDCAD	50.86%	1.00	0.96	1787
BTCUSD	53.59%	1.21	1.05	1056
ETHUSD	56.76%	1.30	0.99	953
GOLD	51.76%	1.10	1.02	2007
S&P500	54.00%	1.32	1.13	237
FTSE100	45.95%	0.62	0.73	198

You can see that the Double Trouble pattern outperforms the Quintuplets pattern since it has relatively better profit factors and risk-reward ratios.

These results show that the Double Trouble pattern may be more predictive than the Quintuplets pattern. This also suggests that taking into account volatility can offer new insights and opportunities in the world of pattern recognition.

The Bottle Pattern

This is one of the simplest and most intuitive patterns I have discovered. It relies on two candlesticks to signal a continuation of the trend. I have named it the *Bottle* pattern because the second candle looks like a Bottle. This pattern also marks the return of the gap as it is one of the conditions needed to validate a bullish or bearish signal.

The bullish Bottle pattern is composed of a bullish candle followed by another bullish candle with no wick on the low side but with a wick on the high side. At the same time, the second candle must open below the last candle's close, which is considered a gap lower (or inside gap).

The pattern can be used to confirm upside continuations, as the underlying psychology is that the market has failed to make a new low after its opening, suggesting strong bullish pressure. Figure 5-11 illustrates the bullish Bottle.

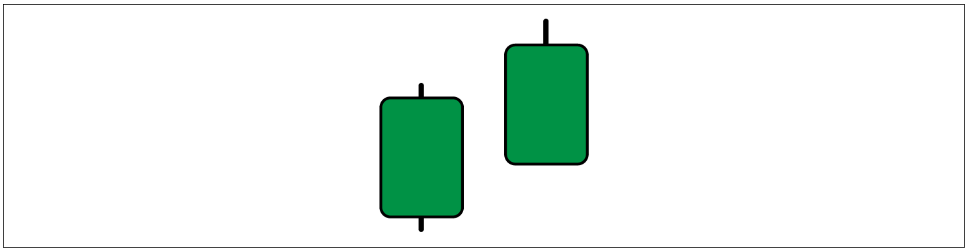


Figure 5-11. A bullish Bottle pattern

The bearish Bottle pattern is composed of a bearish candlestick followed by another bearish candlestick with no wick on the high side but with a wick on the low side. At the same time, the second candlestick must open above the last candlestick's close, which is considered a gap higher. Figure 5-12 illustrates the bearish Bottle.

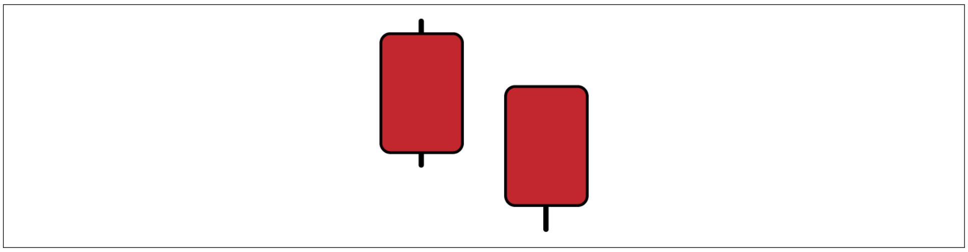


Figure 5-12. A bearish Bottle pattern

It is relatively easy to spot the configuration, but you must be careful to respect all the conditions as sometimes the gap may not occur, thus rendering the pattern invalid.

You can express the signal function for the pattern the following way:

```python
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] > data[i, open_column] and \
              data[i, open_column] == data[i, low_column] and \
              data[i - 1, close_column] > data[i - 1, open_column] and \
              data[i, open_column] < data[i - 1, close_column] and \
              data[i, buy_column] == 0:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] < data[i, open_column] and \
                data[i, open_column] == data[i, high_column] and \
                data[i - 1, close_column] < data[i - 1, open_column] and \
                data[i, open_column] > data[i - 1, close_column] and \
                data[i, sell_column] == 0:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

At this point, you should understand how the conditions are written and, therefore, the conditions should become second nature to you. Furthermore, the simplicity of this pattern makes the code basic.

Let's see the signals in action (Figure 5-13) before performing the back-testing and interpreting the results.

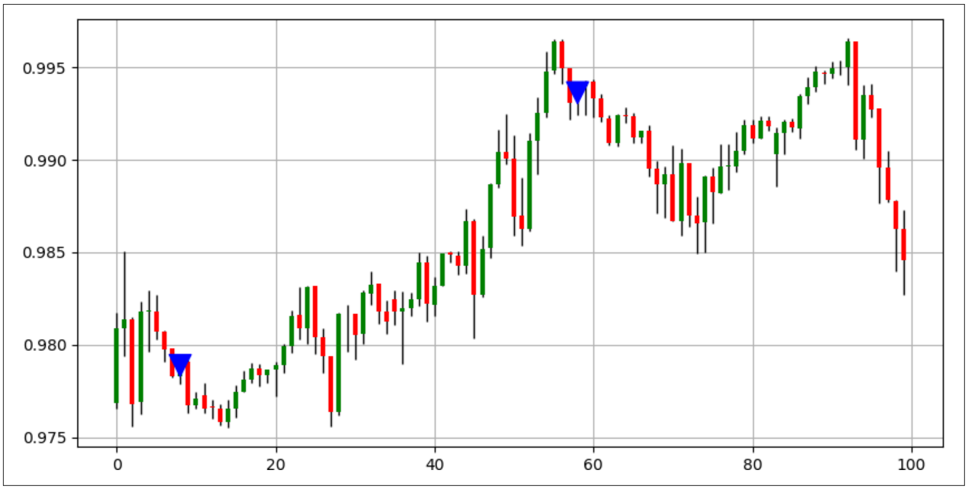


Figure 5-13. Signal chart on USDCHF

The first thing you notice is that over the last 100 hours, there was only two bearish Bottle patterns on USDCHF. This is normal and can occur from time to time, especially with a condition that imposes equality. Figure 5-14 shows AUDJPY where you can also see one bearish signal during the last 100 trading hours.

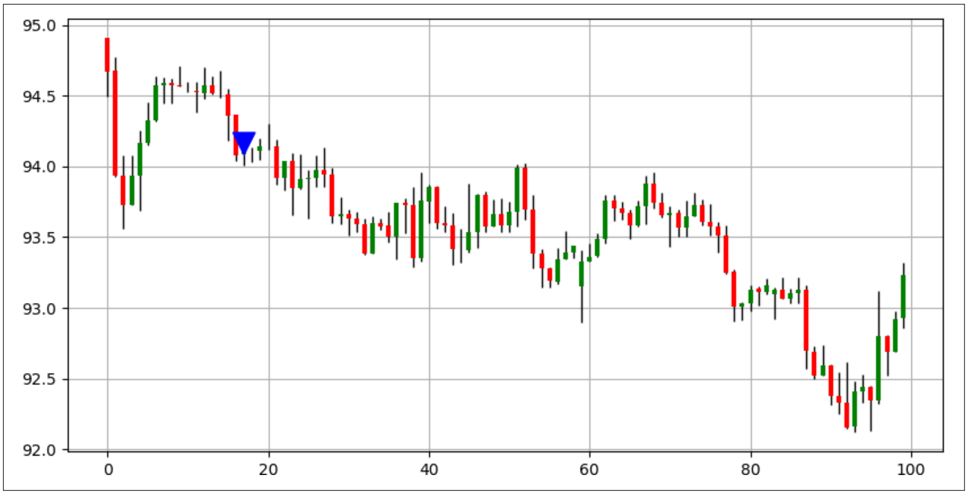


Figure 5-14. Signal chart on the AUDJPY

You can use the rounding function to increase the signal frequency of the Bottle pattern. I have chosen not to do it so as to keep the true essence of the pattern.

Table 5-4 summarizes the performance of the Bottle pattern.

Table 5-4. Bottle pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	51.39%	0.99	0.94	395
USDCHF	52.85%	1.17	1.04	507
GBPUSD	50.75%	0.85	0.83	398
USDCAD	50.48%	1.14	1.11	513
BTCUSD	50.44%	0.98	0.96	337
ETHUSD	50.95%	0.83	0.80	365
GOLD	47.98%	0.82	0.89	471
S&P500	41.81%	0.72	1.01	55
FTSE100	53.91%	1.81	1.55	115

The pattern seems to perform better on the USDCHF and the FTSE 100 than the other back-tested assets. It is important to know that patterns, whether good or bad ones, are unlikely to work on every market; therefore, when you find a good pattern, you must trade it on the right market.

The simplest example is the underperformance of reversal patterns on trending markets and the underperformance of continuation patterns on ranging markets. You need to keep in mind that you are looking for good techniques (patterns or strategies), not perfect ones.

The Slingshot Pattern

The name of this pattern comes from the fact it deals with a breakout system that confirms the new trend. Therefore, the *Slingshot* pattern is a four-candlestick trend detection system that uses the pull-back method after a breakout occurs. The definition may seem complicated, but in reality, it is very simple. Let's start by the conditions of the bullish Slingshot.

First, the pattern is characterized by having a first bullish candle followed by a higher one confirming a bullish bias, and then it has two more candlesticks, with the latter one not surpassing the high of the former. Finally, the last candlestick must have a low at or below the high of the first candlestick and a close higher than the high of the second candlestick. There is no strict rule about the color of the second and third candles. Figure 5-15 shows the perfect bullish Slingshot pattern.

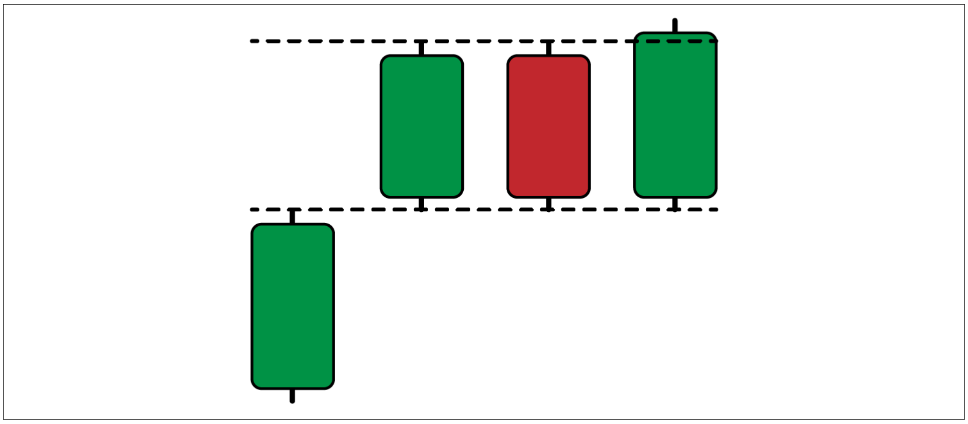


Figure 5-15. A bullish Slingshot pattern

The bearish Slingshot pattern (Figure 5-16) is characterized by having a first bearish candlestick followed by a lower one confirming a bearish bias, and then it has two more candlesticks, with the latter one not breaking the low of the former. Finally, the last candlestick must have a high at or above the low of the first candlestick and a close lower than the low of the second candlestick.

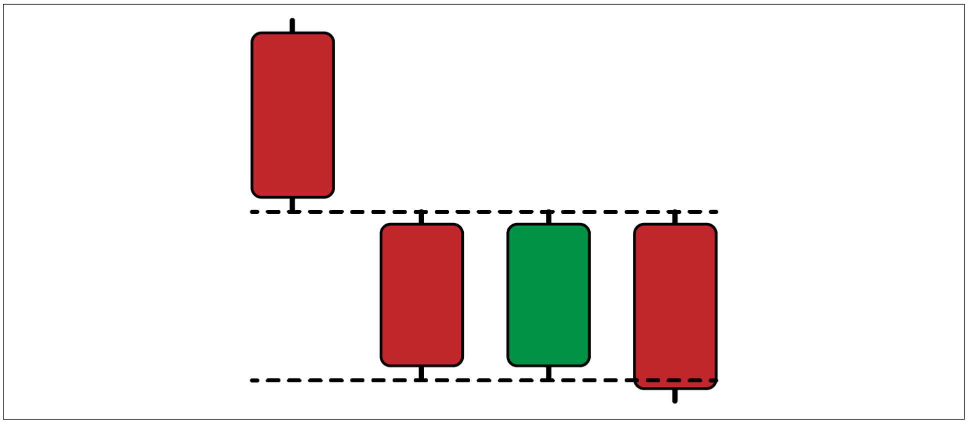


Figure 5-16. A bearish Slingshot pattern

Detecting the Slingshot pattern is a bit complex due to the conditions and its rarity but can be simplified by an algorithm, as in the following:

```python
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] > data[i - 1, high_column] and \
              data[i, close_column] > data[i - 2, high_column] and \
              data[i, low_column] <= data[i - 3, high_column] and \
              data[i, close_column] > data[i, open_column] and \
              data[i - 1, close_column] >= data[i - 3, high_column] and \
              data[i - 2, low_column] >= data[i - 3, low_column] and \
              data[i - 2, close_column] > data[i - 2, open_column] and \
              data[i - 2, close_column] > data[i - 3, high_column] and \
              data[i - 1, high_column] <= data[i - 2, high_column]:

                 data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] < data[i - 1, low_column] and \
                data[i, close_column] < data[i - 2, low_column] and \
                data[i, high_column] >= data[i - 3, low_column] and \
                data[i, close_column] < data[i, open_column] and \
                data[i - 1, high_column] <= data[i - 3, high_column] and \
                data[i - 2, close_column] <= data[i - 3, low_column] and \
                data[i - 2, close_column] < data[i - 2, open_column] and \
                data[i - 2, close_column] < data[i - 3, low_column] and \
                data[i - 1, low_column] >= data[i - 2, low_column]:

                 data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 5-17 shows an example of the gold commodity highlighting the signals given by the Slingshot pattern.

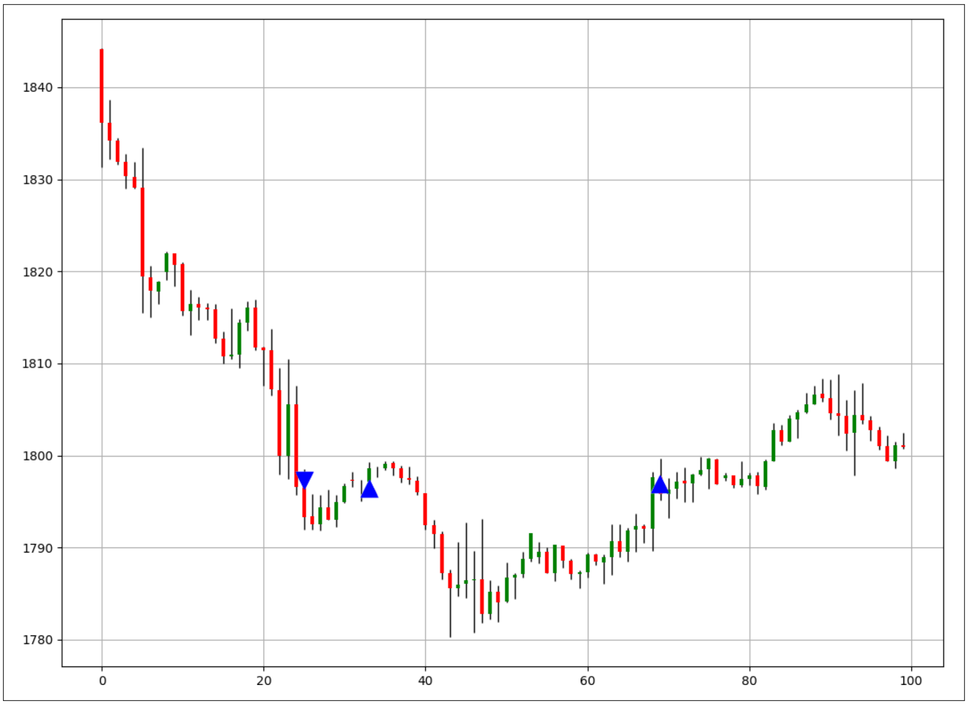


Figure 5-17. Signal chart on gold

The pattern is usually for short-term trend continuation moves; therefore, the aim is not to continue for days in the same direction but at least shape marginal new highs (bullish Slingshot) or new lows (bearish Slingshot).

Hence, exit strategies can greatly impact the results of this pattern. Nevertheless, I back-test it using the same conditions as the rest of the patterns (by exiting upon the next signal whether bullish or bearish).

In reality, you must create your own entry and exit rules based on risk and reward preferences so that you optimize the expected results. Figure 5-18 shows an example for AUDNZD.

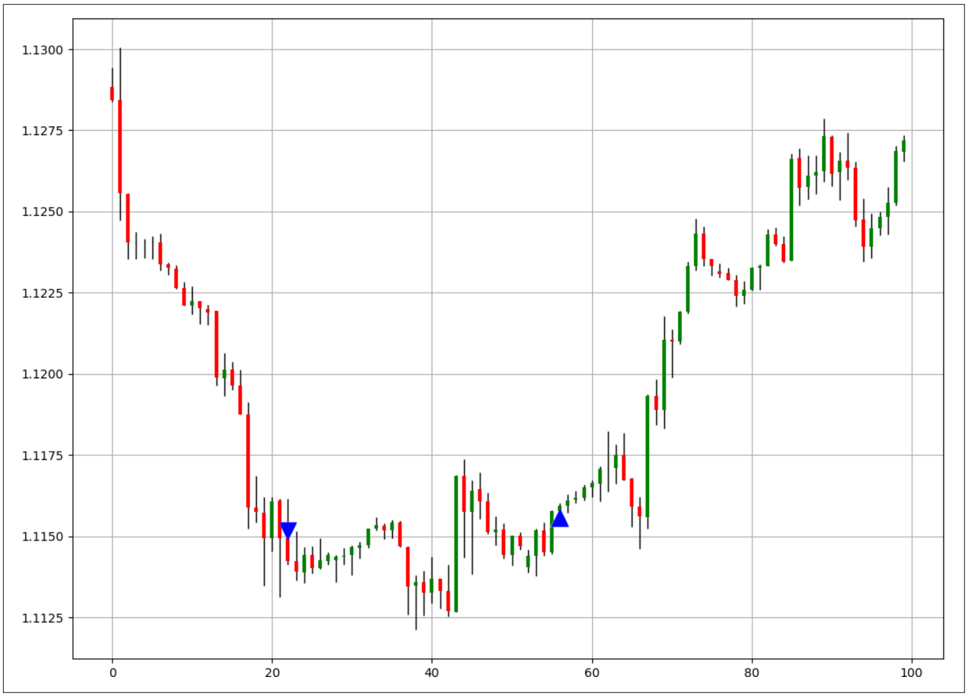


Figure 5-18. Signal chart on AUDNZD

Table 5-5 summarizes the results of the Slingshot pattern.

Table 5-5. Slingshot pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	50.03%	0.98	0.98	1591
USDCHF	51.81%	0.96	0.90	1621
GBPUSD	48.27%	1.01	1.08	1622
USDCAD	48.61%	0.94	0.99	1728
BTCUSD	52.01%	0.88	0.82	721
ETHUSD	48.52%	1.01	1.07	577
GOLD	47.90%	0.85	0.93	1332
S&P500	49.36%	1.01	1.04	158
FTSE100	57.14%	1.38	1.04	189

The Slingshot pattern has a relatively healthy number of signals when compared to other candlestick patterns. However, as it is greatly impacted by exit techniques (covered in subsequent chapters), its predictability is relatively low with the exception of the FTSE 100.

The H Pattern

The *H* pattern is a three-candle continuation configuration. The bullish H pattern (Figure 5-19) is composed of a bullish candle followed by an indecision candlestick[3] that has its open price equal to its close price. Next, the third candle must be bullish, and its close price must be higher than the indecision candle's close. Finally, the low of the third candle must be higher than the low of the indecision candle.

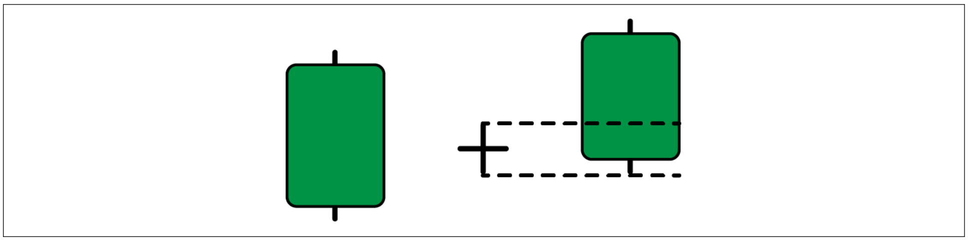


Figure 5-19. A bullish H pattern

The bearish H pattern (Figure 5-20) is composed of a bearish candle followed by an indecision candle; next, the third candle must be bearish, and its close must be lower than the indecision candle's close. Finally, the high of the third candle must be lower than the high of the indecision candle.

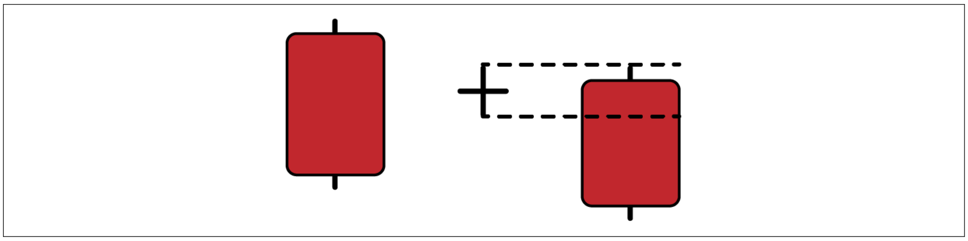


Figure 5-20. A bearish H pattern

3 The indecision pattern is also called a Doji.

The following code snippet shows how to scan for the H pattern:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] > data[i, open_column] and \
              data[i, close_column] > data[i - 1, close_column] and \
              data[i, low_column] > data[i - 1, low_column] and \
              data[i - 1, close_column] == data[i - 1, open_column] and \
              data[i - 2, close_column] > data[i - 2, open_column] and \
              data[i - 2, high_column] < data[i - 1, high_column]:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] < data[i, open_column] and \
                data[i, close_column] < data[i - 1, close_column] and \
                data[i, low_column] < data[i - 1, low_column] and \
                data[i - 1, close_column] == data[i - 1, open_column] and \
                data[i - 2, close_column] < data[i - 2, open_column] and \
                data[i - 2, low_column] > data[i - 1, low_column]:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 5-21 shows an example of a signal generated by the S&P 500. Visibly, the pattern may be rare depending on the market's characteristics. As is the case theoretically, trend-following patterns have on average better results on trending markets than on ranging markets.

With equities being mostly trending, the results shown in the performance summary table show that the H pattern outperforms on equity indices.

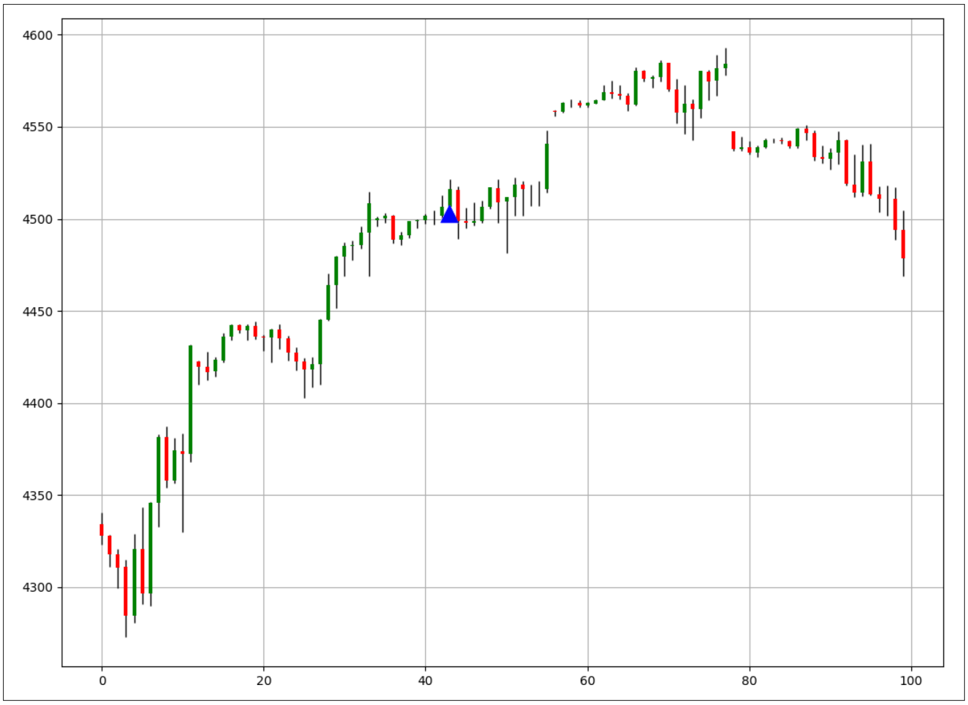


Figure 5-21. Signal chart on the S&P 500

Patterns generally have targets that are called *potential*. Up to this point, you have been back-testing them with the assumption of an exit at the next signal. This is fine because you want to harmonize the results across the patterns to have an apples-to-apples comparison.

With the H pattern and the other modern candlestick patterns presented in this book, I have found that there is not really an added value of imposing a certain exit as each market performs differently. Therefore, I have found that it is better to use your own judgment, apply the exits from your own strategy, or simply exit at the next signal if you are just trading on the pattern.

Figure 5-22 shows a signal generated on the hourly values of the AUDNZD pair.

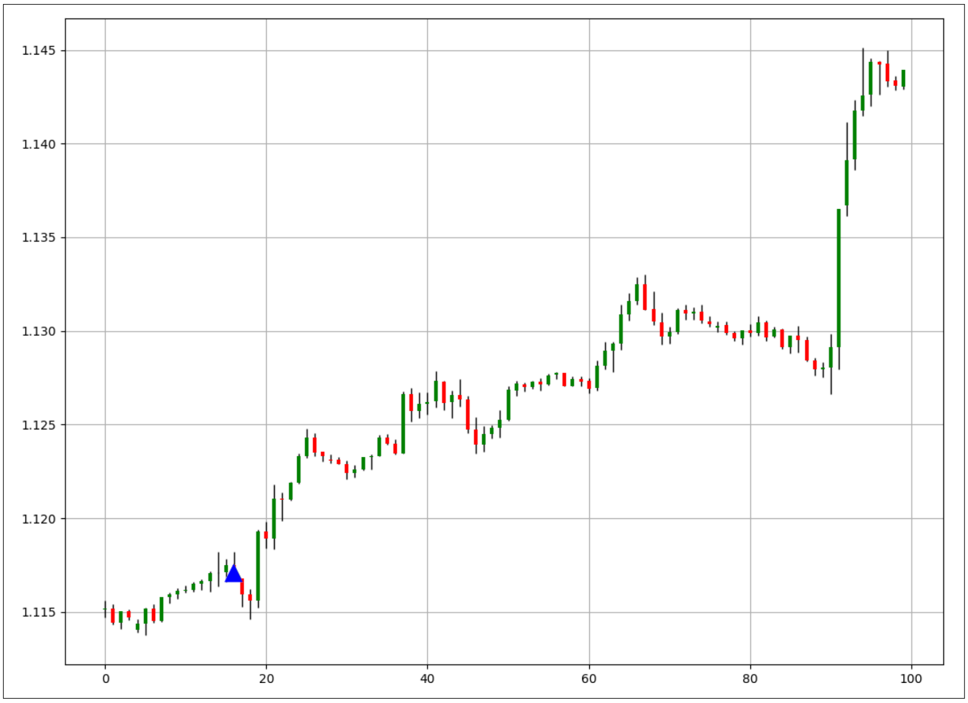


Figure 5-22. Signal chart on AUDNZD

Let's now take a look at the back-testing results and see how you can interpret them (see Table 5-6).

Table 5-6. H pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	51.56%	0.89	0.84	128
USDCHF	45.83%	0.77	0.91	144
GBPUSD	44.68%	0.93	1.15	94
USDCAD	57.49%	1.31	0.97	120
BTCUSD	61.53%	3.25	2.03	26
ETHUSD	41.83%	0.55	0.77	98
GOLD	65.38%	2.55	1.34	52
S&P500	59.64%	1.33	0.90	57
FTSE100	50.00%	1.35	1.35	36

There seems to be more predictive strength with this pattern, but the disadvantage is clear here: the frequency of signals. You can clearly see a lack of trades. As expected, you see better results on trending markets like the S&P 500 and the FTSE 100 than on ranging markets like the USDCHF and GBPUSD.

CHAPTER 6

Classic Contrarian Patterns

By now you should be familiar with the concept of trend following, the technique of riding an existing trend in the expectation that it persists. It is time to see the contrarian patterns that signal a reversal or at least a correction in the initial move. Therefore, contrarian patterns suggest a strategy of fading the trend as opposed to riding it.

The chapter covers the classic contrarian patterns. Remember that by *classic*, I mean the patterns that are already known and established in the world of technical analysis. The purpose of this chapter is to create the contrarian patterns' objective conditions and back-test them so that you can form an opinion about their frequency and predictability.

The Doji Pattern

Whenever someone mentions candlestick patterns, many think about the Doji pattern, as it is the most known and intuitive configuration. The word *doji* means "mistake" in Japanese, and this is logical as traders label it as an indecision pattern.

Like the Marubozu pattern, the *Doji* pattern is a one-candle configuration where the open price equals the close price. This is why Doji is an indecision pattern.

To distinguish a bullish Doji from a bearish one, you have to check the previous candlestick as well as the one after it. Figure 6-1 shows the theoretical bullish Doji configuration. In a falling market, the apparition of an indecision candle such as the Doji can be a first clue to a change of direction. It must be confirmed by a subsequent bullish candle.

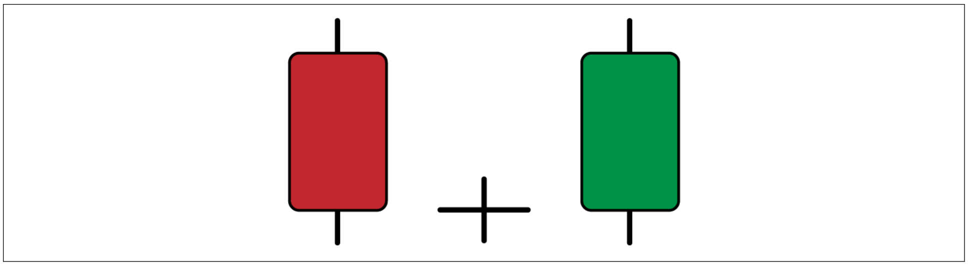


Figure 6-1. A bullish Doji

In comparison, Figure 6-2 shows the theoretical bearish Doji configuration. In a rising market, the apparition of the Doji signals a possible change in the current direction. Similarly, it must be confirmed by a subsequent bearish candle.

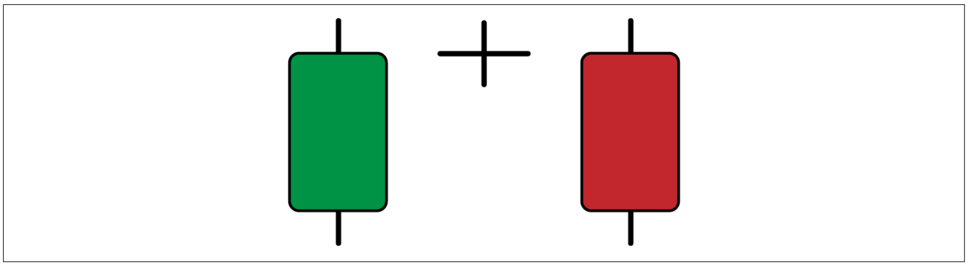


Figure 6-2. A bearish Doji

The rationale behind the Doji pattern lies within the concept of power balance. When there are more buyers than sellers, the balance of power is tilted upward as demand is greater than supply; thus, the market rises.

When demand starts losing momentum and approaches the supply level, an equilibrium occurs as evidenced by the loss of momentum in shaping new highs. An *equilibrium* occurs when a Doji pattern appears as a close price that equals an open price, showing that neither party has successfully pushed the price toward its desired direction.

In contrast, when supply is increasing relative to demand, the market should start dropping. The Doji pattern shapes the equilibrium point where the market switches from bullish to an expected bearish regime.

It is interesting to consider that there are other types of Doji patterns such as the following.

The Dragonfly Doji

This type of Doji has its high price equal to the close and open price. This means that it is a Doji where only the low price is different from the others. It is also an indecision pattern that can be either bullish or bearish depending on the prior price action. The issue with this pattern is that you can think of it in two ways: either the market has not made a higher high, thus failing to continue higher and giving a bearish bias, or the market failed to close lower and went back up to square one, giving a bullish bias. In any case, it must be treated as all Doji patterns are treated: as a reversal pattern.

Figure 6-3 illustrates a Dragonfly Doji.

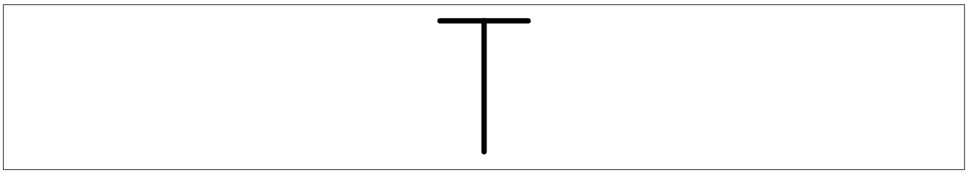


Figure 6-3. A Dragonfly Doji

The Gravestone Doji

This is the opposite of the Dragonfly Doji. It follows the same intuition and has the same expectation, which depends on the previous price action. In the case of a falling market, if you encounter a Gravestone Doji, then you must have a bullish bias. Similarly, if you have a rising market and you encounter a Gravestone Doji, then you must have a bearish bias.

Figure 6-4 illustrates a Gravestone Doji.

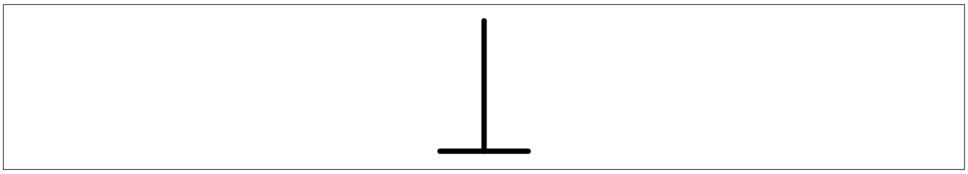


Figure 6-4. A Gravestone Doji

The Flat Doji

This type of pattern occurs generally when the retail markets are closed or when the market is trading close to the holidays. Low liquidity and volume are the main characteristics of the Flat Doji, and it is unlikely to indicate a directional. It remains however, a type of market indecision where the four prices equal each other. This is why it looks like a horizontal line.

Figure 6-5 illustrates a Flat Doji.

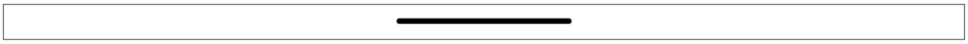


Figure 6-5. A Flat Doji

The Double Doji

Indecision can last for more than one period. It is possible that you might see more than one successive Doji pattern. Two successive Doji patterns signify prolonged indecision, and it is possible that the reversal's conviction is lowered as the contrarian traders are still hesitating and struggling to force the market their way. Generally, two Doji patterns are not worth more than one Doji.

Figure 6-6 illustrates a Double Doji.

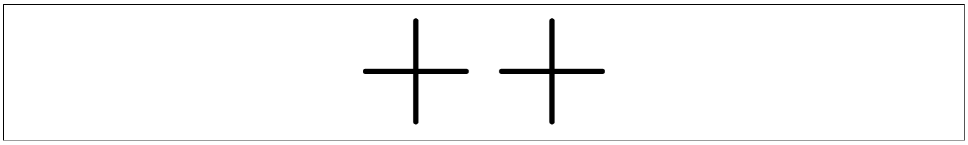


Figure 6-6. A Double Doji

The Tri Star Doji

This might be the rarest Doji pattern out of all the others. It is characterized by having three Doji patterns where the one in the middle gaps over the other two.

Figure 6-7 illustrates a Tri Star Doji.

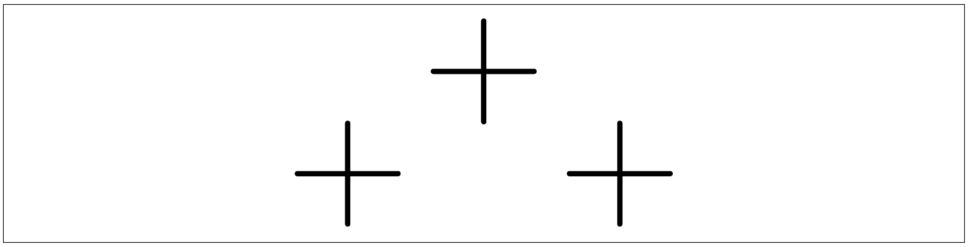


Figure 6-7. A Tri Star Doji

A Doji pattern is not perfect as it may occur multiple times with no market reaction afterward. That's because an equilibrium between the open price and the close price does not necessarily mean an equilibrium in the expected buying and selling activity.

Algorithmically, the conditions are not hard. You need to round the values as shown previously and then follow this intuition:

- If the current close price is greater than the current open price, the previous close price equals the open price, and the prior close price is lower than the prior open price, then print 1 in the next row.
- If the current close price is lower than the current open price, the previous close price equals the open price, and the prior close price is greater than the prior open price, then print –1 in the next row.

I have previously mentioned that the Doji pattern is a one-candle configuration. This is true because the Doji is the candle that has the same open and close prices. However, to confirm the pattern so that the algorithm predicts the next likely direction, you must impose the conditions previously mentioned.

You can code the signal function for the Doji pattern through the following code snippet:

```
def signal(data, open_column, close_column, buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] > data[i, open_column] and \
              data[i, close_column] > data[i - 1, close_column] and \
              data[i - 1, close_column] == data[i - 1, open_column] and \
              data[i - 2, close_column] < data[i - 2, open_column] and \
              data[i - 2, close_column] < data[i - 2, open_column]:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] < data[i, open_column] and \
              data[i, close_column] < data[i - 1, close_column] and \
              data[i - 1, close_column] == data[i - 1, open_column] and \
              data[i - 2, close_column] > data[i - 2, open_column] and \
              data[i - 2, close_column] > data[i - 2, open_column]:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 6-8 shows a signal chart on USDCHF.

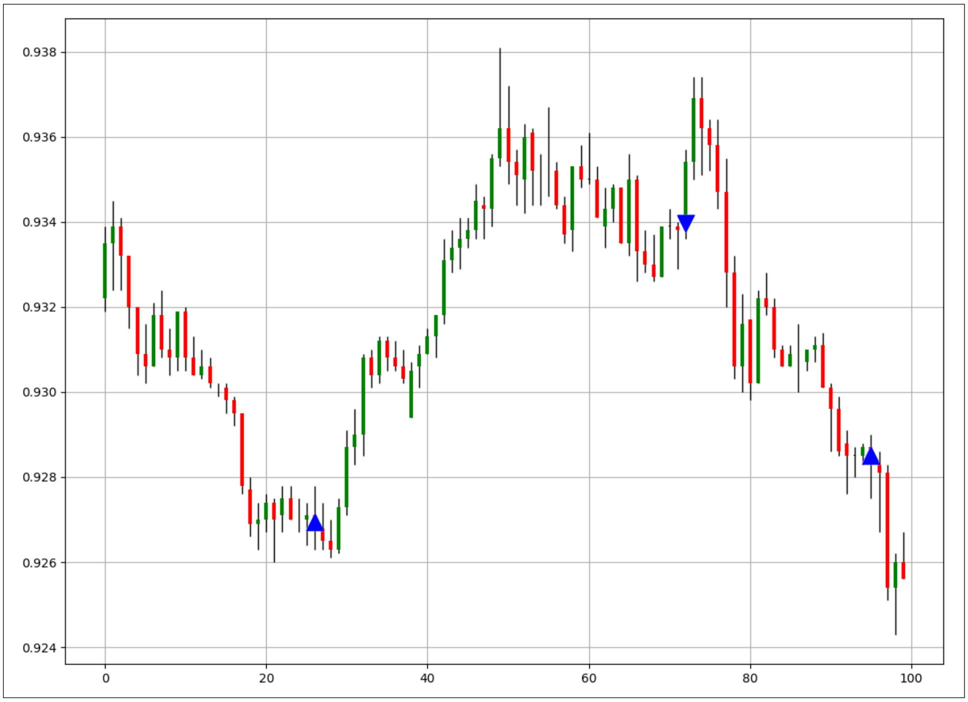


Figure 6-8. Signal chart on USDCHF

Obviously, the Doji pattern is not uncommon. Figure 6-9 shows another signal chart on the FTSE 100.

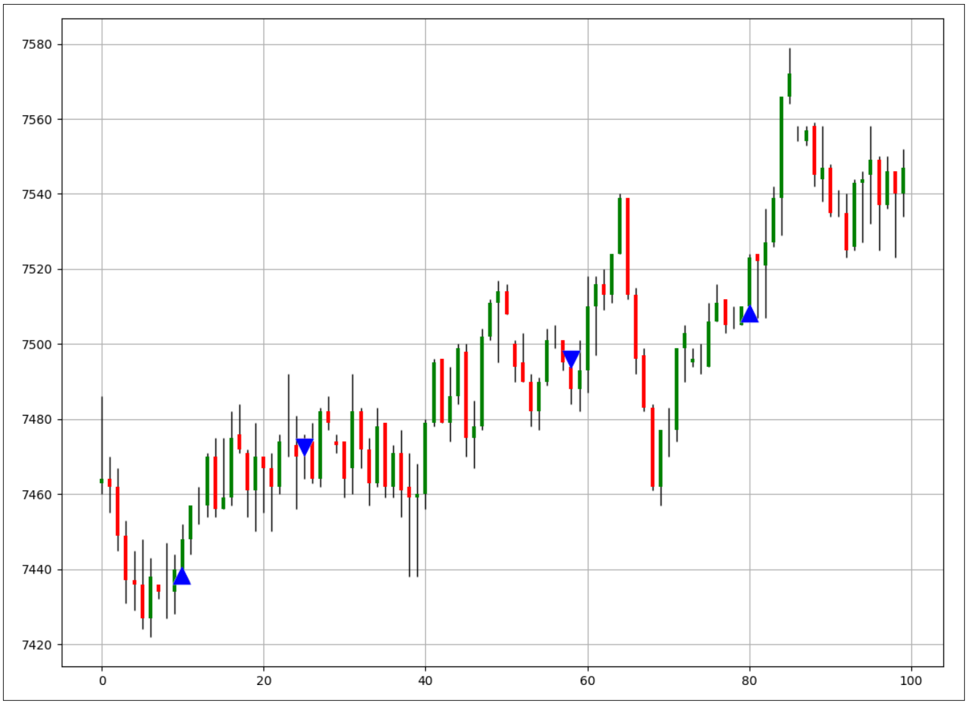


Figure 6-9. Signal chart on the FTSE 100

Table 6-1 shows the performance summary of the Doji pattern. Before checking the results, let's recap the trading conditions mentioned previously:

- The entry is done on the next open price after the validation of the signal on the current close price.
- The exit is done upon getting another signal in either direction (bullish or bearish).
- The transaction costs are omitted from the performance evaluation metrics.
- There is no risk management system.

Table 6-1. Doji pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	45.86%	0.97	1.14	1537
USDCHF	45.33%	0.83	1.00	1661
GBPUSD	49.78%	1.03	1.04	1177
USDCAD	48.06%	0.92	1.00	1315
BTCUSD	48.69%	0.93	0.98	345

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
ETHUSD	45.69%	0.91	1.09	1254
GOLD	46.17%	1.05	1.22	3166
S&P500	48.84%	1.11	1.16	303
FTSE100	50.27%	1.02	1.00	179

The back-tested assets show that the Doji pattern is a common configuration but generally not that predictive in nature. In my experience, it is essential to define the current market regime before relying on a Doji pattern for a future reaction. For example, in a bullish trending market, the bearish Doji pattern is unlikely to provide a signal of quality, but in a sideways market, the efficacy increases (remember the invisible hand).

To summarize, the Doji pattern is ground zero of technical pattern recognition due to the simplicity and abundance of the configuration but is far from being the most predictive pattern.

The Harami Pattern

Synonymous with the old word for *pregnant* in Japanese, the *Harami* pattern is a classic contrarian two-candle configuration where the body of the second candle is englobed by the body of the first candle. There are generally two types of Harami patterns, the strict one and the flexible one, and I discuss both in this chapter. Let's start with the flexible version.

The bullish Harami shows a first bearish candle (the mother) followed by a second bullish candle (the baby), which has its close price lower than the open price of the first candle and its open price higher than the close price of the first candle. The highs and lows of both candles are not relevant in the flexible version and are discussed in the strict version instead. Figure 6-10 illustrates a flexible bullish Harami.

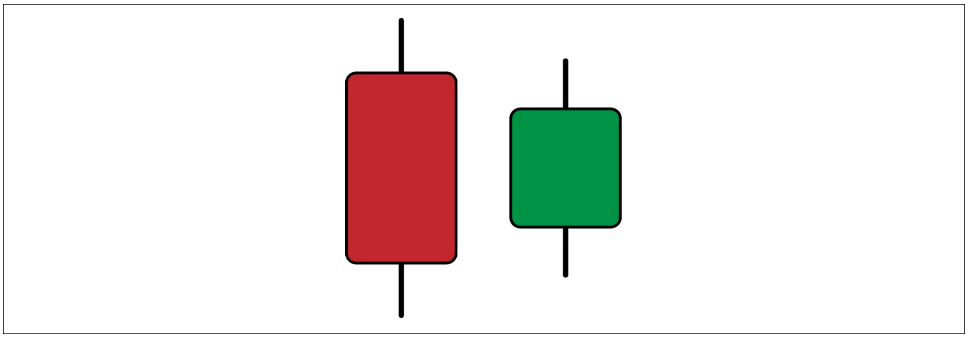


Figure 6-10. A bullish flexible Harami

The bearish Harami shows a first bullish candle (the mother) followed by a second bearish candle (the baby), which has its open price lower than the close price of the first candle and its close price higher than the open price of the first candle. Figure 6-11 illustrates a flexible bearish Harami.

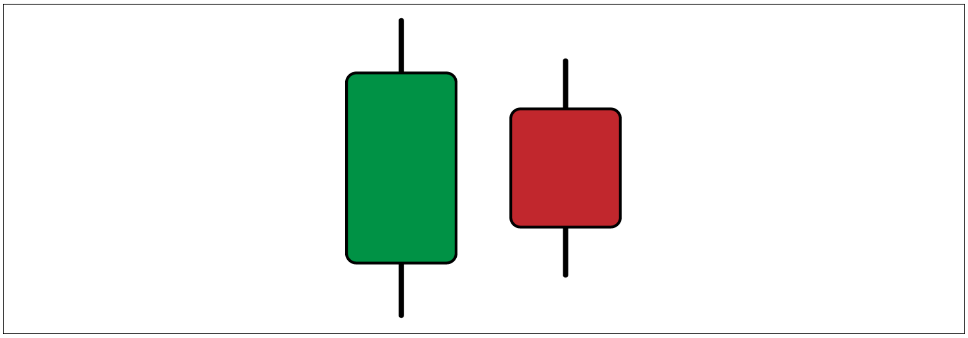


Figure 6-11. A bearish flexible Harami

The psychological explanation for the Harami pattern is that the decreasing candle size signals hesitation in the initial trend and that a reversal may be imminent. Generally, with classic patterns, there may be an aspect of self-fulfilling prophecy where a certain percentage of the predictability is due to the reputation of the pattern as opposed to its intuitive explanation.

Some patterns may not have a valid intuition but are nevertheless monitored by professional traders. This is because many other traders also monitor them, creating an infinite spiral that may increase their predictability.

Algorithmically, the conditions should follow this intuition:

- If the current close price is greater than the current open price and the previous close price, and the current open price is greater than the previous close price but lower than the previous open price, then print 1 in the next buy row.
- If the current close price is lower than the current open price and the previous close price, and the current open price is lower than the previous close price but greater than the previous open price, then print –1 in the next sell row.

The signal function for the flexible Harami pattern is as follows:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] < data[i - 1, open_column] and \
              data[i, open_column] > data[i - 1, close_column] and \
              data[i, high_column] < data[i - 1, high_column] and \
              data[i, low_column] > data[i - 1, low_column] and \
              data[i, close_column] > data[i, open_column] and \
              data[i - 1, close_column] < data[i - 1, open_column] and \
              data[i - 2, close_column] < data[i - 2, open_column]:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] > data[i - 1, open_column] and \
                data[i, open_column] < data[i - 1, close_column] and \
                data[i, high_column] < data[i - 1, high_column] and \
                data[i, low_column] > data[i - 1, low_column] and \
                data[i, close_column] < data[i, open_column] and \
                data[i - 1, close_column] > data[i - 1, open_column] and \
                data[i - 2, close_column] > data[i - 2, open_column]:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 6-12 shows a signal chart on ETHUSD where a bullish Harami appears in a minor bearish correction within an overall bullish trend.

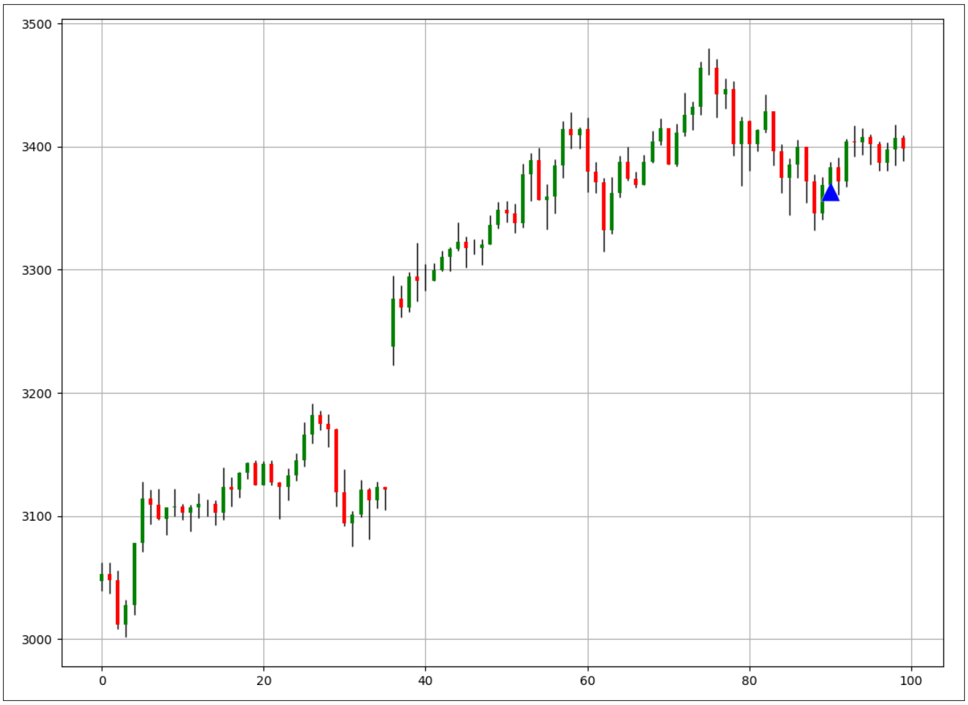


Figure 6-12. Signal chart on ETHUSD

Table 6-2 summarizes the performance of the flexible Harami pattern.

Table 6-2. Flexible Harami pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	50.06%	1.15	1.15	759
USDCHF	53.14%	1.12	0.99	668
GBPUSD	49.81%	1.06	1.06	823
USDCAD	52.15%	1.06	0.97	696
BTCUSD	48.19%	0.96	1.04	554
ETHUSD	48.81%	0.63	0.66	379
GOLD	48.73%	0.77	0.82	870
S&P500	53.75%	1.12	0.96	80
FTSE100	57.44%	1.42	1.05	94

Figure 6-13 shows a signal chart on GBPUSD with a bearish Harami appearing before a local top.

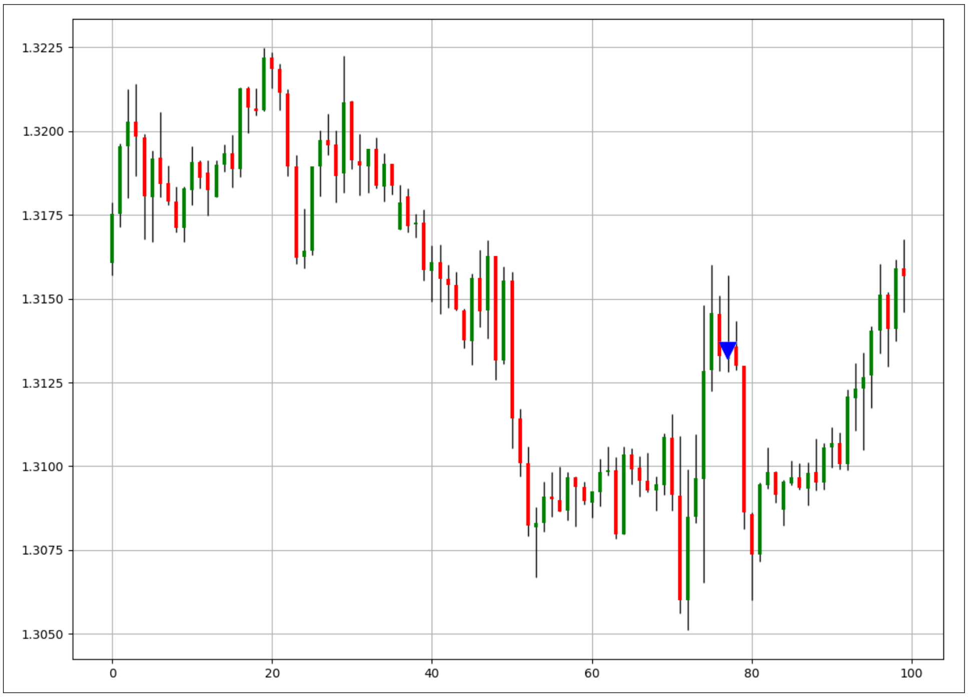


Figure 6-13. Signal chart on GBPUSD

Let's now move on to the strict Harami pattern. The bullish Harami shows a first bearish candle (the mother) followed by a second bullish candle (the baby), whose high price is lower than the open price of the first candle and whose low price is higher than the close price of the first candle. Figure 6-14 illustrates a bullish strict Harami.

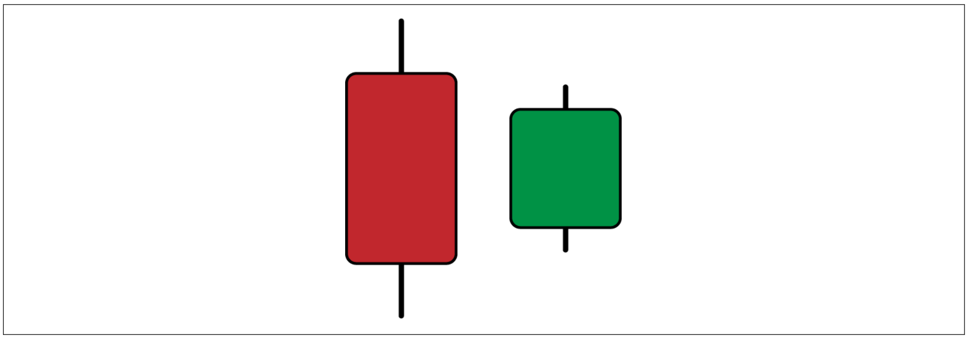


Figure 6-14. A bullish strict Harami

The bearish Harami shows a first bullish candle (the mother) followed by a second bearish candle (the baby), whose high price is lower than the close price of the first candle and whose low price is higher than the open price of the first candle. Figure 6-15 illustrates a bearish strict Harami.

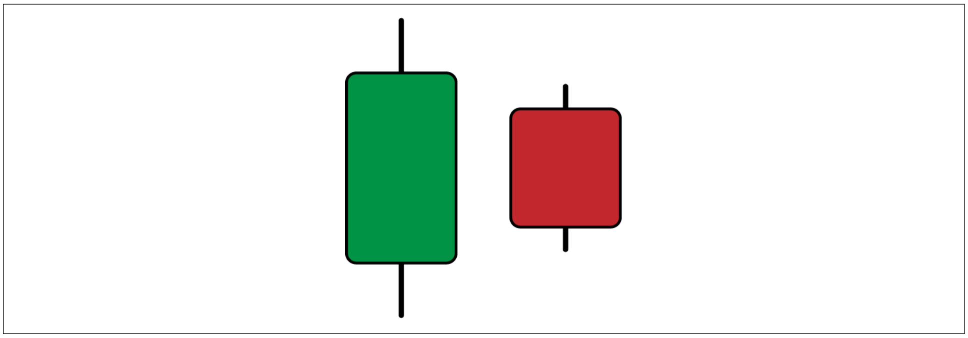


Figure 6-15. A bearish strict Harami

Algorithmically, the conditions should follow the following intuition:

- If the current candlestick is bullish and fully englobed within the previous bearish one, then print 1 in the next buy row.
- If the current candlestick is bearish and fully englobed within the previous bullish one, then print –1 in the next sell row.

The signal function for the strict Harami pattern is as follows:

```python
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] > data[i, open_column] and \
              data[i, high_column] < data[i - 1, open_column] and \
              data[i, low_column] > data[i - 1, close_column] and \
              data[i - 1, close_column] < data[i - 1, open_column] and \
              data[i - 2, close_column] < data[i - 2, open_column]:


                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] < data[i, open_column] and \
                data[i, high_column] < data[i - 1, close_column] and \
                   data[i, low_column] > data[i - 1, open_column] and \
                   data[i - 1, close_column] > data[i - 1, open_column] and \
                   data[i - 2, close_column] > data[i - 2, open_column]:

                       data[i + 1, sell_column] = -1

            except IndexError:

                pass

        return data
```

Because of the rarity of the strict Harami pattern, a back-test is not very informative. Data analysis requires a lot of output; therefore, a few signals are unlikely to lead to any robust conclusion. In practice, you should not commonly see the strict Harami as opposed to the flexible one.

To summarize, the Harami pattern is a two-candle configuration that can have two variations: flexible and strict. The former version is the most commonly used as it has enough data to evaluate it, while the latter suffers from extreme rarity due to the strenuous conditions imposed.

The On Neck Pattern

The *On Neck* pattern is a two-candle configuration that is easy to understand. The bullish On Neck is composed of a first bearish candlestick and a second bullish candlestick, which opens at a gap lower but closes exactly at the close price of the first candlestick. Figure 6-16 illustrates a bullish On Neck pattern.

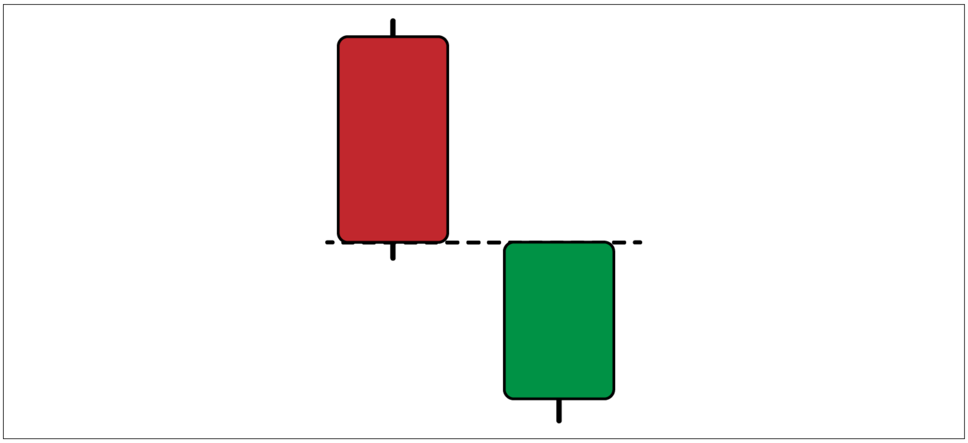


Figure 6-16. A bullish On Neck

The bearish On Neck is composed of a first bullish candlestick and a second bearish candlestick that opens at a gap higher but closes exactly at the close price of the first candlestick. Figure 6-17 shows a bearish On Neck pattern.

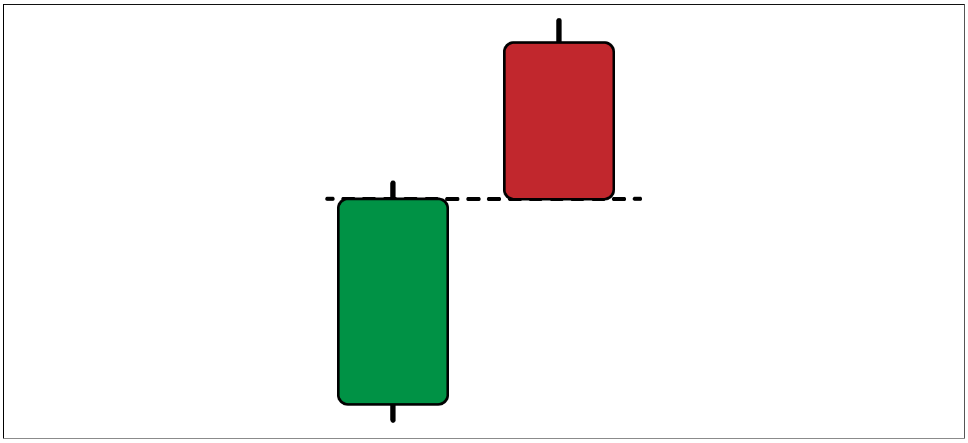


Figure 6-17. A bearish On Neck

Naturally, with such conditions, you need to round the values as I have previously shown you. This is to increase the signal frequency without forgoing any condition.

The psychological intuition of the bullish On Neck pattern comes from the fact that even though sellers were in charge, the gap lower failed to push the market price further down, and the buyers have managed to close at the previous close, thus taking control. The bearish On Neck pattern has the same intuition but in the opposite direction, where buyers were initially in charge before sellers managed to fill the gap lower and signal their presence.

The name On Neck comes from the fact that the second candlestick looks like it is grabbing on the neck of the first candlestick.

Algorithmically, the conditions should follow these steps:

- If the current close price is greater than the current open price and is equal to the previous close price, and the current open price is lower than the previous close price, then print 1 in the next buy row.
- If the current close price is lower than the current open price and is equal to the previous close price, and the current open price is greater than the previous close price, then print –1 in the next sell row.

The signal function for the On Neck pattern is as follows:

```python
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] > data[i, open_column] and \
              data[i, close_column] == data[i - 1, close_column] and \
              data[i, open_column] < data[i - 1, close_column] and \
              data[i - 1, close_column] < data[i - 1, open_column]:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] < data[i, open_column] and \
                data[i, close_column] == data[i - 1, close_column] and \
                data[i, open_column] > data[i - 1, close_column] and \
                data[i - 1, close_column] > data[i - 1, open_column]:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 6-18 shows a signal chart on EURUSD.

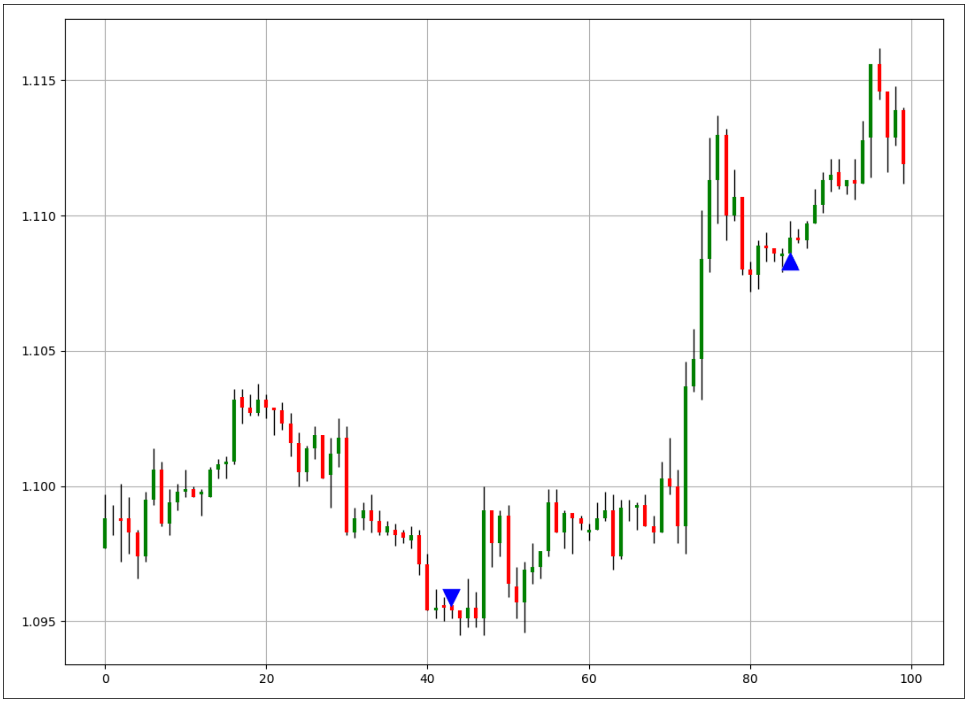


Figure 6-18. Signal chart on EURUSD

Table 6-3 summarizes the performance of the pattern.

Table 6-3. On Neck pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	51.30%	1.14	1.08	191
USDCHF	46.36%	0.88	1.02	220
GBPUSD	54.63%	1.28	1.06	194
USDCAD	50.84%	1.00	0.97	177
BTCUSD	44.04%	0.42	0.53	84
ETHUSD	41.41%	0.56	0.79	99
GOLD	51.56%	1.27	1.19	128
S&P500	36.11%	0.35	0.61	36
FTSE100	57.50%	0.79	0.58	40

Figure 6-19 shows a signal chart on gold.

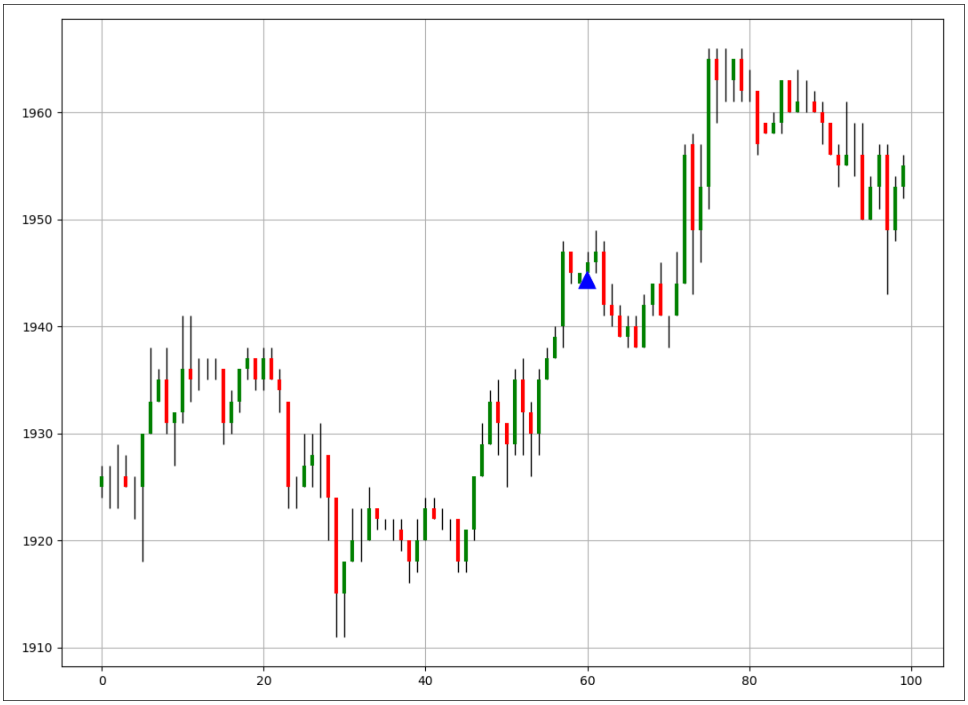


Figure 6-19. Signal chart on gold

To summarize, the On Neck pattern shows mixed results between the markets. Furthermore, it can be quite uncommon.

The Tweezers Pattern

The *Tweezers* pattern is a two-candle contrarian configuration. It is one of the simplest and most straightforward contrarian patterns. The bullish Tweezers pattern is composed of a first bearish candlestick and a second bullish candlestick that shares the same low as the first candlestick. Figure 6-20 illustrates a bullish Tweezers pattern.

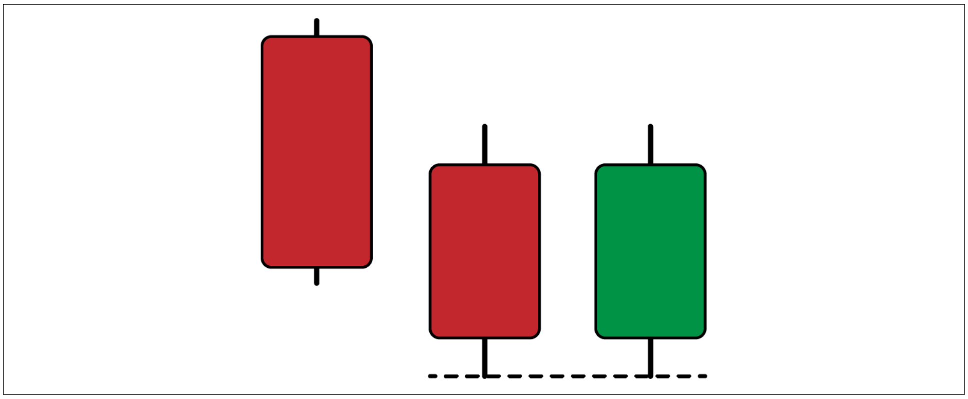


Figure 6-20. A bullish Tweezers

The bearish Tweezers pattern is composed of a first bullish candlestick and a second bearish candlestick that shares the same high as the first candlestick. Figure 6-21 illustrates a bearish Tweezers pattern.

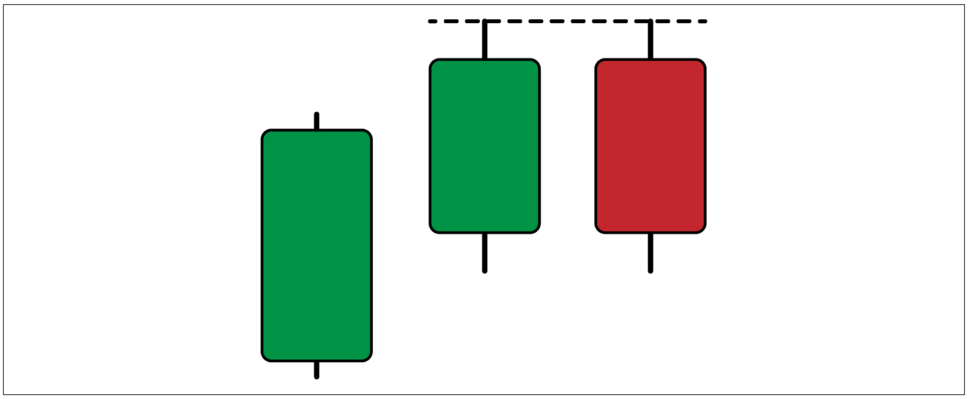


Figure 6-21. A bearish Tweezers

The pattern is called *Tweezers* since the equality condition resembles a pair of tweezers. Arguably, this is a far-fetched way of naming the pattern.

The psychological intuition of the bullish Tweezers pattern comes from the fact that by failing to shape a new low, the market may have found support. The bearish Tweezers pattern comes from the fact that by failing to shape a new high, the market may have found resistance.

Algorithmically, the conditions should follow this intuition:

- If the current low price of a bullish candlestick equals the previous low price of a bearish candle, then print 1 in the next buy row.
- If the current high price of a bearish candlestick equals the previous high price of a bullish candle, then print –1 in the next sell row.

The signal function for the Tweezers pattern is as follows:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] > data[i, open_column] and \
              data[i, low_column] == data[i - 1, low_column] and \
              data[i, close_column] - data[i, open_column] < body and \
              data[i - 1, close_column] - data[i - 1, open_column] < body
              and data[i - 1, close_column] < data[i - 1, open_column] \
              and data[i - 2, close_column] < data[i - 2, open_column]:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] < data[i, open_column] and \
                data[i, high_column] == data[i - 1, high_column] and \
                data[i, close_column] - data[i, open_column] < body and \
                data[i - 1, close_column] - data[i - 1, open_column] < \
                body and data[i - 1, close_column] > data[i - 1, \
                open_column] and data[i - 2, close_column] > data[i - 2, \
                open_column]:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 6-22 shows a signal chart on gold. You can see that sometimes there can be many Tweezers around the same area.

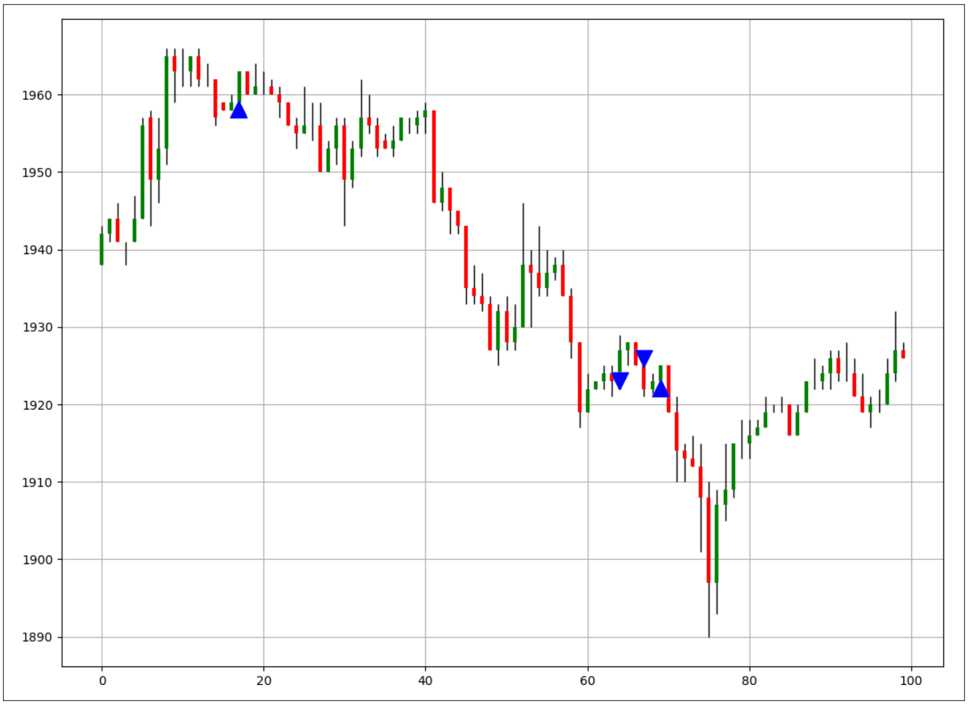


Figure 6-22. Signal chart on gold

Table 6-4 summarizes the performance of the pattern.

Table 6-4. Tweezers pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	50.77%	1.20	1.17	1028
USDCHF	48.27%	1.12	1.20	1104
GBPUSD	49.32%	0.89	0.91	667
USDCAD	49.71%	0.87	0.88	883
BTCUSD	46.61%	1.17	1.35	354
ETHUSD	50.16%	0.82	0.81	911
GOLD	50.57%	1.03	1.01	1038
S&P500	48.61%	0.96	1.01	253
FTSE100	49.04%	1.09	1.14	157

To summarize, the Tweezers pattern is a simple configuration with slightly negative to mixed results.

Figure 6-23 shows a signal chart on the FTSE 100, where a Tweezers pattern can be seen on the bottom.

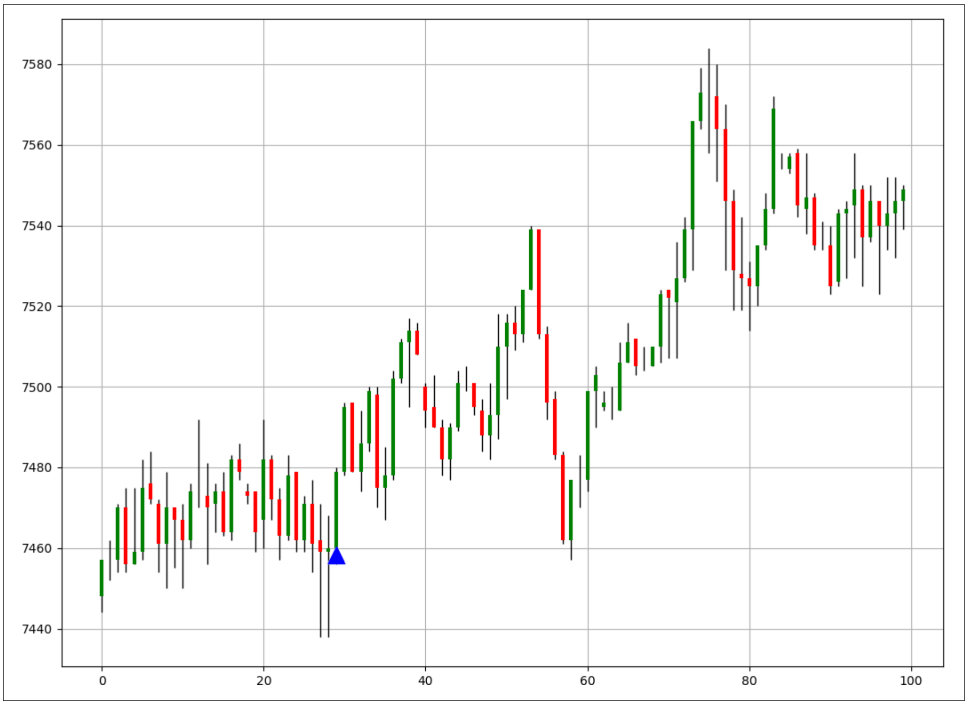


Figure 6-23. Signal chart on the FTSE 100

The Stick Sandwich Pattern

The *Stick Sandwich* pattern is a three-candle contrarian configuration composed of candlesticks that alternate in color. The bullish Stick Sandwich is composed of a first bearish candlestick followed by a smaller bullish candlestick and a bearish candlestick bigger than the second candlestick. Figure 6-24 illustrates a bullish Stick Sandwich pattern.

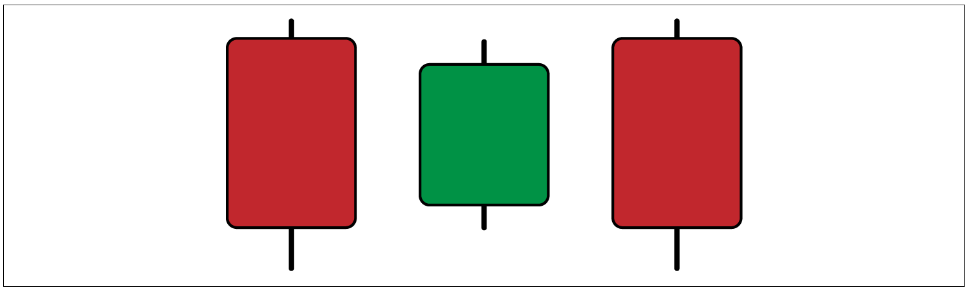


Figure 6-24. A bullish Stick Sandwich

The bearish Stick Sandwich is composed of a first bullish candlestick followed by a smaller bearish candlestick and a bullish candlestick bigger than the second candlestick. Figure 6-25 illustrates a bearish Stick Sandwich pattern.

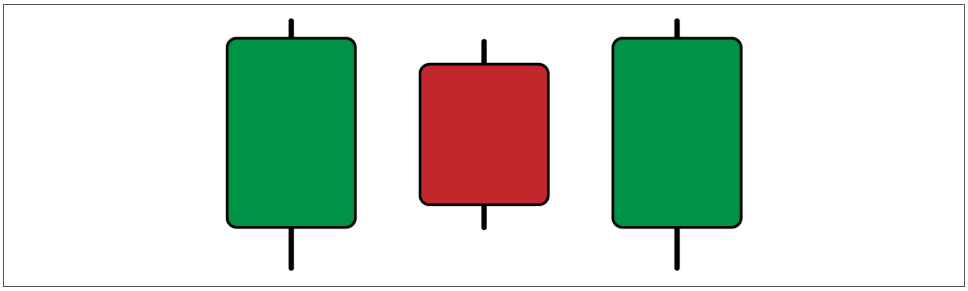


Figure 6-25. A bearish Stick Sandwich

Generally, the Stick Sandwich pattern appears in a trend; hence, the bullish Stick Sandwich is preceded by a bearish price action, and the bearish Stick Sandwich is preceded by a bullish price action.

The name of the Stick Sandwich pattern comes from the fact that it looks like it is a sandwich, with the big outside candlesticks representing bread and the middle candlestick representing the filling.

The psychological intuition of the bullish Stick Sandwich comes from the fact that by failing to shape new lows, the market may have found support (similar to the Tweezers intuition). The bearish Stick Sandwich pattern comes from the fact that by failing to shape new highs, the market may have found resistance.

The signal function for the Stick Sandwich is as follows:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

        try:

            # Bullish pattern
            if data[i, close_column] < data[i, open_column] and \
               data[i, high_column] > data[i - 1, high_column] and \
               data[i, low_column] < data[i - 1, low_column] and \
               data[i - 1, close_column] > data[i - 1, open_column] and \
               data[i - 2, close_column] < data[i - 2, open_column] and \
               data[i - 2, high_column] > data[i - 1, high_column] and \
                data[i - 2, low_column] < data[i - 1, low_column] and \
                data[i - 2, close_column] < data[i - 3, close_column] and \
                data[i - 3, close_column] < data[i - 3, open_column]:

                    data[i + 1, buy_column] = 1

             # Bearish pattern
             elif data[i, close_column] > data[i, open_column] and \
                  data[i, high_column] > data[i - 1, high_column] and \
                  data[i, low_column] < data[i - 1, low_column] and \
                  data[i - 1, close_column] < data[i - 1, open_column] and \
                  data[i - 2, close_column] > data[i - 2, open_column] and \
                  data[i - 2, high_column] > data[i - 1, high_column] and \
                  data[i - 2, low_column] < data[i - 1, low_column] and \
                  data[i - 2, close_column] > data[i - 3, close_column] and\
                  data[i - 3, close_column] > data[i - 3, open_column]:

                    data[i + 1, sell_column] = -1

        except IndexError:

             pass

    return data
```

Figure 6-26 shows a signal chart on GBPUSD.

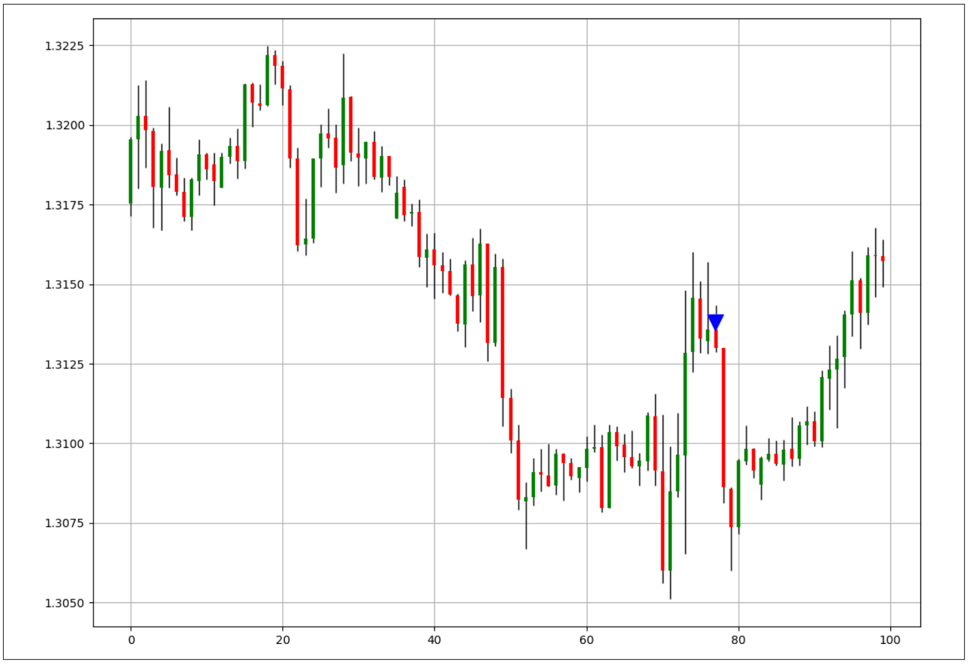


Figure 6-26. Signal chart on GBPUSD

Figure 6-27 shows a signal chart on USDCAD.

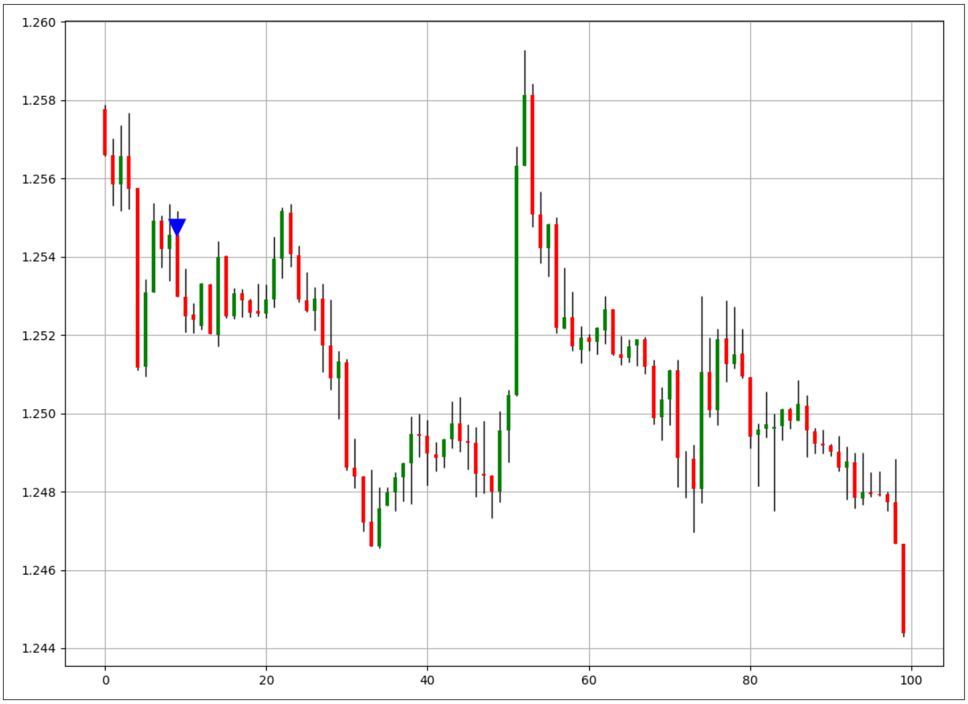


Figure 6-27. Signal chart on USDCAD

Table 6-5 summarizes the performance of the pattern.

Table 6-5. Stick Sandwich pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	44.26%	0.73	0.92	253
USDCHF	47.53%	1.00	1.10	284
GBPUSD	49.37%	0.97	0.99	318
USDCAD	49.12%	1.01	1.05	285
BTCUSD	51.12%	1.88	1.80	178
ETHUSD	50.37%	1.56	1.54	135
GOLD	50.99%	0.96	0.92	251
S&P500	56.86%	1.19	0.90	51
FTSE100	53.84%	2.62	2.24	39

To summarize, the Stick Sandwich pattern slightly outperforms the other patterns shown so far in this chapter. This makes it an interesting candidate to combine with technical indicators and other exit techniques.

The Hammer Pattern

Technically, the *Hammer* pattern is just one name for four different (but similar) candlestick figures known as the Shooting Star, the Hanging Man, the Hammer, and the Inverted Hammer. In my experience, only the Hammer and the Inverted Hammer are worth pursuing since they have a valid intuition discussed later in this section.

The bullish Hammer is a bullish candlestick with a long low wick and no high wick. In addition, it must have a relatively small body. Figure 6-28 illustrates a bullish Hammer.

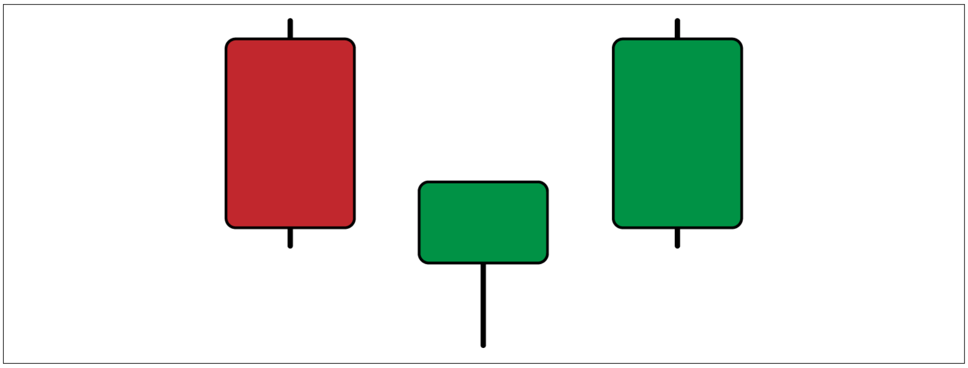


Figure 6-28. A bullish Hammer

The bearish Hammer is a bearish candlestick with a long high wick and no low wick. In addition, it must have a relatively small body. Figure 6-29 illustrates a bearish Hammer.

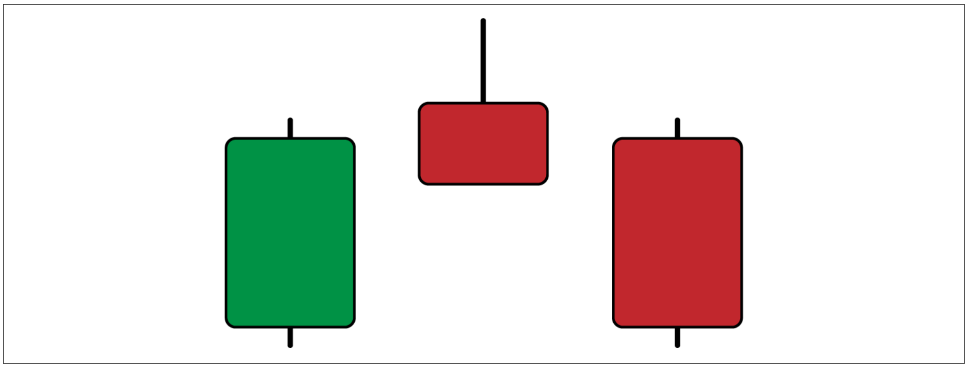


Figure 6-29. A bearish Hammer

The name of the Hammer pattern comes from the fact that the long wick looks like a hammer handle while the body of the candle looks like a hammer head.

The intuition behind the bullish Hammer is that after shaping extreme lows during the hour (or the candlestick's time period), the buyers have managed to close higher than the open price. In contrast, the intuition behind the bearish Hammer is that after shaping extreme highs, the sellers have managed to close lower than the open price, thus becoming in charge.

The signal function for the Hammer pattern is as follows:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

            # Bullish pattern
            if data[i, close_column] > data[i, open_column] and \
               abs(data[i - 1, close_column] - data[i - 1, open_column]) \
               < body and min(data[i - 1, close_column], data[i - 1, \
               open_column]) - data[i - 1, low_column] > 2 * wick and \
               data[i - 1, close_column] == data[i - 1, high_column] and \
               data[i - 2, close_column] < data[i - 2, open_column]:

                     data[i + 1, buy_column] = 1

            # Bearish pattern
            elif data[i, close_column] < data[i, open_column] and \
                 abs(data[i -1, close_column] - data[i - 1, open_column]) \
                 < body and data[i - 1, high_column] - max(data[i - 1, \
                 close_column], data[i - 1, open_column]) > 2 * wick and \
                 data[i - 1, close_column] == data[i - 1, low_column] and \
                 data[i - 2, close_column] > data[i - 2, open_column]:

                     data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 6-30 shows a signal chart on GBPUSD.

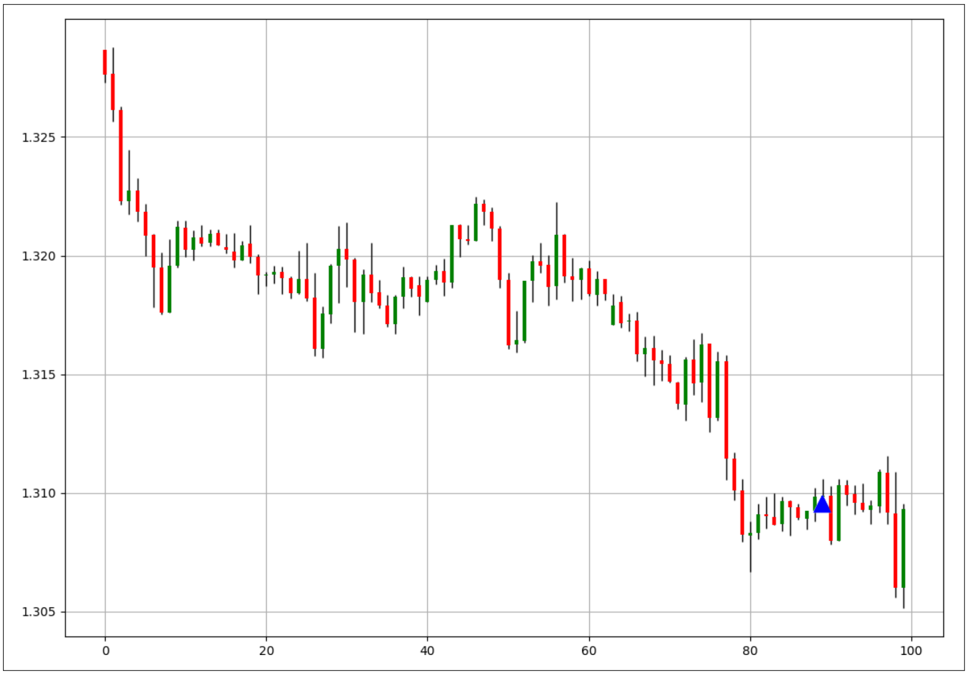


Figure 6-30. Signal chart on GBPUSD

Table 6-6 summarizes the performance of the pattern.

Table 6-6. Hammer pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	47.82%	1.24	1.36	23
USDCHF	61.11%	1.93	1.23	36
GBPUSD	47.61%	0.87	0.96	21
USDCAD	42.85%	0.7	0.94	28
BTCUSD	52.63%	2.39	2.14	19
ETHUSD	60.00%	1.08	0.72	20
GOLD	47.36%	1.09	1.22	38
S&P500	57.57%	2.22	1.63	33
FTSE100	55.00%	1.22	0.99	20

To summarize, the Hammer pattern is a simple and intuitive configuration but can be quite rare, which makes its evaluation biased.

Figure 6-31 shows a signal chart on USDCAD.

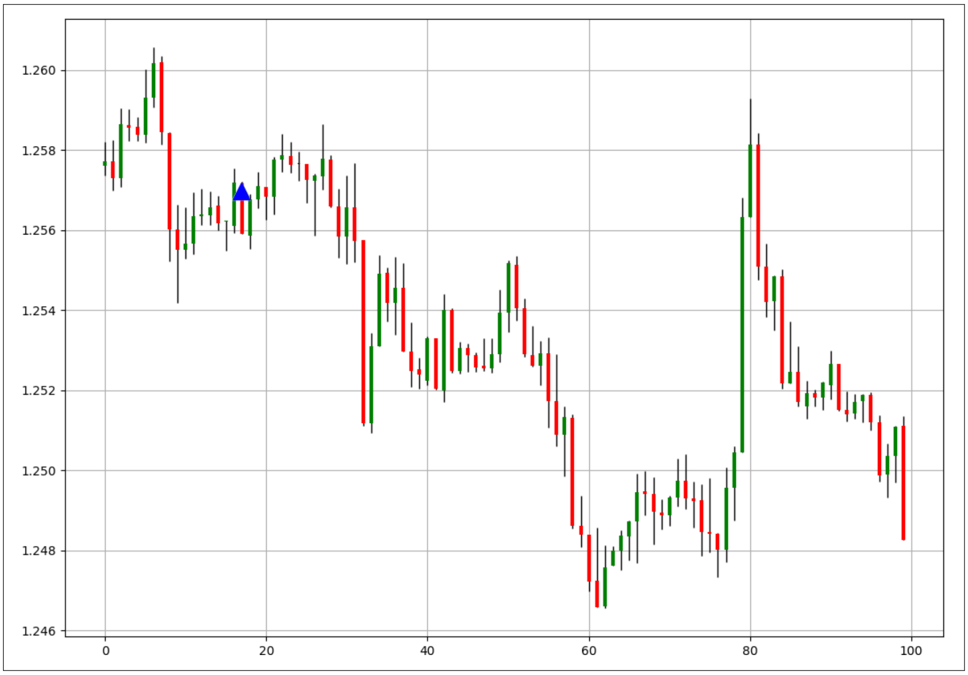


Figure 6-31. Signal chart on USDCAD

The Star Pattern

The *Star* pattern is a three-candle contrarian configuration that is less common than the other patterns. This is because it contains gaps combined with color alternation, which makes it unlikely to occur frequently.

The bullish Star pattern, also known as the Morning Star, is composed of a bearish candle followed by a small-bodied candle that gaps below it, before a third bullish candle, which gaps higher than the middle candle. Figure 6-32 illustrates a Morning Star pattern.

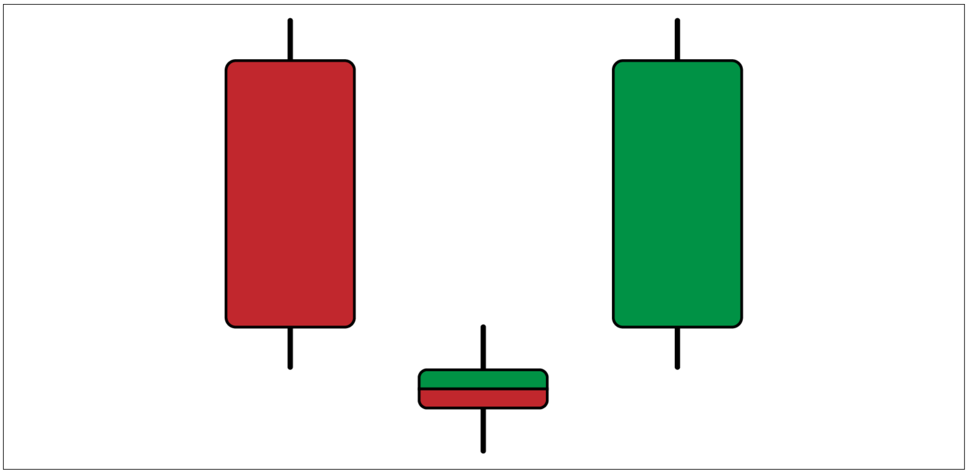


Figure 6-32. A Morning Star

The bearish Star pattern, also known as the Evening Star, is composed of a bullish candle followed by a small-bodied candle that gaps over it, before a third bearish candle, which gaps lower than the middle candle. Figure 6-33 illustrates an evening Star pattern.

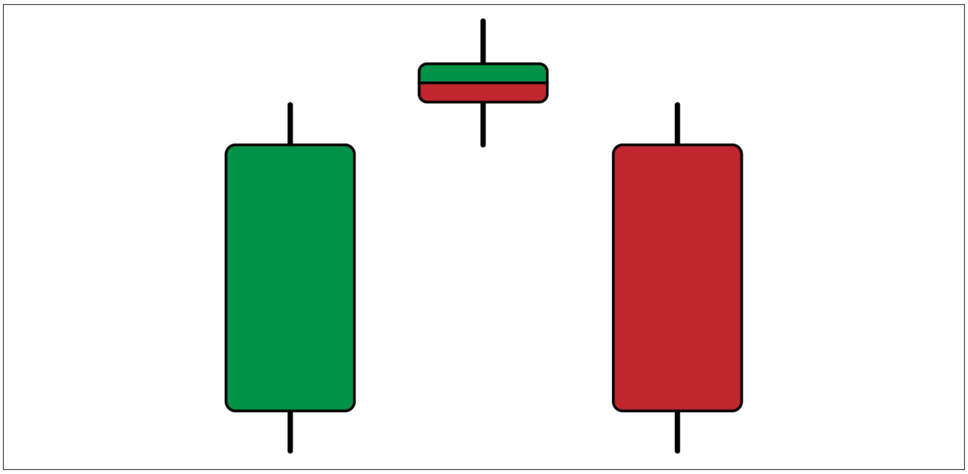


Figure 6-33. An Evening Star

The name of the Star pattern comes from the fact that the middle candlestick is isolated, making it look like a distant star.

The intuition behind the Morning Star is that the market has perfectly shaped a U-turn and demand is now healthier than before due to the shift of balance (confirmed by the third bullish candle). The same intuition applies with regard to the Evening Star.

The signal function for the Star pattern is as follows:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

            # Bullish pattern
            if data[i, close_column] > data[i, open_column] and \
               max(data[i - 1, close_column], data[i - 1, open_column]) \
               < data[i, open_column] and max(data[i - 1, close_column], \
               data[i - 1, open_column]) < data[i - 2, close_column] and \
               data[i - 2, close_column] < data[i - 2, open_column]:

                  data[i + 1, buy_column] = 1

            # Bearish pattern
            elif data[i, close_column] < data[i, open_column] and \
                 min(data[i - 1, close_column], data[i - 1, open_column]) \
                 > data[i, open_column] and min(data[i - 1, close_column],\
                 data[i - 1, open_column]) > data[i - 2, close_column] \
                 and data[i - 2, close_column] > data[i - 2, open_column]:

                  data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 6-34 shows a signal chart on the S&P 500.

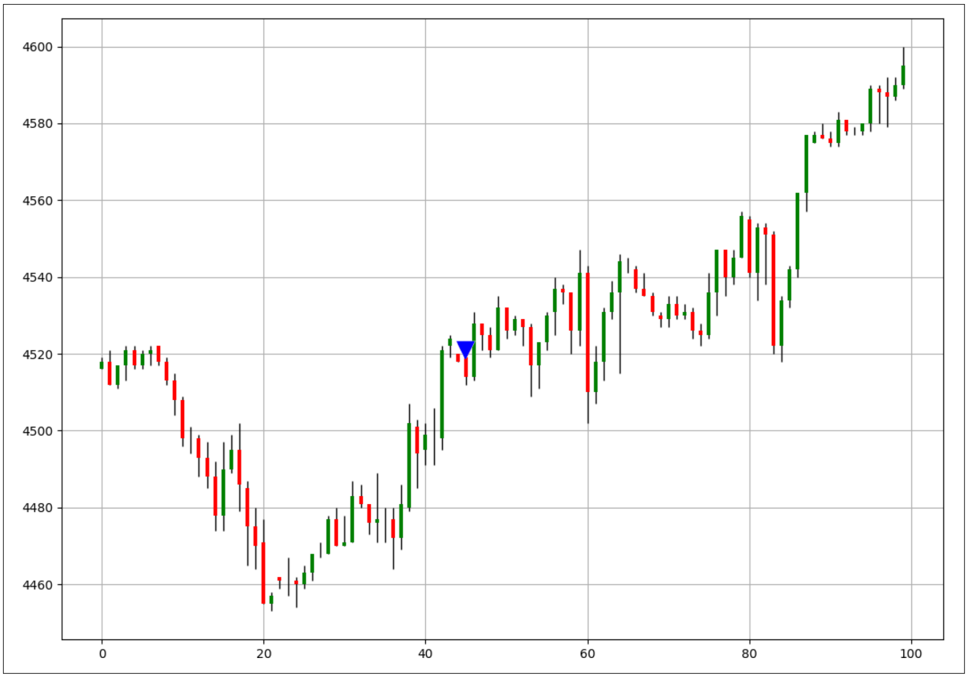


Figure 6-34. Signal chart on the S&P 500

The back-tests are omitted due to the rarity of the pattern.

The Piercing Pattern

The *Piercing* pattern is one of the most well-known two-candle configurations in the world of technical pattern recognition. The bullish Piercing pattern is composed of a bearish candle followed by a bullish candlestick that opens at a gap lower and closes above the close of the previous candlestick but below its open price. Figure 6-35 illustrates a bullish Piercing pattern.

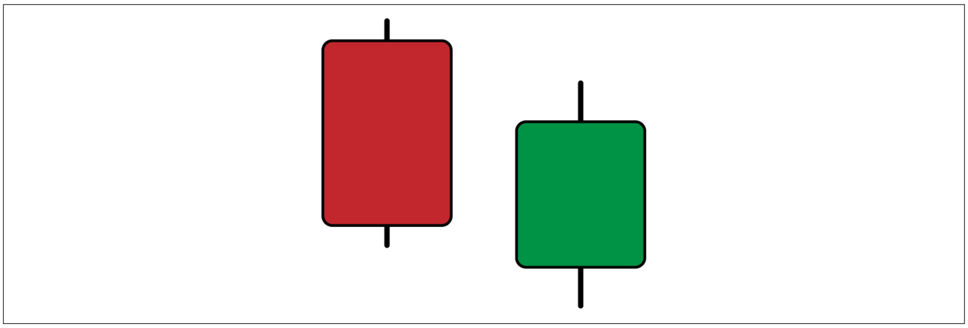


Figure 6-35. A bullish Piercing pattern

The bearish Piercing pattern is composed of a bullish candle followed by a bearish candlestick that opens at a gap higher and closes below the close of the previous candlestick but above its close price. Figure 6-36 illustrates a bullish Piercing pattern.

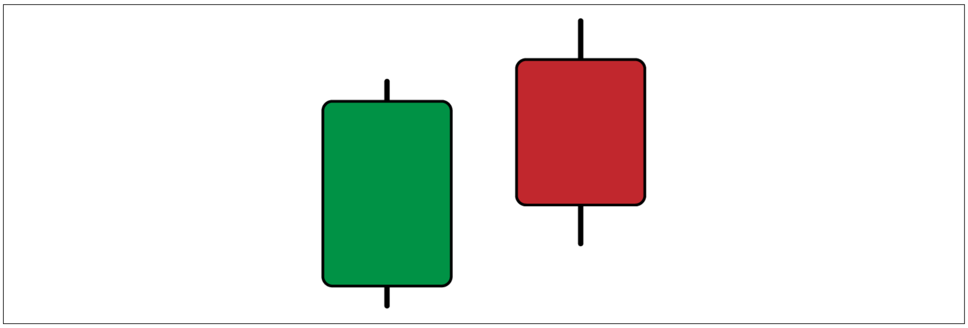


Figure 6-36. A bearish Piercing pattern

> The name of the Piercing pattern comes from the fact that the second candle comes from the outside and penetrates the first candle.

The intuition behind the bullish Piercing pattern is that the previous close is surpassed even though the market has opened with a gap lower. The pattern shows a reluctance to abandon the purchase from the buyers whose buying activities prevented the market from continuing lower. Similarly, with a bearish Piercing pattern, the sellers are refusing to leave the gap unfilled and are taking control.

The signal function for the Piercing pattern is as follows:

```python
def signal(data, open_column, close_column, buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] > data[i, open_column] and \
              data[i, close_column] < data[i - 1, open_column] and \
              data[i, close_column] > data[i - 1, close_column] and \
              data[i, open_column] < data[i - 1, close_column] and \
              data[i - 1, close_column] < data[i - 1, open_column] and \
              data[i - 2, close_column] < data[i - 2, open_column]:

                    data[i + 1, buy_column] = 1
```

```
            # Bearish pattern
            elif data[i, close_column] < data[i, open_column] and \
                 data[i, close_column] > data[i - 1, open_column] and \
                 data[i, close_column] < data[i - 1, close_column] and \
                 data[i, open_column] > data[i - 1, close_column] and \
                 data[i - 1, close_column] > data[i - 1, open_column] and \
                 data[i - 2, close_column] > data[i - 2, open_column]:

                    data[i + 1, sell_column] = -1

        except IndexError:

            pass

    return data
```

Figure 6-37 shows a signal chart on gold.

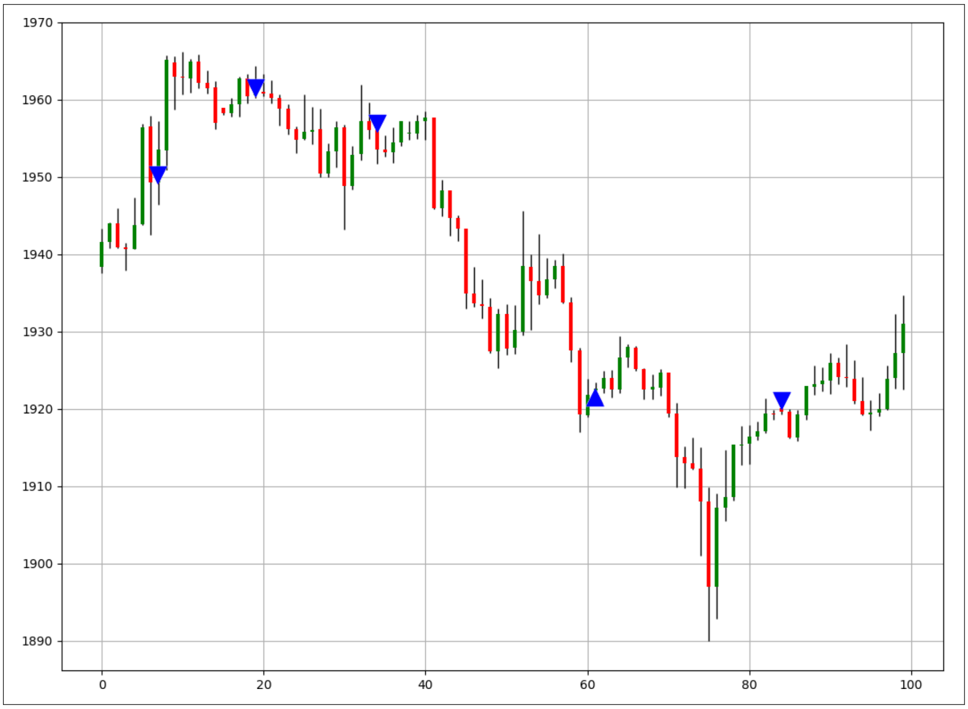


Figure 6-37. Signal chart on gold

Figure 6-38 shows a signal chart on the FTSE 100.

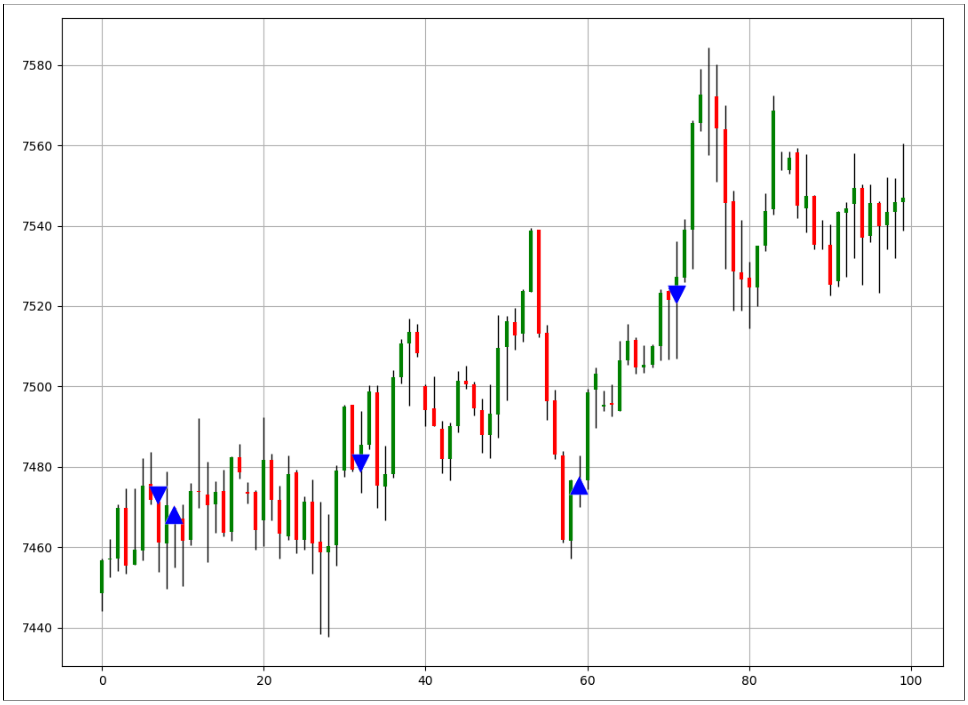


Figure 6-38. Signal chart on the FTSE 100

Table 6-7 summarizes the performance of the pattern.

Table 6-7. Piercing pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	51.41%	0.95	0.90	1904
USDCHF	49.26%	0.93	0.95	1961
GBPUSD	50.83%	0.84	0.81	1985
USDCAD	50.73%	0.93	0.90	1961
BTCUSD	50.88%	0.87	0.84	1584
ETHUSD	48.68%	0.91	0.96	1066
GOLD	51.30%	0.96	0.92	2341
S&P500	49.46%	1.06	1.07	186
FTSE100	49.49%	0.96	0.98	295

The Piercing pattern can be common, but its profitability is low compared to other patterns.

The Engulfing Pattern

The *Engulfing* pattern, a two-candle contrarian configuration, is the mirror image of the Harami pattern. Interestingly, they are both contrarian in nature. The bullish Engulfing pattern is composed of a first bearish candle and a subsequent bullish candlestick that fully englobes it in a strict way. Figure 6-39 illustrates a bullish Engulfing pattern.

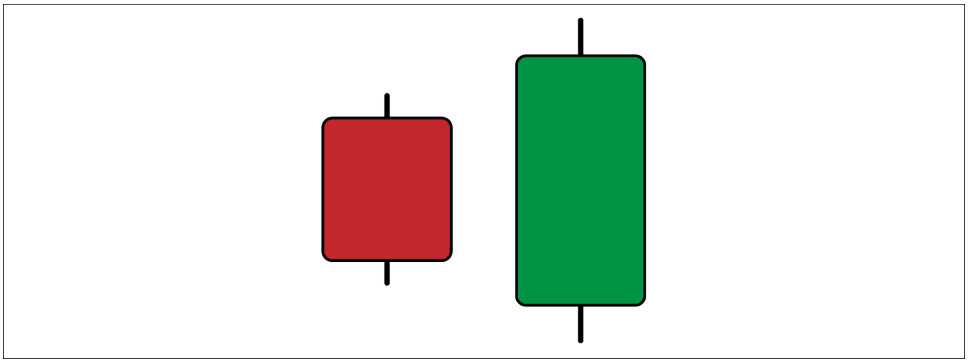


Figure 6-39. A bullish Engulfing

The bearish Engulfing pattern is composed of a first bullish candle and a subsequent bearish candlestick that fully englobes it also in a strict way. Figure 6-40 illustrates a bearish Engulfing pattern.

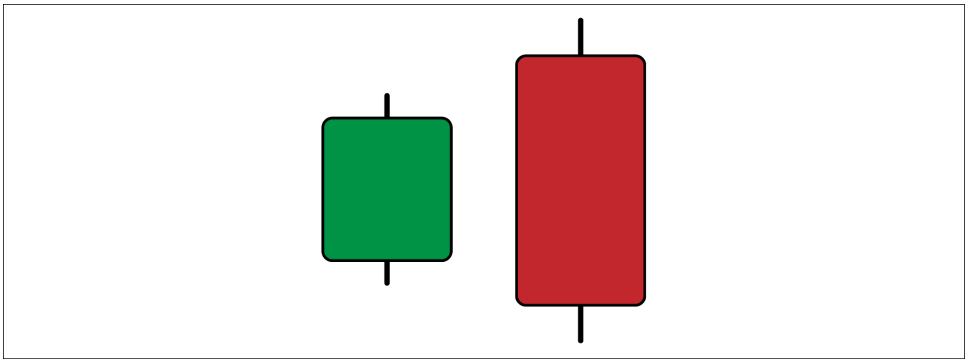


Figure 6-40. A bearish Engulfing

The name of the Engulfing pattern comes from the fact that the second candle totally covers (or engulfs) the first candle.

The intuition behind the Engulfing pattern lies in the total eclipse of the first candle by the second candle. Obviously, when an Engulfing occurs, a change of dynamic presents itself, indicated by the bigger candlestick in the opposite direction.

The signal function for the Engulfing pattern is as follows:

```python
def signal(data, open_column, close_column, buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] > data[i, open_column] and \
              data[i, open_column] < data[i - 1, close_column] and \
              data[i, close_column] > data[i - 1, open_column] and \
              data[i - 1, close_column] < data[i - 1, open_column] and \
              data[i - 2, close_column] < data[i - 2, open_column]:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] < data[i, open_column] and \
                data[i, open_column] > data[i - 1, close_column] and \
                data[i, close_column] < data[i - 1, open_column] and \
                data[i - 1, close_column] > data[i - 1, open_column] and \
                data[i - 2, close_column] > data[i - 2, open_column]:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 6-41 shows a signal chart on EURUSD.

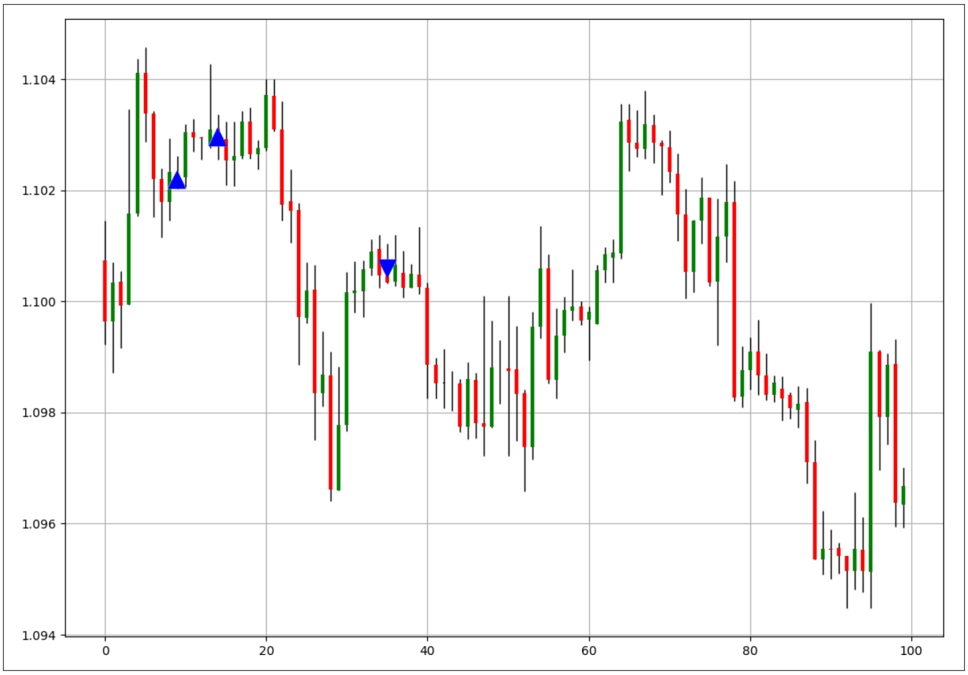


Figure 6-41. Signal chart on EURUSD

Table 6-8 summarizes the performance of the pattern.

Table 6-8. Engulfing pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	46.73%	0.94	1.07	1881
USDCHF	47.46%	0.95	1.05	1890
GBPUSD	49.87%	1.06	1.06	1989
USDCAD	48.96%	1.00	1.04	1793
BTCUSD	47.88%	0.92	1.00	1251
ETHUSD	47.71%	1.03	1.13	876
GOLD	47.08%	0.97	1.09	2130
S&P500	50.25%	0.97	0.96	193
FTSE100	42.49%	0.64	0.87	313

Figure 6-42 shows a signal chart on BTCUSD.

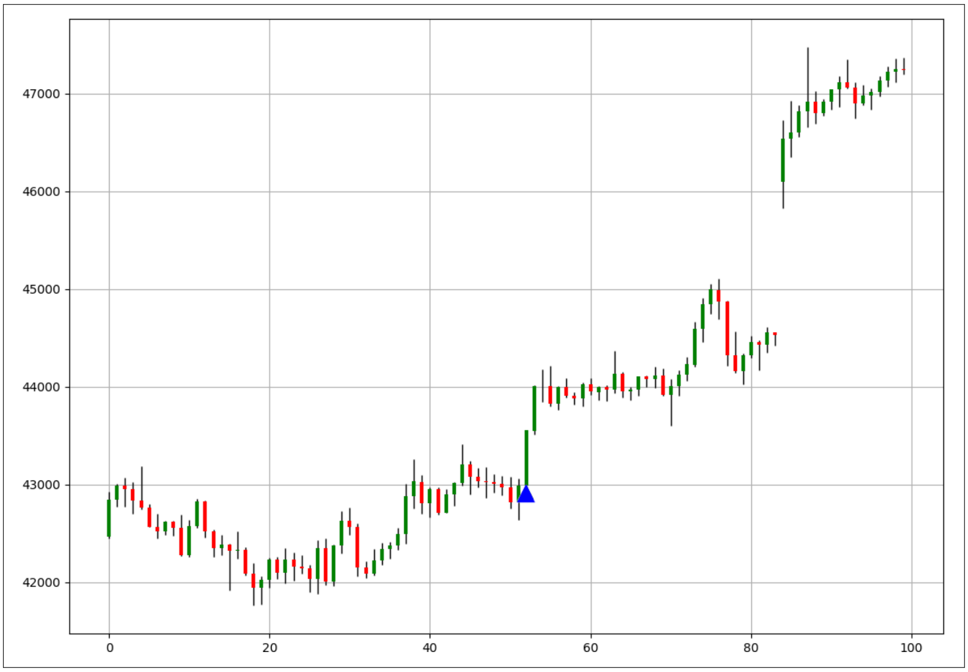


Figure 6-42. Signal chart on BTCUSD

To summarize, the Engulfing pattern resembles the Piercing pattern with regard to predictability. Both patterns may benefit from changed trading conditions such as entries and exits.

The Abandoned Baby Pattern

This is not really a tradable pattern because it is extremely rare. I am covering it for educational purposes only as you should not expect to see much of it in practice. The *Abandoned Baby* is a three-candle configuration where the middle candlestick is a Doji. The bullish Abandoned Baby is composed of a bearish candle followed by a Doji candlestick that gaps lower and has a high that does not touch the low of the first candlestick. Finally, it has a third bullish candlestick, also with a low that does not touch the high of the middle candlestick. Figure 6-43 illustrates a bullish Abandoned Baby pattern.

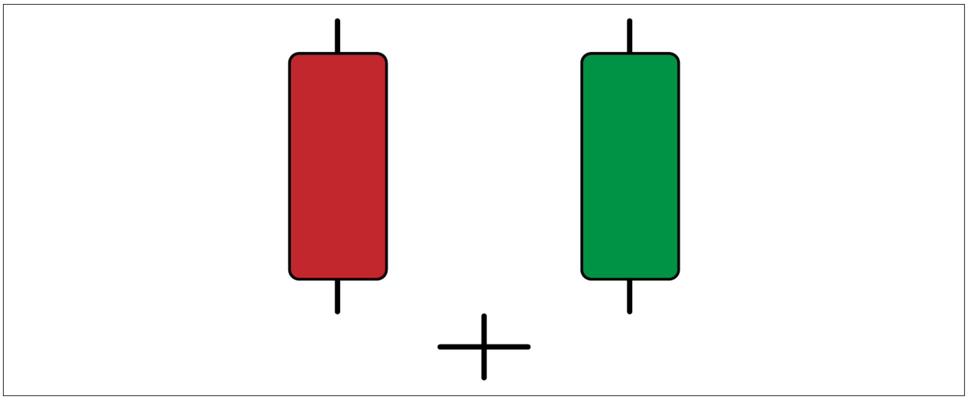


Figure 6-43. A bullish Abandoned Baby

The bearish Abandoned Baby is composed of a bullish candle followed by a Doji candlestick that gaps higher and has a low that does not touch the high of the first candlestick. Finally, there is a third bearish candlestick, also with a high that does not touch the low of the middle candlestick. Figure 6-44 illustrates a bearish Abandoned Baby pattern.

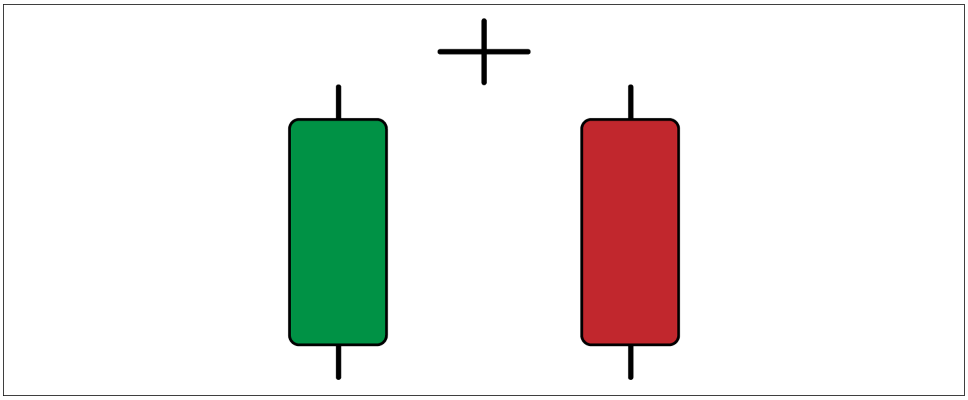


Figure 6-44. A bearish Abandoned Baby

The name of the Abandoned Baby pattern comes from the fact that the middle candlestick is completely isolated with its wicks not touching the other two candlesticks' wicks.

The pattern may be more common for less liquid stocks and other assets. Generally, liquid and major markets rarely see such complex configurations.

The intuition behind the bullish Abandoned Baby is that the market has shaped a U-turn and demand is now healthier than before due to the shift of balance. The same intuition applies to the bearish Abandoned Baby.

The signal function for the Abandoned Baby pattern is as follows:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

          # Bullish pattern
          if data[i, close_column] > data[i, open_column] and \
             data[i - 1, close_column] == data[i - 1, open_column] and \
             data[i - 1, high_column] < data[i, low_column] and \
             data[i - 1, high_column] < data[i - 2, low_column] and \
             data[i - 2, close_column] < data[i - 2, open_column]:

                 data[i + 1, buy_column] = 1

          # Bearish pattern
          elif data[i, close_column] < data[i, open_column] and \
               data[i - 1, close_column] == data[i - 1, open_column] and \
               data[i - 1, low_column] > data[i, high_column] and \
               data[i - 1, low_column] > data[i - 2, high_column] and \
               data[i - 2, close_column] > data[i - 2, open_column]:

                 data[i + 1, sell_column] = -1

       except IndexError:

          pass

    return data
```

I won't show the performance for this pattern because there is not enough data to make any conclusion.

To summarize, the Abandoned Baby is mostly a theoretical pattern that may look good in books but, in reality, it is so rare that you cannot judge its performance. Also, the few patterns that are detected do not show much predictive potential, rendering the Abandoned Baby a beautiful mythological configuration that underperforms its reputation.

The Spinning Top Pattern

The *Spinning Top* pattern is a three-candle contrarian configuration that resembles the Doji pattern but is generally more common and hints indirectly at a rise in volatility. The bullish Spinning Top is composed of a bearish candle followed by a small-bodied candle with long wicks (highs and lows). Finally, a bullish candle appears to confirm the expected upside move. Figure 6-45 illustrates a bullish Spinning Top pattern.

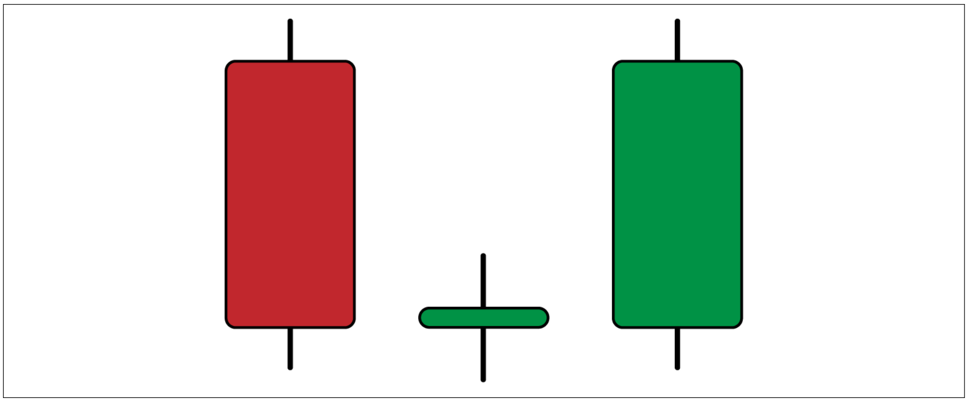


Figure 6-45. A bullish Spinning Top

The bearish Spinning Top is composed of a bullish candle followed by a small-bodied candle with long wicks (highs and lows). Finally, a bearish candle appears to confirm the expected downside move. Figure 6-46 illustrates a bearish Spinning Top pattern.

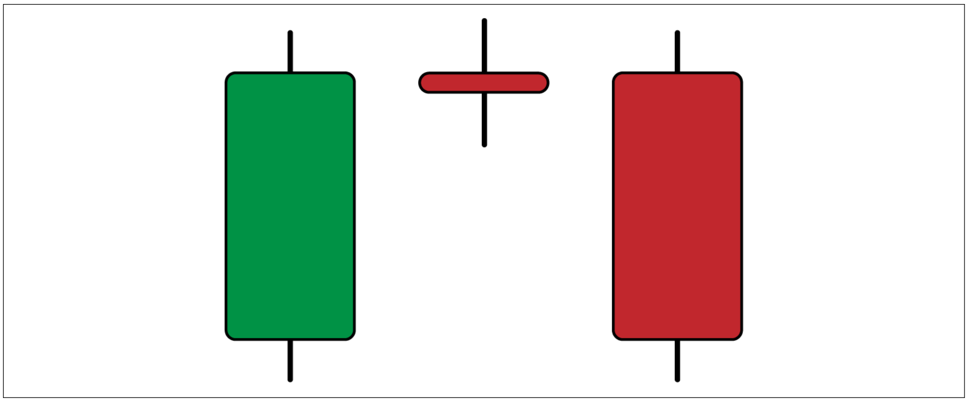


Figure 6-46. A bearish Spinning Top

The name of the Spinning Top pattern is obviously related to the fact that it resembles a spinning top.

The intuition behind the Spinning Top pattern is the same as behind the Doji pattern, albeit it is less powerful because the close price does not exactly equal the open price. However, the volatility stemming from the long wicks shows the battle between buyers and sellers and may signal that a clear direction should appear after the pattern.

The signal function for the Spinning Top pattern is as follows:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

            # Bullish pattern
            if data[i, close_column] - data[i, open_column] > body and \
               data[i - 1, high_column] - data[i - 1, close_column] >= wick \
               and data[i - 1, open_column] - data[i - 1, low_column] >= \
               wick and data[i - 1, close_column] - data[i - 1, \
               open_column] < body and data[i - 1, close_column] > data[i \
               - 1, open_column] and data[i - 2, close_column] < data[i - \
               2, open_column] and data[i - 2, open_column] - data[i - 2, \
               close_column] > body:

                   data[i + 1, buy_column] = 1

            # Bearish pattern
            elif data[i, open_column] - data[i, close_column] > body and \
                 data[i - 1, high_column] - data[i - 1, open_column] >= \
                 wick and data[i - 1, close_column] - data[i - 1, \
                 low_column] >= wick and data[i - 1, open_column] - \
                 data[i - 1, close_column] < body and data[i - 1, \
                 close_column] < data[i - 1, open_column] and data[i - 2, \
                 close_column] > data[i - 2, open_column] and data[i - 2, \
                 close_column] - data[i - 2, open_column] > body:

                   data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 6-47 shows a signal chart on the FTSE 100.

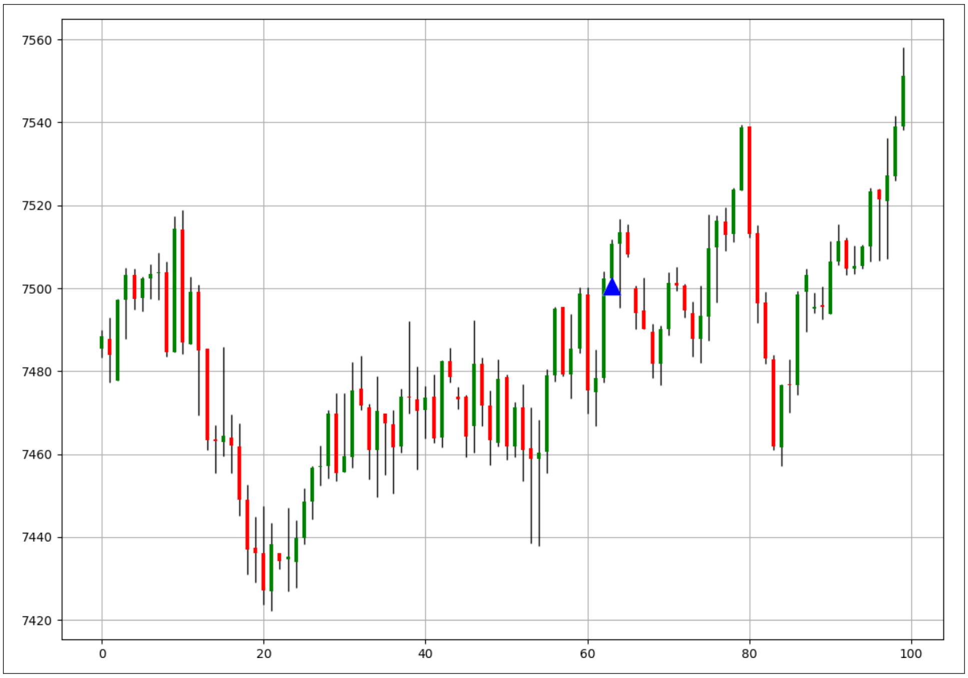


Figure 6-47. Signal chart on the FTSE 100

Table 6-9 summarizes the performance of the pattern.

Table 6-9. Spinning Top pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	51.02%	0.87	0.83	243
USDCHF	49.69%	0.79	0.80	163
GBPUSD	48.79%	1.14	1.19	414
USDCAD	45.22%	0.79	0.96	272
BTCUSD	48.62%	0.77	0.81	401
ETHUSD	38.37%	0.66	1.06	185
GOLD	41.22%	0.62	0.88	228
S&P500	50.00%	1.18	1.18	36
FTSE100	42.34%	0.84	1.14	111

The Inside Up/Down Pattern

The *Inside Up/Down* pattern is a three-candle contrarian configuration that signals the end of an initial move using two confirmation candles after an initial candle. The bullish Inside Up pattern is composed of a bearish candle followed by a smaller bullish candle with a body within the body of the first one. Finally, a bullish candlestick must appear and surpass the open of the first candlestick. Figure 6-48 illustrates a bullish Inside Up pattern.

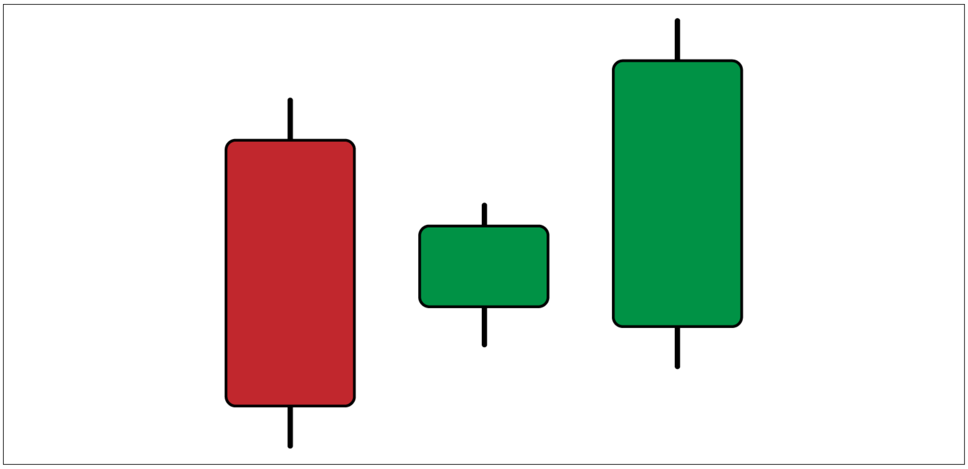


Figure 6-48. A bullish Inside Up

The bearish Inside Down pattern is composed of a bullish candle followed by a smaller bearish candle with a body within the body of the first one. Finally, a bearish candlestick must appear and break the open of the first candlestick. Figure 6-49 illustrates a bearish Inside Down pattern.

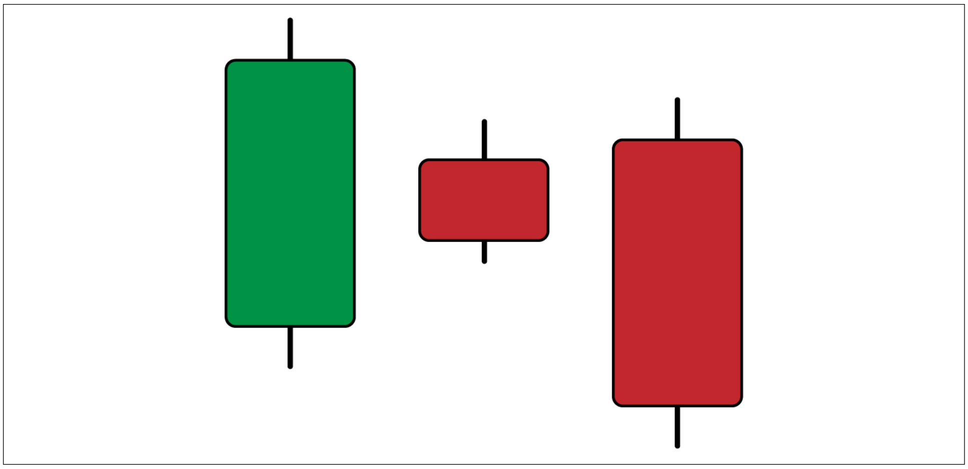


Figure 6-49. A bearish Inside Down

The body of the candle is the absolute difference between the open and close prices, while the range of the candle is the absolute difference between the high and low prices.

The intuition behind the Inside Up pattern is that after a prevailing bearish trend, the buyers are taking control. The second candle shows the first balance of power before the third candle confirms this new move by surpassing the close of the second candle. The second candle's containment within the body of the first one signals that the sellers could not push the price lower—a sign that they are starting to struggle. The intuition behind the Inside Down pattern is that after a prevailing bullish trend, the sellers are taking control. The second candle's containment within the body of the first one signals that the buyers could not push the price higher.

Notice that the Inside Up/Down pattern is like a version of a Harami followed by a big candlestick that confirms the new direction.

The signal function for the Inside Up/Down pattern is as follows:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i - 2, close_column] < data[i - 2, open_column] and \
              abs(data[i - 2, open_column] - data[i - 2, close_column]) > \
              body and data[i - 1, close_column] < data[i - 2, \
              open_column] and data[i - 1, open_column] > data[i - 2, \
              close_column] and data[i - 1, close_column] > data[i - 1, \
              open_column] and data[i, close_column] > data[i - 2, \
              open_column] and data[i, close_column] > data[i, \
              open_column] and abs(data[i, open_column] - data[i, \
              close_column]) > body:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i - 2, close_column] > data[i - 2, open_column] and \
                abs(data[i - 2, close_column] - data[i - 2, open_column]) \
                > body and data[i - 1, close_column] > data[i - 2, \
                open_column] and data[i - 1, open_column] < data[i - 2, \
                close_column] and data[i - 1, close_column] < data[i - 1, \
                open_column] and data[i, close_column] < data[i - 2, \
                open_column] and data[i, close_column] < data[i, \
                open_column] and abs(data[i, open_column] - data[i, \
                close_column]) > body:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 6-50 shows a signal chart on USDCAD.

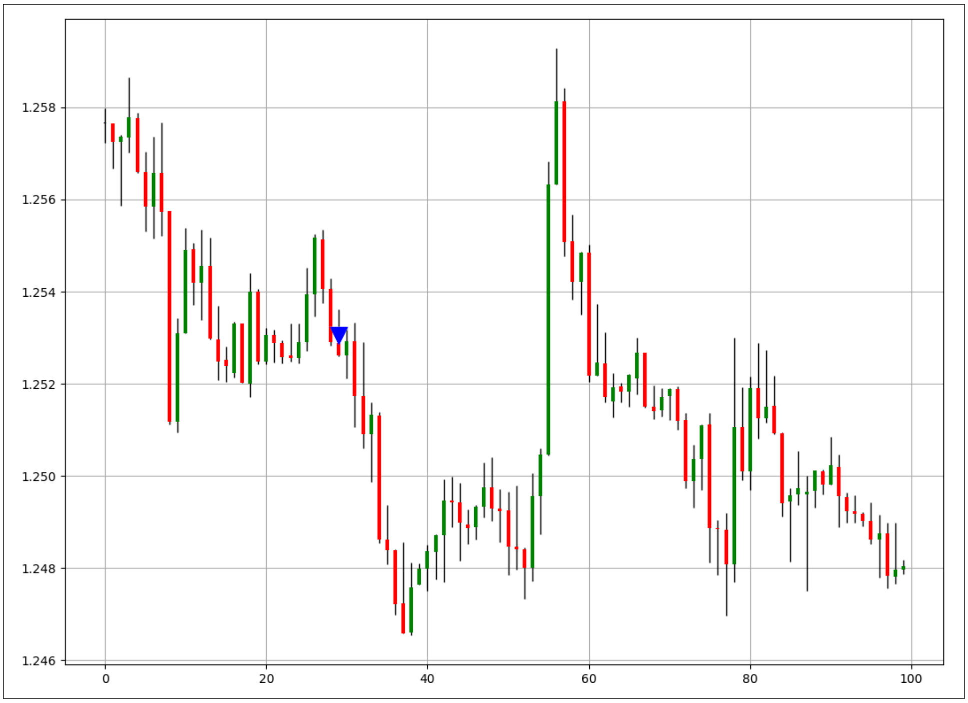


Figure 6-50. Signal chart on USDCAD

Table 6-10 summarizes the performance of the pattern.

Table 6-10. Inside Up/Down pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	49.82%	0.94	0.95	572
USDCHF	48.89%	0.98	1.02	452
GBPUSD	48.86%	1.15	1.20	659
USDCAD	50.93%	1.26	1.22	591
BTCUSD	51.40%	1.30	1.23	214
ETHUSD	42.22%	0.73	1.00	90
GOLD	46.73%	0.73	0.83	199
S&P500	50.00%	0.62	0.62	64
FTSE100	54.13%	1.04	0.88	133

Figure 6-51 shows a signal chart on ETHUSD.

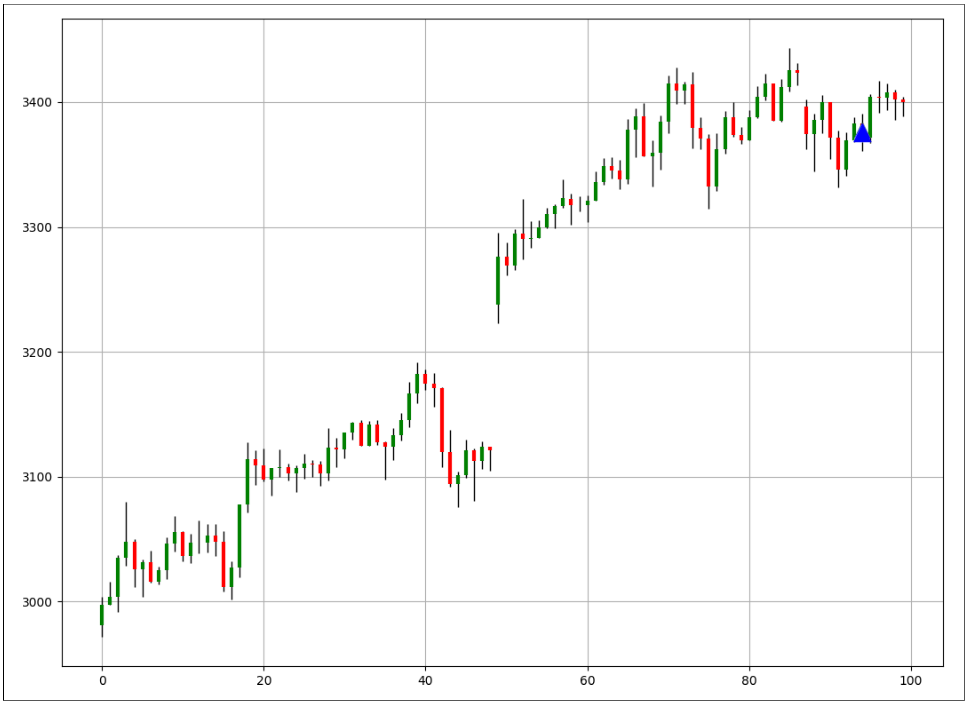


Figure 6-51. Signal chart on ETHUSD

The Tower Pattern

The *Tower* pattern is a multicandle (generally five) contrarian configuration that signals the end of a gradual trend. The bullish Tower Bottom pattern starts with a bearish candlestick followed by three small-bodied candles, with the middle one being slightly lower than the other two. Finally, a normal-sized bullish candlestick must appear to confirm the upside exit. Figure 6-52 illustrates a Tower Bottom pattern.

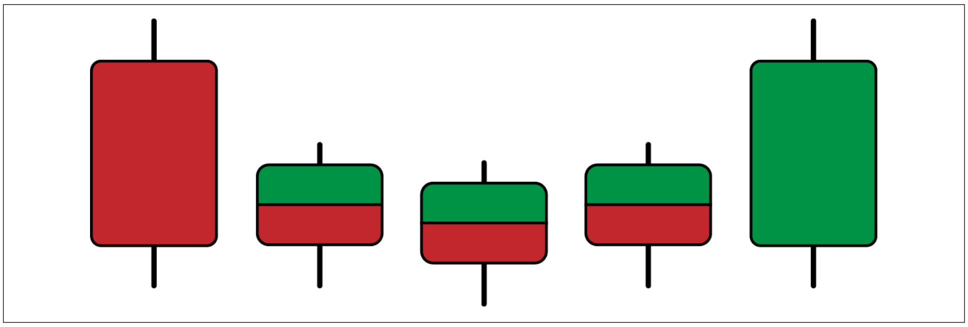


Figure 6-52. A Tower Bottom

The bearish Tower Top pattern starts with a bullish candlestick followed by three small-bodied candles, with the middle one being slightly higher than the other two. Finally, a normal-sized bearish candlestick must appear to confirm the downside exit. Figure 6-53 illustrates a Tower Top pattern.

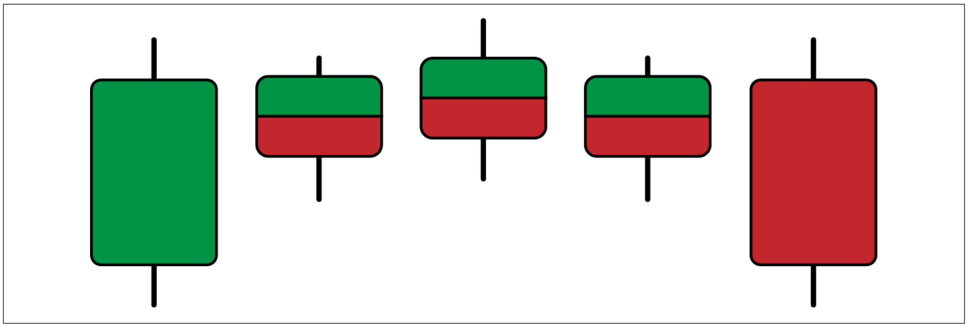


Figure 6-53. A Tower Top

The Tower pattern is a stabilization configuration that hints at a possible end of the initial trend.

The intuition behind the Tower Bottom pattern is that after a bearish move, the small-bodied candles hint at a possible stabilization, which turns out to be an expected bullish reversal by the last bullish candle. Similarly, the Tower Top pattern hints at a possible consolidation before the bearish candle validates the downside move.

Multicandle patterns are sometimes hard to code due to their vague conditions. Many versions of the Tower pattern exist, but I have found that sticking with a maximum of five candles is better than creating conditions involving more candles.

The signal function for the Tower pattern is as follows:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

        try:

            # Bullish pattern
            if data[i, close_column] > data[i, open_column] and \
               data[i, close_column] - data[i, open_column] > body and \
               data[i - 2, low_column] < data[i - 1, low_column] and \
               data[i - 2, low_column] < data[i - 3, low_column] and \
               data[i - 4, close_column] < data[i - 4, open_column] and \
               data[i - 4, open_column] - data[i, close_column] > body:

                     data[i + 1, buy_column] = 1

            # Bearish pattern
            elif data[i, close_column] < data[i, open_column] and \
                 data[i, open_column] - data[i, close_column] > body and \
                 data[i - 2, high_column] > data[i - 1, high_column] and \
                 data[i - 2, high_column] > data[i - 3, high_column] and \
                 data[i - 4, close_column] > data[i - 4, open_column] and \
                 data[i - 4, close_column] - data[i, open_column] > body:

                     data[i + 1, sell_column] = -1

        except IndexError:

             pass

    return data
```

Figure 6-54 shows a signal chart on USDCAD.

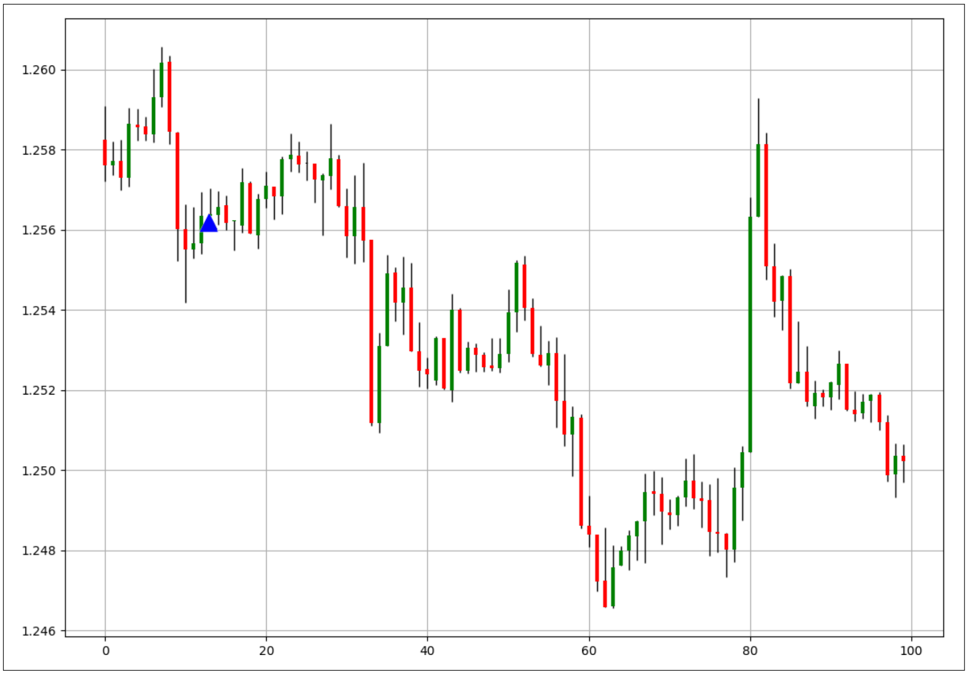


Figure 6-54. Signal chart on USDCAD

Figure 6-55 shows a signal chart on GBPUSD.

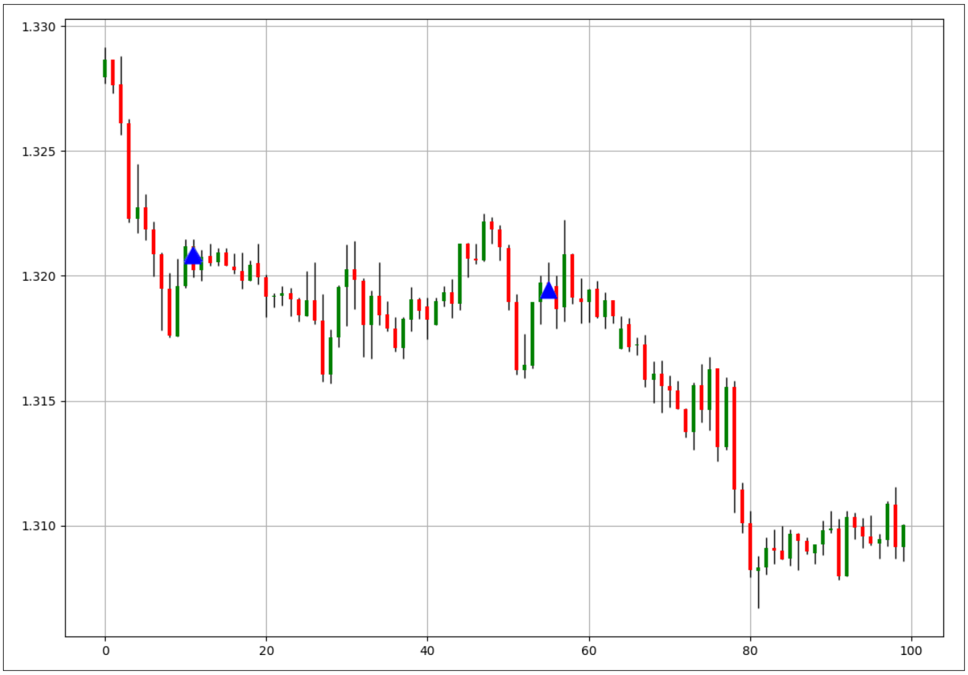


Figure 6-55. Signal chart on GBPUSD

Table 6-11 summarizes the performance of the pattern.

Table 6-11. Tower pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	49.31%	1.01	1.04	878
USDCHF	50.47%	1.2	1.18	735
GBPUSD	46.30%	0.95	1.10	1203
USDCAD	50.44%	1.09	1.07	1013
BTCUSD	43.33%	1.06	1.38	503
ETHUSD	45.45%	0.88	1.05	231
GOLD	47.32%	0.73	0.81	336
S&P500	43.75%	0.85	1.09	16
FTSE100	50.00%	1.27	1.27	60

CHAPTER 7

Modern Contrarian Patterns

Continuing in the spirit of contrarian trading, this chapter discusses the modern contrarian patterns, which are candlestick formations that I have noticed over the years and can be an addition to the already known patterns.

The purpose of this chapter is, as usual, to describe the patterns' objective conditions and back-test them so that you form an opinion about their frequency and predictability.

The Doppelgänger Pattern

In German, the word *Doppelgänger* means "double goer" or "double walker" and is generally used to refer to people who look exactly the same but are biologically unrelated. The Doppelgänger candlestick pattern is a three-candlestick reversal configuration that I like to use to confirm intermediate reversals.

Figure 7-1 illustrates a bullish Doppelgänger. The pattern is composed of a bearish candle followed by two numerically similar candlesticks (same highs and lows, and either they must have the same close and open prices or the close price must equal the open price) with flexibility on their type (bullish or bearish).

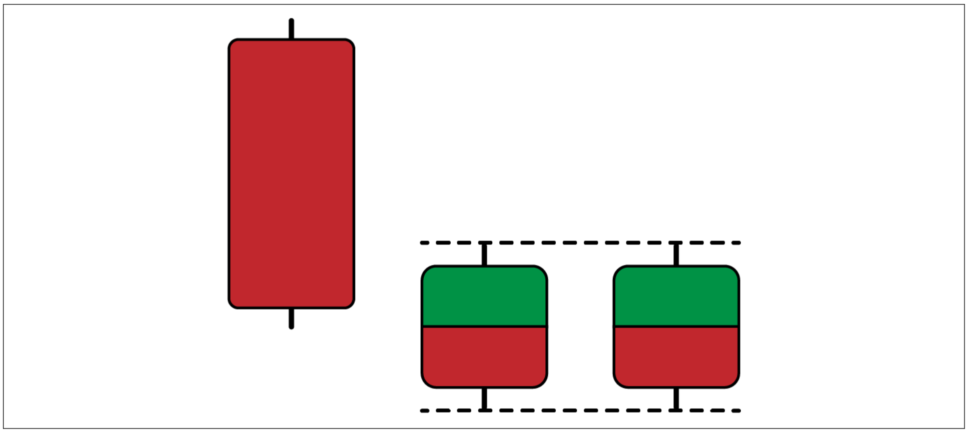


Figure 7-1. A bullish Doppelgänger

In comparison, Figure 7-2 shows the theoretical bearish Doppelgänger pattern. The pattern is composed of a bullish candle followed by two numerically similar candlesticks with flexibility on their type (bullish or bearish). Note that these are theoretical conditions; in real life, some flexibility may be added (as you will see in this chapter).

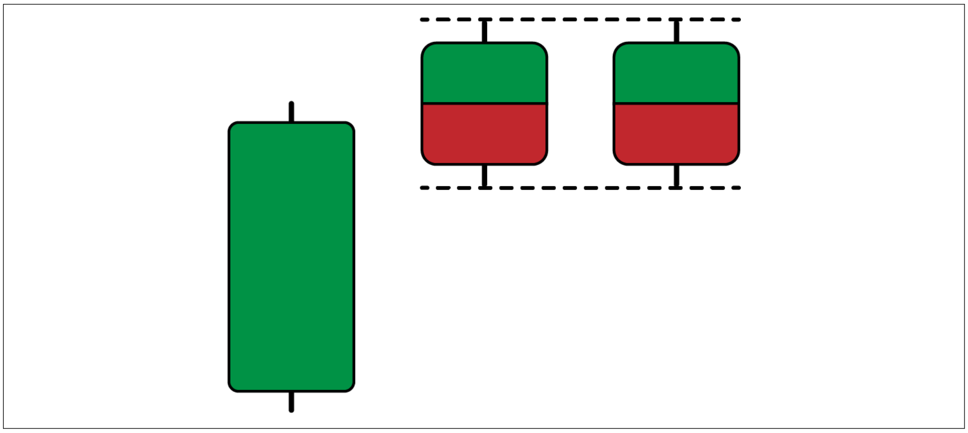


Figure 7-2. A bearish Doppelgänger

Ideally, the bullish reversal is confirmed when the next candle surpasses the similar highs. Similarly, the bearish reversal is confirmed when the next candle breaks the similar lows. This is an optional condition that transforms the pattern into a four-candlestick configuration.

The rationale behind the Doppelgänger pattern lies within the concept of power balance. With similar candles, hesitation looms around the current aggregate direction, which may cause a complete shift of regime. Also, the equality of the lows (in a bullish Doppelgänger) and the highs (in a bearish Doppelgänger) is equivalent to the concept of support and resistance levels, respectively.

To increase the frequency of the signals, you must round the values as you have seen in previous chapters. With FX data, you need four decimals, and with the rest of the back-tested assets, you need zero decimals.

Algorithmically, the real conditions are as follows:

- If the current high equals the previous high, the current low equals the previous low, and the candlestick from two periods ago is bullish, then print 1 in the next row as a representation of a buy signal on the open price.
- If the current high equals the previous high, the current low equals the previous low, and the candlestick from two periods ago is bearish, then print –1 in the next row as a representation of a sell signal on the open price.

Try coding the optional condition that adds a fourth candlestick. As a reminder, this condition seeks to filter false signals, but it may end up filtering too much.

You can code the signal function for the Doppelgänger pattern through the following code snippet:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    data = rounding(data, 4) # Put 0 instead of 4 as of pair 4

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i - 2, close_column] < data[i - 2, open_column] and \
              data[i - 1, close_column] < data[i - 2, open_column] and \
              data[i, high_column] == data[i - 1, high_column] and \
              data[i, low_column] == data[i - 1, low_column]:

                    data[i + 1, buy_column] = 1
```

```
            # Bearish pattern
            elif data[i - 2, close_column] > data[i - 2, open_column] and \
                 data[i - 1, close_column] > data[i - 2, open_column] and \
                 data[i, high_column] == data[i - 1, high_column] and \
                 data[i, low_column] == data[i - 1, low_column]:

                    data[i + 1, sell_column] = -1

        except IndexError:

             pass

    return data
```

Figure 7-3 shows a signal chart on GBPUSD.

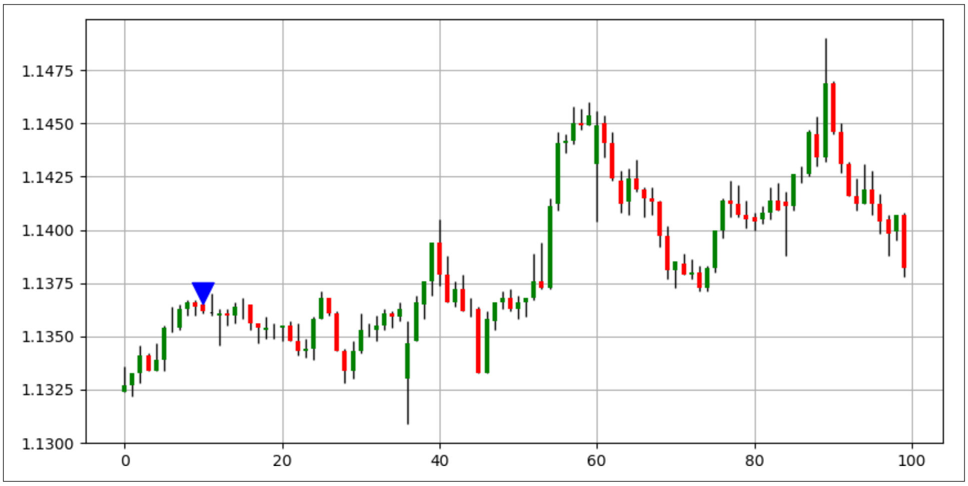


Figure 7-3. Signal chart on GBPUSD

The Doppelgänger pattern is uncommon. Figure 7-4 shows another signal chart, this one on USDCHF.

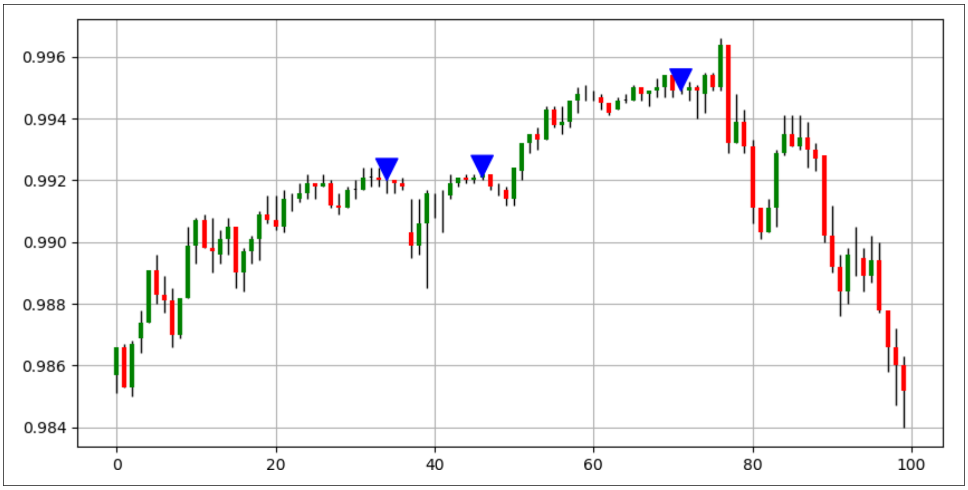


Figure 7-4. Signal chart on USDCHF

Table 7-1 summarizes the performance of the Doppelgänger pattern.

Table 7-1. Doppelgänger pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	46.42%	0.97	1.12	448
USDCHF	50.00%	0.89	0.89	558
GBPUSD	48.05%	1.07	1.16	283
USDCAD	45.52%	0.76	0.91	380
BTCUSD	45.45%	1.78	2.13	99
ETHUSD	50.87%	0.92	0.88	1201
GOLD	49.74%	0.85	0.85	2738
S&P500	50.90%	0.98	0.95	110
FTSE100	48.27%	1.27	1.36	29

To summarize, the pattern shows mixed signals and mixed frequency across asset classes. This may be because of the exit technique, which is unsuited to the pattern as it is mostly concerned with intermediate reversal moves, but to harmonize the backtests across all patterns in the book, the exit condition is the same.

As a reminder, the exit condition is encountering a bullish or bearish signal; otherwise, the trade keeps going.

The Blockade Pattern

This pattern resembles the concept of a market that has found support or resistance and is showing signs of a reversal. The *Blockade* pattern is a four-candlestick configuration with a few conditions that are visually difficult to detect.

Figure 7-5 illustrates a bullish Blockade. The first candlestick must be bearish and followed by three candlesticks with a low equal to or higher than the first candlestick's low but lower than its close price. Finally, the fourth candlestick must be bullish and must have a close price higher than the high price of the first candlestick. These conditions validate the Blockade pattern and generate a bullish signal.

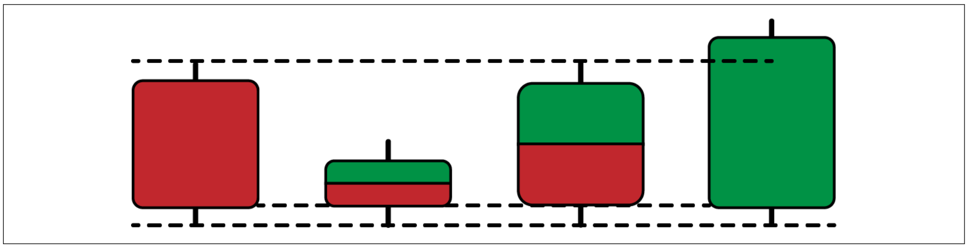


Figure 7-5. A bullish Blockade

Figure 7-6 illustrates a bearish Blockade. The first candlestick must be bullish and followed by three candlesticks with a high price equal to or lower than the first candlestick's low price but lower than its close price. Finally, the fourth candlestick must be bearish and must have a close price lower than the low price of the first candlestick. These conditions validate the Blockade pattern and generate a bearish signal.

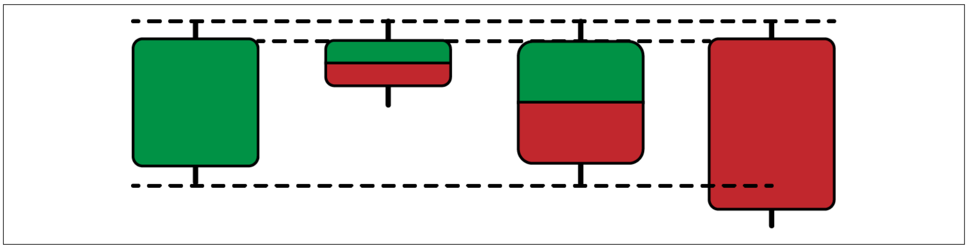


Figure 7-6. A bearish Blockade

The rationale of the Blockade pattern is that the stabilization phase around a support or resistance zone may offer the chance of a reversal confirmed by a directional (fourth) candlestick.

The following code snippet shows how to code the signal function for the Blockade pattern:

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i - 3, close_column] < data[i - 3, open_column] and \
              data[i - 2, close_column] < data[i - 3, open_column] and \
              data[i - 2, low_column] >= data[i - 3, low_column] and \
              data[i - 2, low_column] <= data[i - 3, close_column] and \
              data[i - 1, low_column] >= data[i - 3, low_column] and \
              data[i - 1, low_column] <= data[i - 3, close_column] and \
              data[i, low_column] >= data[i - 3, low_column] and \
              data[i, low_column] <= data[i - 3, close_column] and \
              data[i, close_column] > data[i, open_column] and \
              data[i, close_column] > data[i - 3, high_column]:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i - 3, close_column] > data[i - 3, open_column] and \
                data[i - 2, close_column] > data[i - 3, open_column] and \
                data[i - 2, high_column] <= data[i - 3, high_column] and \
                data[i - 2, high_column] >= data[i - 3, close_column] and\
                data[i - 1, high_column] <= data[i - 3, high_column] and \
                data[i - 1, high_column] >= data[i - 3, close_column] and\
                data[i, high_column] <= data[i - 3, high_column] and \
                data[i, high_column] >= data[i - 3, close_column] and \
                data[i, close_column] < data[i, open_column] and \
                data[i, close_column] < data[i - 3, low_column]:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 7-7 shows a signal chart on GBPUSD. The first thing to notice is the rarity of signals due to the multiple conditions imposed on the algorithm.

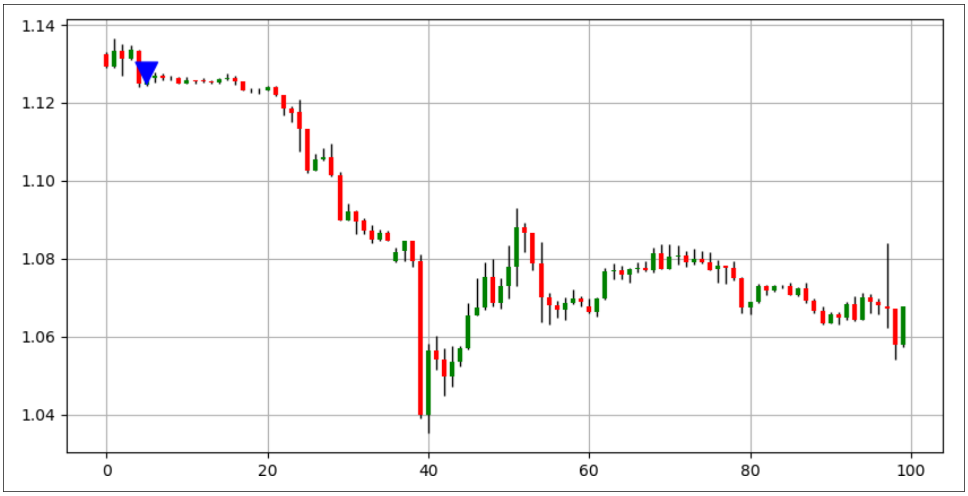


Figure 7-7. Signal chart on GBPUSD

Rare patterns exist everywhere, whether classical or modern, and they are not much different from common patterns results-wise. The important thing is to confirm patterns with other signals and indicators.

Figure 7-8 shows a signal chart on AUDNZD. Notice that, generally, patterns are not identical to their perfect theoretical form as that would make them extremely rare or even impossible to find; therefore, a certain kind of flexibility can be applied from time to time.

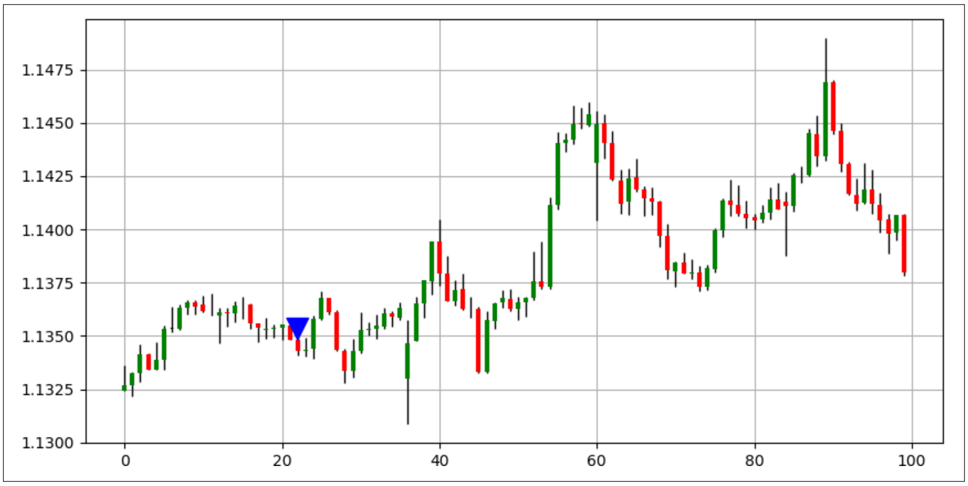


Figure 7-8. Signal chart on AUDNZD

For example, the flexibility measure I have applied here is that I have removed the need for the lows to be exactly the same in the case of a bullish Blockade and for the highs to be exactly the same in the case of a bearish Blockade. Remember, the aim is to detect some patterns and back-test them so as to have an idea on their predictive power, which would then integrate more sophisticated strategies.

Table 7-2 summarizes the performance of the Blockade pattern.

Table 7-2. Blockade pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	40.77%	0.63	0.92	233
USDCHF	47.00%	1.1	1.24	234
GBPUSD	50.19%	1.06	1.06	263
USDCAD	46.15%	0.92	1.07	260
BTCUSD	51.64%	1.79	1.68	182
ETHUSD	45.40%	1.64	1.96	174
GOLD	51.25%	1.13	1.08	240
S&P500	46.66%	0.86	0.98	240
FTSE100	35.71%	1.1	1.98	28

To summarize, the Blockade pattern is a complex formation that can be rare in some markets. It is based on the concept of a market that has found a support or resistance zone and has shown a reversal intention through a bullish or bearish candlestick.

The Euphoria Pattern

At first glance, the *Euphoria* pattern is basically the same as the Three-Candle pattern discussed in the classical trend-following patterns. However, I have found that adding one condition to the Three-Candle pattern is enough to make it into a contrarian configuration. Of course, the conflict remains visually the same, as the extra condition is more easily detected by an algorithm than by a trader.

The extra condition is the obligation that every candlestick must be bigger than the preceding one. The word *bigger* refers to the real range, which is, as mentioned earlier in the book, the difference between the close price and the open price in absolute terms.

Figure 7-9 illustrates a bullish Euphoria pattern. The pattern is composed of three successive bearish candlesticks where each one is bigger than the one before.

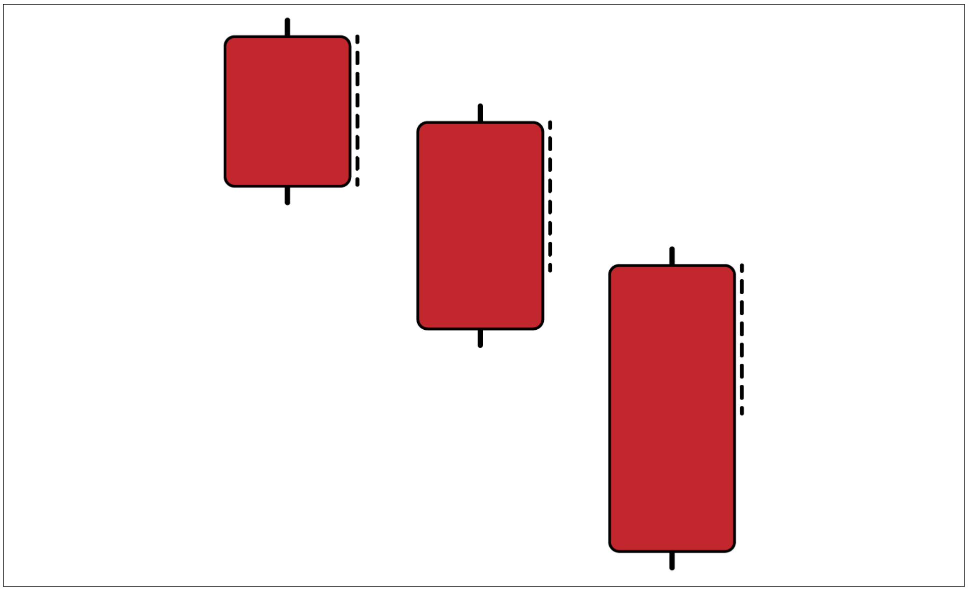


Figure 7-9. A bullish Euphoria pattern

Figure 7-10 illustrates a bearish Euphoria pattern. The pattern is composed of three successive bullish candlesticks where each one is bigger than the one before.

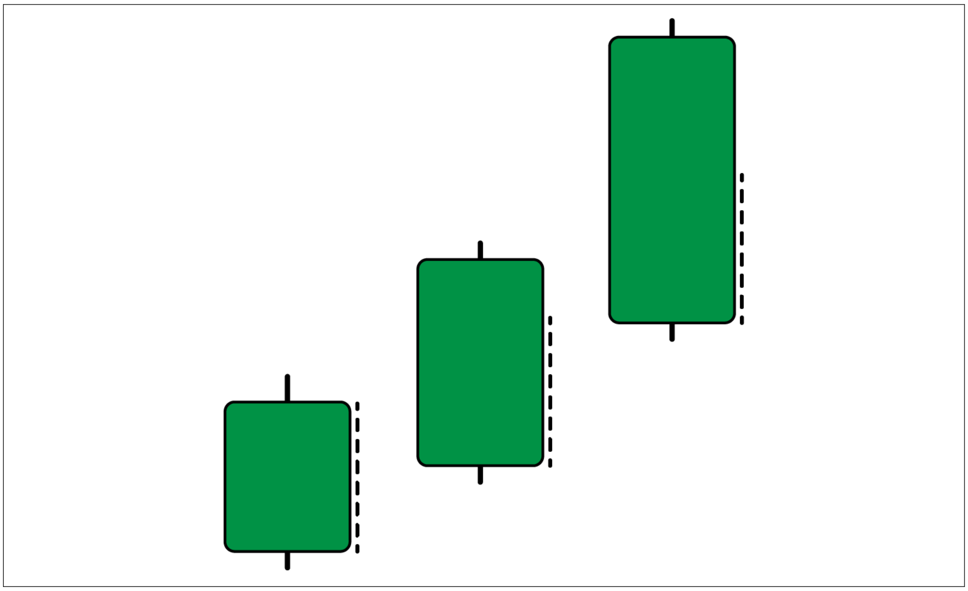


Figure 7-10. A bearish Euphoria pattern

The Three-Candle pattern and the Euphoria pattern closely resemble each other, but the increasing size of the candlesticks is what makes the difference. In the original literature, the Three-Candle pattern does not have a size limit, which can cause a conflict between these two patterns.

Intuitively, you must follow the pattern that makes you more comfortable, and you must also confirm with other techniques and indicators to help you decide whether the market will continue trending in the same direction (Three Candle pattern) or will reverse (Euphoria pattern).

The following code snippet shows how to code the signal function for the Euphoria pattern. You also need to round the values:

```
def signal(data, open_column, close_column, buy_column, sell_column):

    data = add_column(data, 5)

    data = rounding(data, 4) # Put 0 instead of 4 as of pair 4

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, open_column] > data[i, close_column] and \
              data[i - 1, open_column] > data[i - 1, close_column] and \
              data[i - 2, open_column] > data[i - 2, close_column] and \
              data[i, close_column] < data[i - 1, close_column] and \
              data[i - 1, close_column] < data[i - 2, close_column] and \
              (data[i, open_column] - data[i, close_column]) > \
              (data[i - 1, open_column] - data[i - 1, close_column]) and\
              (data[i - 1, open_column] - data[i - 1, close_column]) > \
              (data[i - 2, open_column] - data[i - 2, close_column]):

                  data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, open_column] < data[i, close_column] and \
                data[i - 1, open_column] < data[i - 1, close_column] and \
                data[i - 2, open_column] < data[i - 2, close_column] and \
                data[i, close_column] > data[i - 1, close_column] and \
                data[i - 1, close_column] > data[i - 2, close_column] and\
                (data[i, open_column] - data[i, close_column]) > \
                (data[i - 1, open_column] - data[i - 1, close_column]) and\
                (data[i - 1, open_column] - data[i - 1, close_column]) > \
                (data[i - 2, open_column] - data[i - 2, close_column]):

                  data[i + 1, sell_column] = -1
```

```
            except IndexError:

                pass

        return data
```

Figure 7-11 shows a signal chart on USDCAD. Visually, the Euphoria pattern is more abundant than the Blockade pattern, which may give you more signals but that says nothing about the quality, which I present at the end of the section as usual with the performance summary table.

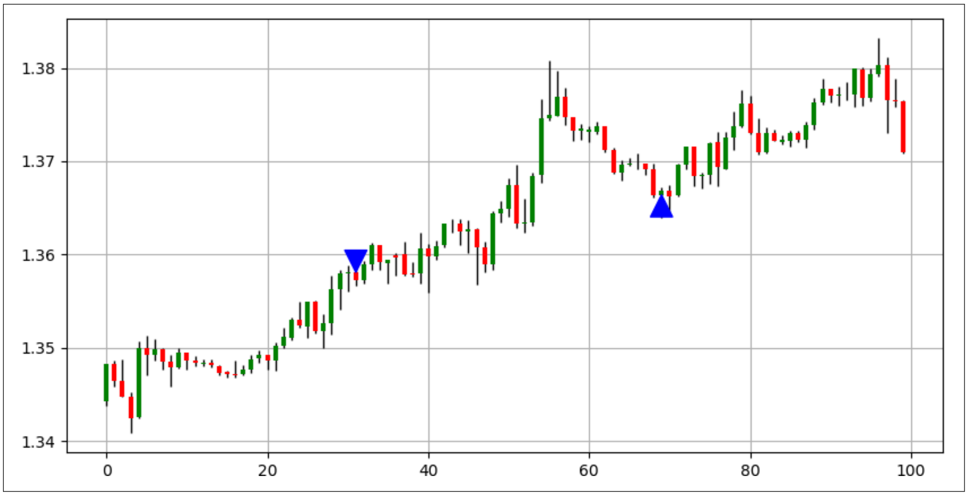


Figure 7-11. Signal chart on USDCAD

The risk with the Euphoria pattern is that it tries to time the end of a greedy phase, which is extremely risky. However, for short moves based on short-term events, the Euphoria pattern outperforms its average performance. For example, during an established bullish trend, the Euphoria pattern has a smaller hit ratio than during sideways markets.

Figure 7-12 shows a signal chart on AUDNZD. Notice that in the congestion period, the pattern was common and had a relatively good quality.

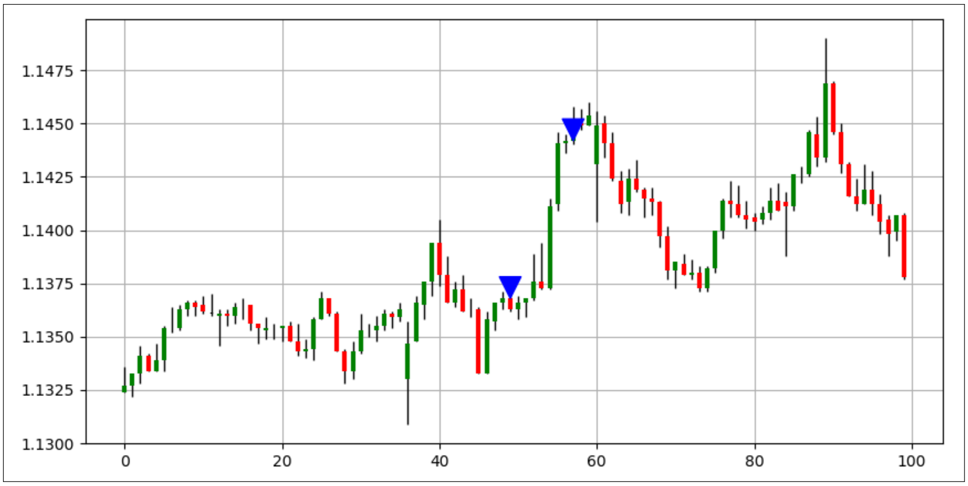


Figure 7-12. Signal chart on AUDNZD

Make sure to also distinguish between the Euphoria pattern and the Double Trouble pattern, as the latter uses a volatility measure that includes highs and lows to account for the continued move. Some candlestick patterns are bound to resemble each other, with a few subtle differences that could change the expected reaction and even the expected targets.

Table 7-3 summarizes the performance of the Euphoria pattern. Note that the usual exit condition is not the optimal exit technique on the Euphoria pattern.

Table 7-3. Euphoria pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	49.17%	1.17	1.21	1513
USDCHF	45.17%	0.99	1.21	1523
GBPUSD	46.69%	1.08	1.24	1741
USDCAD	46.96%	1.00	1.12	1584
BTCUSD	47.24%	1.12	1.24	1327
ETHUSD	45.71%	0.98	1.16	1155
GOLD	44.48%	0.94	1.17	2039
S&P500	46.17%	1.16	1.35	379
FTSE100	46.34%	1.05	1.21	369

To summarize, the Euphoria pattern acts like a brake on an initial acceleration phase based on the outstretched optimism of market participants. Basically, bigger candles mean bigger risks, as fear is more powerful than greed and traders may start unwinding their positions after strong moves.

It is important to distinguish between the Euphoria pattern and the Three-Candle pattern discussed in Chapter 4.

The Barrier Pattern

Basic and very intuitive, the *Barrier* pattern can be considered a simplified version of the Blockade pattern. It is composed of three candles and also borrows from the concept of support and resistance levels. This pattern also uses rounding to stabilize the frequency of signals.

Figure 7-13 illustrates a bullish Barrier. The first two candles must be bearish, and the last candle must be bullish. At the same time, the lows of the three candles must be the same, which implies a support zone. The bullish reversal confirmation comes from the last bullish candle.

A bullish candle implies that there is more demand than supply and thus can act as a confirmation factor after a bearish phase.

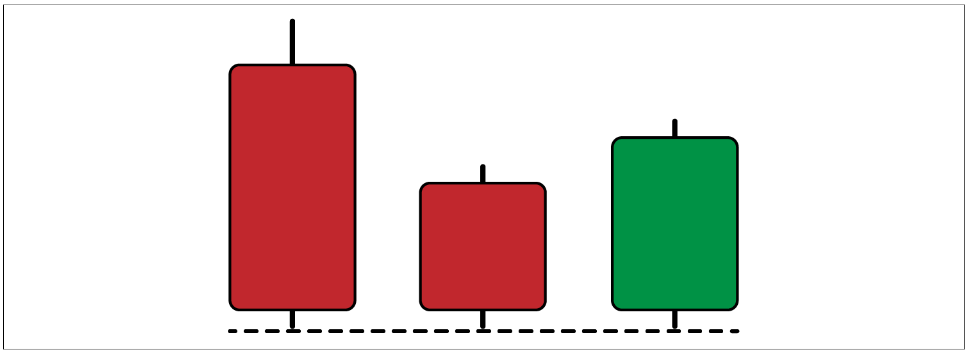


Figure 7-13. A bullish Barrier

Figure 7-14 illustrates a bearish Barrier. The first two candles must be bullish, and the last candle must be bearish. At the same time, the highs of the three candles must be the same, which implies a resistance zone. The bearish reversal confirmation comes from the last bearish candle.

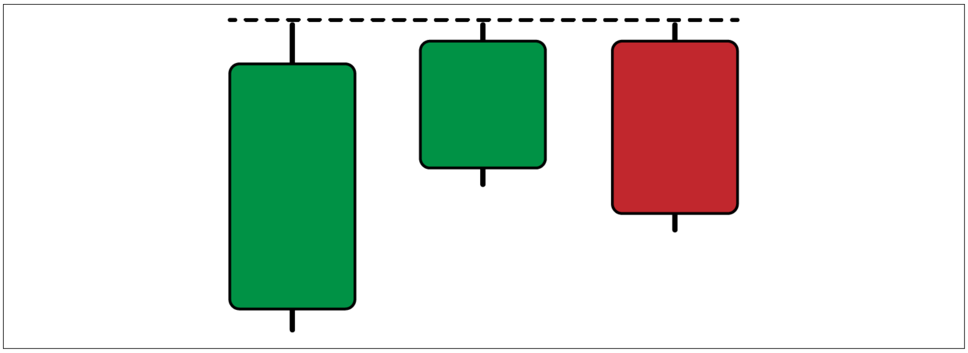


Figure 7-14. A bearish Barrier

Optionally, an extra condition may be imposed. In the case of a bullish Barrier pattern, the last bullish candlestick must have a close price higher than the high price of the middle candlestick. Similarly, in the case of a bearish Barrier pattern, the last bearish candlestick must have a close price lower than the low price of the middle candlestick.

The following code snippet shows how to code the signal function for the Barrier pattern. Notice that I have added the `rounding()` function to maximize the signals. For example, if EURUSD is quoting with five decimals (e.g., 1.05623), then it would be harder to find three candlesticks with the same low/high value than it would be if it is quoting with four decimals (e.g., 1.0562):

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    data = rounding(data, 4) # Put 0 instead of 4 as of pair 4

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] > data[i, open_column] and \
              data[i - 1, close_column] < data[i - 1, open_column] and \
              data[i - 2, close_column] < data[i - 2, open_column] and \
              data[i, low_column] == data[i - 1, low_column] and \
              data[i, low_column] == data[i - 2, low_column]:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] < data[i, open_column] and \
                data[i - 1, close_column] > data[i - 1, open_column] and \
                data[i - 2, close_column] > data[i - 2, open_column] and \
                data[i, high_column] == data[i - 1, high_column] and \
                data[i, high_column] == data[i - 2, high_column]:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 7-15 shows a signal chart on USDCHF. The frequency of signals seems to be acceptable. This can be seen in the performance summary table.

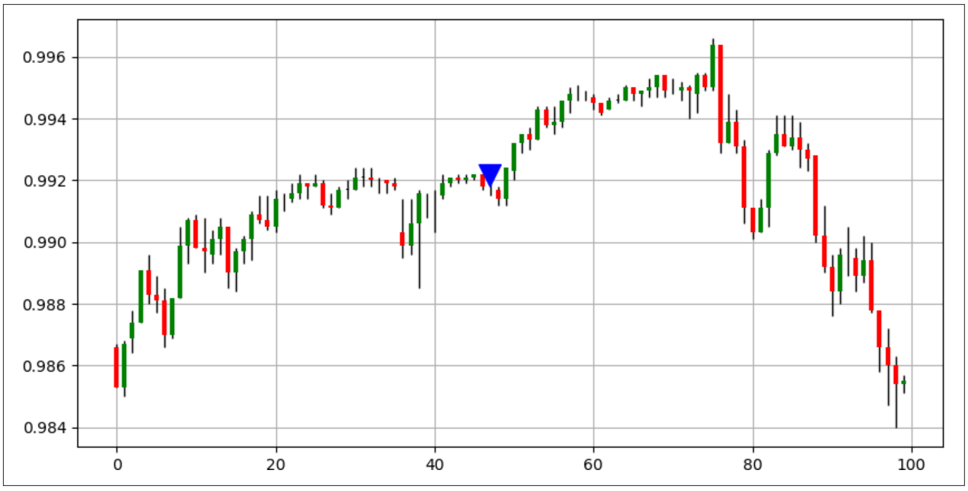


Figure 7-15. Signal chart on USDCHF

Figure 7-16 shows a signal chart on the AUDNZD. In some markets, the signals may be rarer than others. This is a phenomenon seen on most candlestick patterns, and understanding signal frequency allows you to know which patterns to use on which market.

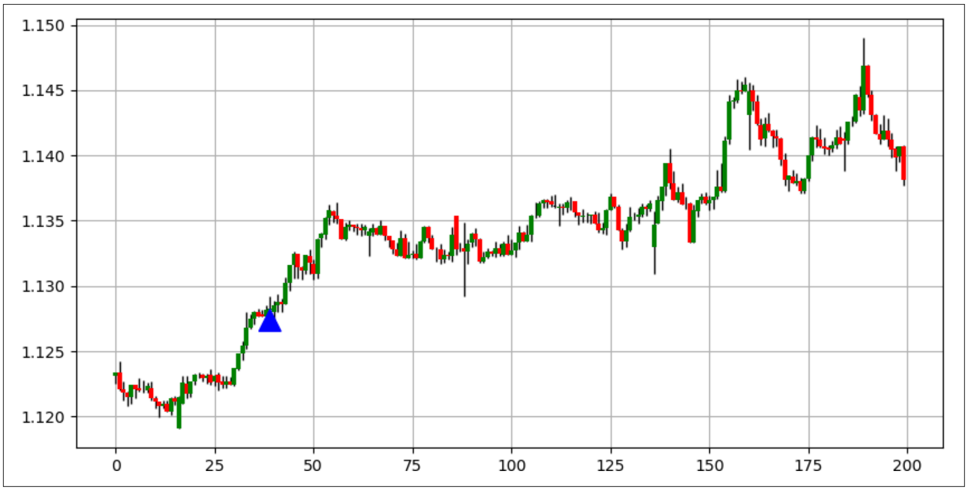


Figure 7-16. Signal chart on the AUDNZD

Table 7-4 summarizes the performance of the Barrier pattern.

Table 7-4. Barrier pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	46.78%	1.05	1.19	109
USDCHF	54.67%	2.26	1.87	139
GBPUSD	43.24%	0.85	1.11	74
USDCAD	52.74%	1.13	1.01	91
BTCUSD	34.61%	0.59	1.12	26
ETHUSD	50.92%	1.11	1.07	163
GOLD	48.39%	1.06	1.13	405
S&P500	53.57%	0.9	0.78	28
FTSE100	37.50%	0.38	0.63	8

To summarize, the Barrier pattern is a simple three-candle configuration that uses rounding to increase its signal frequency due to a severe condition that forces either lows or highs to be equal.

The Mirror Pattern

As the name implies, the *Mirror* pattern is reflection of the past price action with flexible conditions. The idea is that the market is forming a U-turn and should change course.

Figure 7-17 illustrates a bullish Mirror. The first candlestick must be bearish and followed by two candlesticks that have the same highs and lows. Finally, the last candlestick must be bullish and must have a high price equal to the high price of the first candlestick. In the coding part, you will see that there is some flexibility as the theoretical conditions are hard to find.

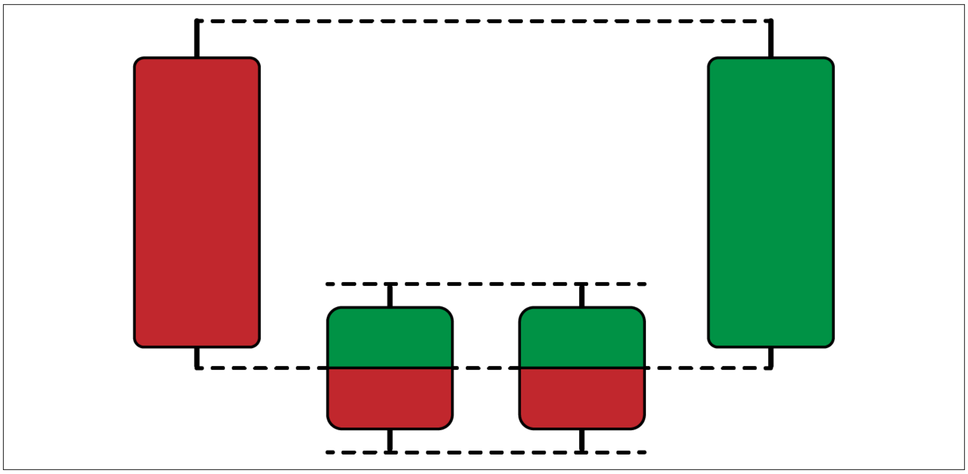


Figure 7-17. A bullish Mirror

The reason there are differences between the theory and the real application of candlestick patterns is that the theory deals with the intuition but does not guarantee the minimum required frequency of patterns detected. In real life, you have to adapt some conditions while retaining the intuition of the pattern.

Figure 7-18 illustrates a bearish Mirror. The first candlestick must be bullish and followed by two candlesticks that have the same highs and lows. Finally, the last candlestick must be bearish and must have a low price equal to the low price of the first candlestick.

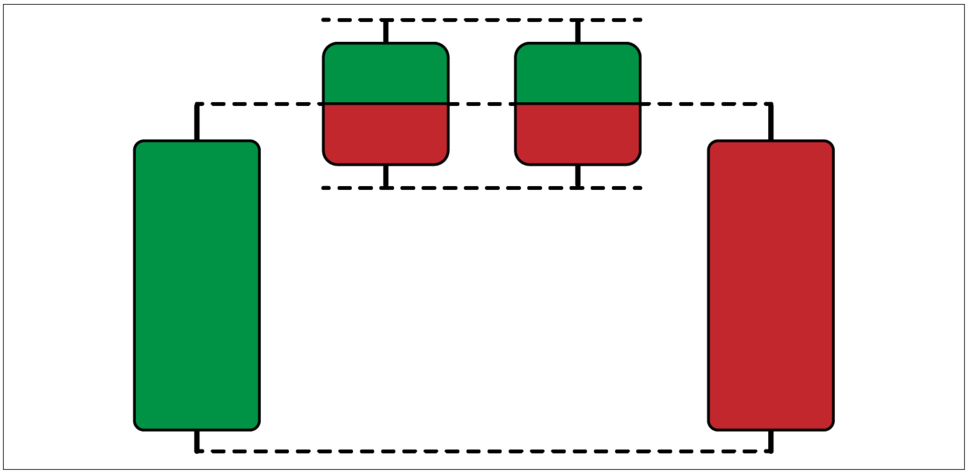


Figure 7-18. A bearish Mirror

The following code snippet shows how to code the signal function for the Mirror pattern. The code assumes flexibility with the highs and lows of the middle candlesticks and creates the condition that equal close prices suffice to validate the pattern (if the close prices of the two middle candlesticks are equal, then the pattern is not invalidated):

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    data = rounding(data, 0) # Put 0 instead of 4 as of pair 4

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i, close_column] > data[i, open_column] and \
              data[i, high_column] == data[i - 3, high_column] and \
              data[i, close_column] > data[i - 1, close_column] and \
              data[i, close_column] > data[i - 2, close_column] and \
              data[i, close_column] > data[i - 3, close_column] and \
              data[i - 3, close_column] < data[i - 3, open_column] and \
              data[i - 1, close_column] == data[i - 2, close_column]:


                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i, close_column] < data[i, open_column] and \
                data[i, low_column] == data[i - 3, low_column] and \
                data[i, close_column] < data[i - 1, close_column] and \
                data[i, close_column] < data[i - 2, close_column] and \
                data[i, close_column] < data[i - 3, close_column] and \
                data[i - 3, close_column] > data[i - 3, open_column] and \
                data[i - 1, close_column] == data[i - 2, close_column]:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 7-19 shows a signal chart on EURCHF.

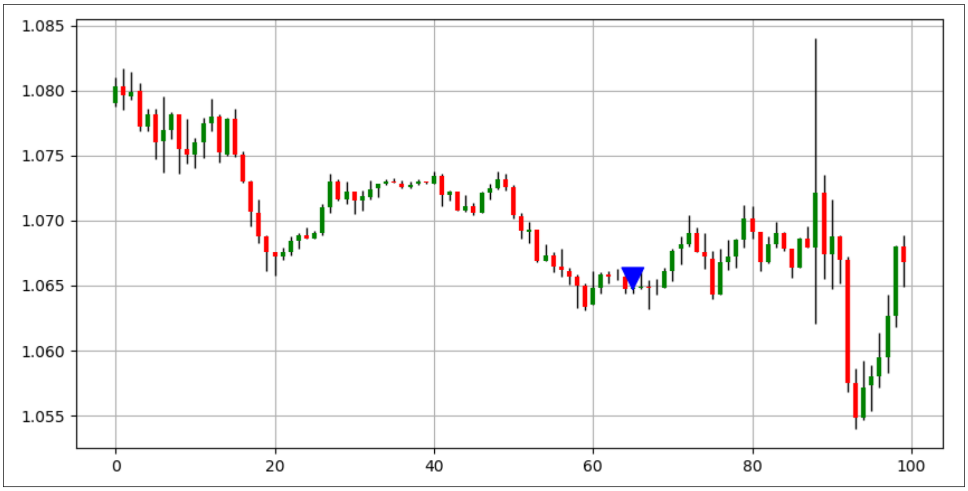


Figure 7-19. Signal chart on EURCHF

Figure 7-20 shows a signal chart on USDCHF. Even with the flexible conditions, it may be hard to find the Mirror pattern.

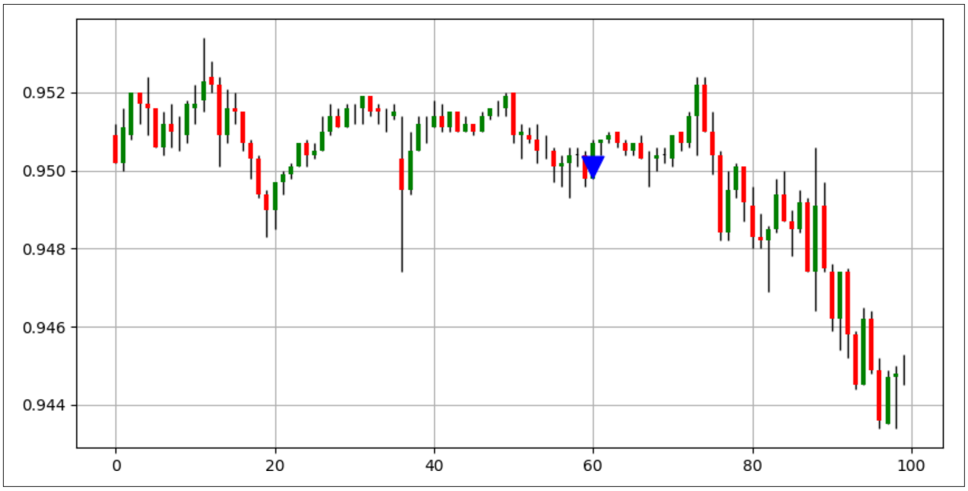


Figure 7-20. Signal chart on USDCHF

Table 7-5 summarizes the performance of the Mirror pattern.

Table 7-5. Mirror pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	45.23%	1.02	1.23	42
USDCHF	41.37%	0.69	0.98	58
GBPUSD	37.50%	0.78	1.3	32
USDCAD	34.21%	0.78	1.5	38
BTCUSD	70.00%	3.02	1.29	10
ETHUSD	44.30%	0.83	1.04	237
GOLD	49.00%	1.14	1.18	500
S&P500	80.00%	5.38	1.34	15

To summarize, the Mirror pattern assumes a U-turn movement where market participants gradually change course. It is a rare pattern, as the conditions to achieve it may be tedious, but the intuition of the pattern is clear. The Mirror pattern can also be considered a continuation of the Doppelgänger pattern.

The Shrinking Pattern

The *Shrinking* pattern is based on the concept of a breakout after a phase of congestion. It is a multicandlestick configuration with the last candlestick confirming the move.

Figure 7-21 illustrates a bullish Shrinking pattern. The first candlestick must be bearish to reflect the proxy of a bearish pressure, and the following three candlesticks may have any color but must be Shrinking in size every time. Finally, the last (fifth) candlestick must be bullish and must surpass the high of the second candlestick.

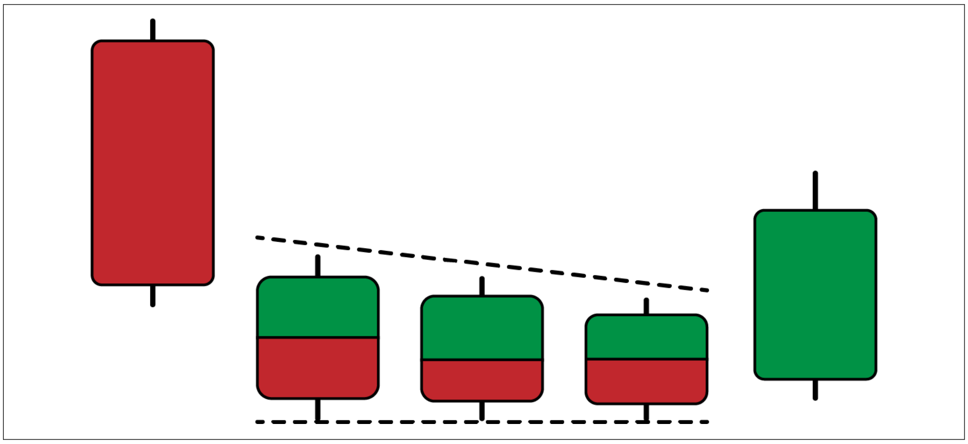


Figure 7-21. A bullish Shrinking pattern

Figure 7-22 illustrates a bearish Shrinking pattern. The first candlestick must be bullish to reflect the proxy of a bullish pressure, and the following three candlesticks may have any color but must be Shrinking in size every time. Finally, the last (fifth) candlestick must be bearish and must break the low of the second candlestick.

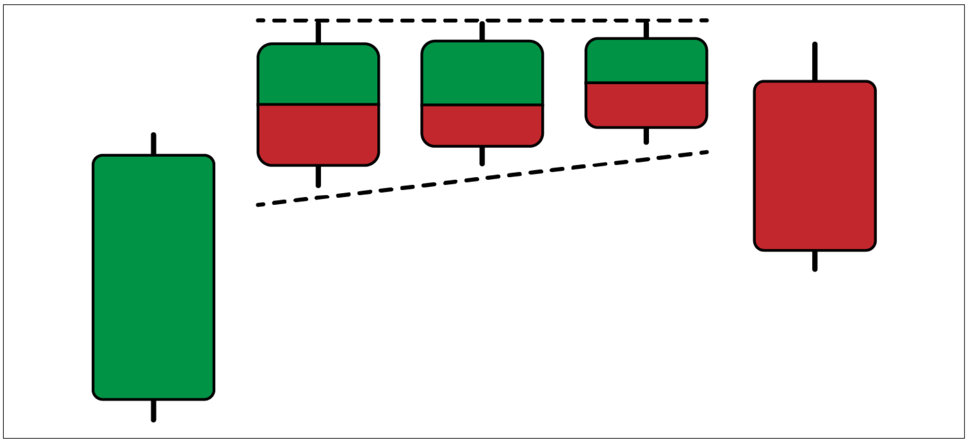


Figure 7-22. A bearish Shrinking pattern

The following code snippet shows how to code the signal function for the Shrinking pattern. I have added some flexibility, which omits the theoretical condition that assumes similar lows for the three middle candles (bullish configuration) and similar highs (bearish configuration), as shown in Figures 7-21 and 7-22 through the dotted horizontal line.

```
def signal(data, open_column, high_column, low_column, close_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    data = rounding(data, 4)

    for i in range(len(data)):

       try:

           # Bullish pattern
           if data[i - 4, close_column] < data[i - 4, open_column] and \
              data[i, close_column] > data[i, open_column] and \
              data[i, close_column] > data[i - 3, high_column] and \
              abs(data[i - 3, close_column] - data[i - 3, open_column]) < \
              abs(data[i - 4, close_column] - data[i - 4, open_column]) \
              and abs(data[i - 2, close_column] - data[i - 2, open_column])\
              < abs(data[i - 3, close_column] - data[i - 3, open_column]) \
              and abs(data[i - 1, close_column] - data[i - 1, \
              open_column]) < abs(data[i - 2, close_column] - data[i - 2, \
              open_column]) and data[i - 1, high_column] < data[i - 2, \
              high_column] and data[i - 2, high_column] < data[i - 3, \
              high_column]:

                    data[i + 1, buy_column] = 1

           # Bearish pattern
           elif data[i - 4, close_column] > data[i - 4, open_column] and \
                data[i, close_column] < data[i, open_column] and \
                data[i, close_column] < data[i - 3, low_column] and \
                abs(data[i - 3, close_column] - data[i - 3, open_column]) \
                < abs(data[i - 4, close_column] - data[i - 4, \
                open_column]) and abs(data[i - 2, close_column] - \
                data[i - 2, open_column]) < abs(data[i - 3, close_column] \
                - data[i - 3, open_column]) and abs(data[i - 1, \
                close_column] - data[i - 1, open_column]) < abs(data[i - \
                2, close_column] - data[i - 2, open_column]) and \
                data[i - 1, low_column] > data[i - 2, low_column] and \
                data[i - 2, low_column] > data[i - 3, low_column]:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 7-23 shows a signal chart on GBPAUD.

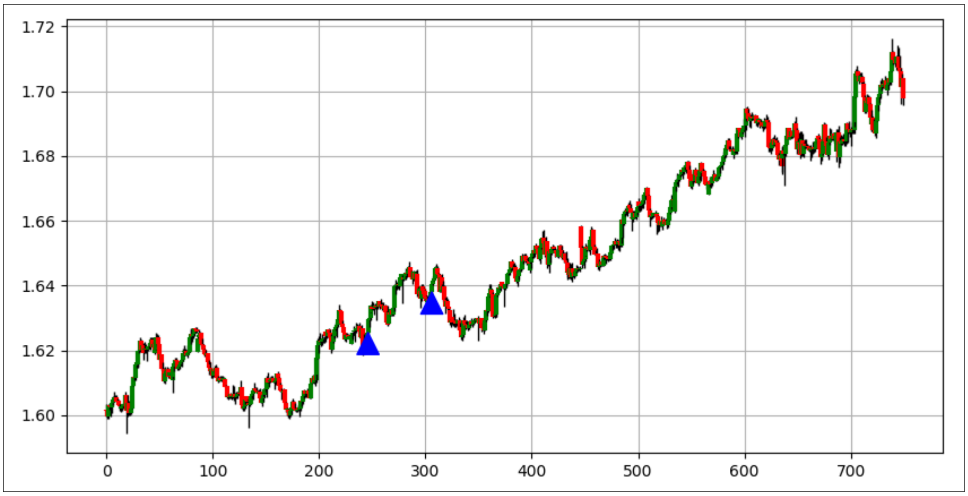


Figure 7-23. Signal chart on GBPAUD

Table 7-6 summarizes the performance of the Shrinking pattern.

Table 7-6. Shrinking pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	37.77%	0.52	0.86	45
USDCHF	37.14%	0.39	0.66	35
GBPUSD	41.37%	0.66	0.94	29
USDCAD	43.47%	1.44	1.87	46
BTCUSD	58.62%	1.24	0.88	29
ETHUSD	55.55%	2.16	1.73	27
GOLD	61.53%	0.73	0.46	26

To summarize, the Shrinking pattern is intuitive and relies on the concept of carefulness, which is manifested through the Shrinking candlesticks before a big candlestick breaks out of the congestion and confirms a new direction.

You are now done learning about the different candlesticks and how to code them in Python. The following chapters deal with how to use these candlesticks and combine them in more sophisticated strategies.

CHAPTER 8

Advanced Candlestick-Charting Systems

So far, you've gotten a grand tour of the many different kinds of patterns that may be of use to you, but candlestick charting can also be creative and doesn't have to stop at the basic construction methods. This chapter presents two candlestick-charting systems that offer diverse alternatives to the original system.

While the original system presents the OHLC data as it is, the two proposed systems covered in this chapter take a different approach to understanding the data and squeezing as much information as possible from it.

The first advanced charting system is Heikin-Ashi, which transforms the OHLC data in order to help trend followers get a less noisy view of the ongoing regime. The second charting system is K's candlesticks, which are simply a smoothed version of OHLC data where candlestick patterns are applied in order to find more signals, all while reducing noise.

The aim of this chapter is to add these two powerful charting systems to your trading framework so that you have a different angle view. Ideally, when checking for patterns using the original candlestick-charting system, you should also check for the same patterns on these two alternative candlestick systems.

Heikin-Ashi System

The Heikin-Ashi system is also known as Heikin-Ashi candlesticks. The term *heikin-ashi* means "average bar" in Japanese, which is intuitive considering the way it is calculated. The main aim of creating a Heikin-Ashi chart is to understand the underlying trend and to filter out the noise caused by random fluctuations.

You can consider the system a noise-canceling technique that leaves you with a smoothed candlestick chart. *Heikin-Ashi* takes the open, high, low, and close prices and transforms them using simple formulas and then charts the results.

By transforming the OHLC data, Heikin-Ashi charts do not represent real values but rather *smoothed* values. For instance, a bearish Heikin-Ashi candlestick does not necessarily represent a true bearish candlestick, and its close price also does not necessarily equal the true candlestick close price. In fact, it is rare that the Heikin-Ashi close price is the real close price.

To calculate the Heikin-Ashi open price, use the following formula:

$$\text{Transformed open price}_i = \frac{\text{Open price}_{i-1} + \text{Close price}_{i-1}}{2}$$

To calculate the Heikin-Ashi high price, use the following formula:

$$\text{Transformed high price}_i = max\left(\text{High price}_i, \text{Transformed open price}_i, \text{Transformed close price}_i\right)$$

To calculate the Heikin-Ashi low price, use the following formula:

$$\text{Transformed low price}_i = min\left(\text{Low price}_i, \text{Transformed open price}_i, \text{Transformed close price}_i\right)$$

To calculate the Heikin-Ashi close price, use the following formula:

$$\text{Transformed close price}_i = \frac{\text{Open price}_i + \text{High price}_i + \text{Low price}_i + \text{Close price}_i}{4}$$

The first thing to notice about Heikin-Ashi charts is that color alternation between candlesticks is less common than in normal candlestick charts due to the smoothing effect, and this is a desirable trait. Figure 8-1 shows the difference between the two systems on AUDNZD. Try figuring out which one is the Heikin-Ashi chart.

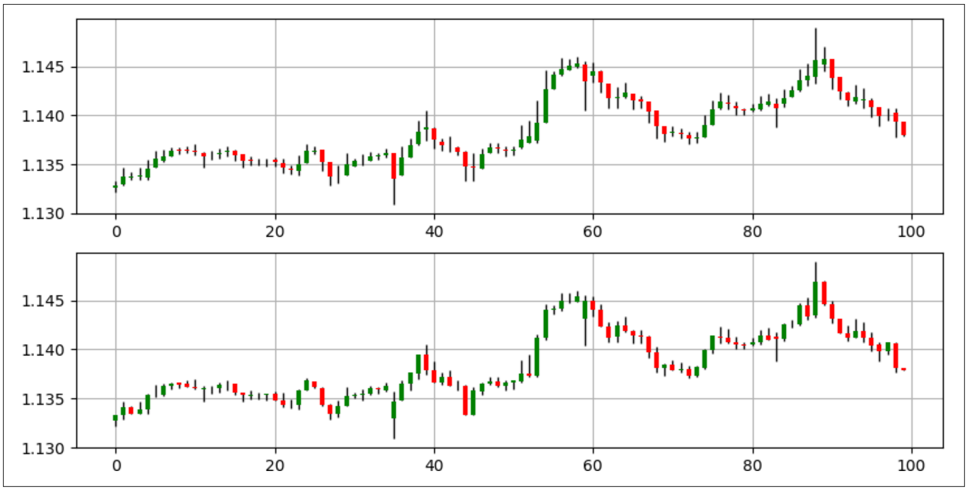


Figure 8-1. On the top, a Heikin-Ashi chart of AUDNZD; on the bottom, a normal candlestick chart of AUDNZD

Notice how relatively easier it is to interpret the Heikin-Ashi chart because green (bullish) candlesticks are clustered together to show that the current trend is bullish and red (bearish) candlesticks are clustered together to show that the current trend is bearish. This is the power of smoothing.

Figure 8-2 shows the difference between the two systems on GBPAUD.

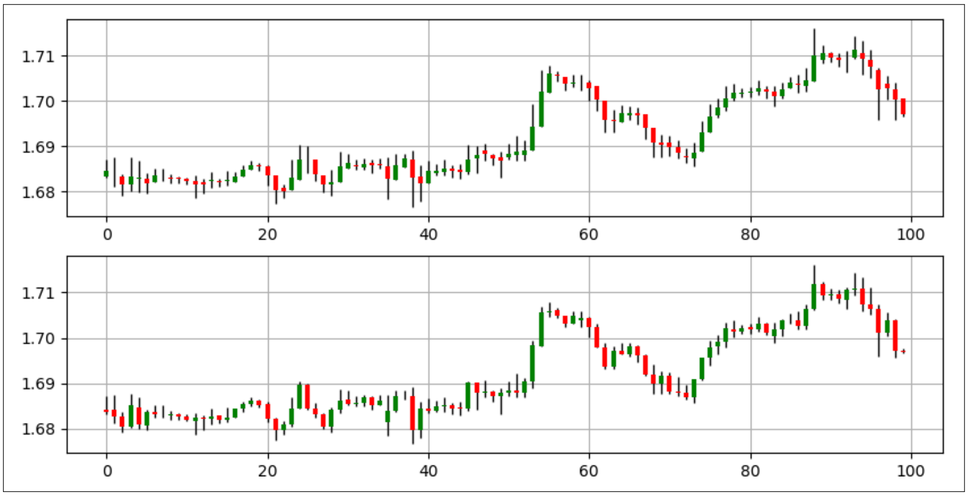


Figure 8-2. On the top, a Heikin-Ashi chart of the GBPAUD; on the bottom, a normal candlestick chart of the GBPAUD

There are some limitations to Heikin-Ashi charts:

- The OHLC Heikin-Ashi values are not real prices as they have been transformed.
- In flat markets, Heikin-Ashi candlesticks will alternate in color, which hinders their ability to give signals.
- A change in color can sometimes be lagging, which may mean part of the move has already occurred by the time it is detected.

The way you can create a Heikin-Ashi candlestick chart is to use the OHLC of the four columns of the array, apply the transformation, and output the results in the next four columns of the array. The function is as follows:

```python
def heikin_ashi(data, open_column, high_column, low_column, close_column,
                position):

    data = add_column(data, 4)

    # Heikin-Ashi Open
    try:
        for i in range(len(data)):
            data[i, position] = (data[i - 1, open_column] +
                                 data[i - 1, close_column]) / 2
    except:
        pass

    # Heikin-Ashi High
    for i in range(len(data)):
        data[i, position + 1] = max(data[i, position],
                                    data[i, position + 3],
                                    data[i, high_column])

    # Heikin-Ashi Low
    for i in range(len(data)):
        data[i, position + 2] = min(data[i, position],
                                    data[i, position + 3],
                                    data[i, low_column])

    # Heikin-Ashi Close
    for i in range(len(data)):
        data[i, position + 3] = (data[i, open_column] +
                                 data[i, high_column] +
                                 data[i, low_column] +
                                 data[i, close_column]) / 4

    return data
```

Let's now see how to apply some of the previously seen candlesticks on Heikin-Ashi charts, which should give a diversification effect.

Detecting the Doji Pattern

As a reminder, the Doji pattern is an indecision and contrarian configuration characterized by the following conditions:

- If the current close price is greater than the current open price, the previous close price equals the open price, and the prior close price is lower than the prior open price, then a bullish Doji has been printed.
- If the current close price is lower than the current open price, the previous close price equals the open price, and the prior close price is greater than the prior open price, then a bearish Doji has been printed.

Figure 8-3 shows a signal chart with the same signals superimposed on a Heikin-Ashi chart (top) and a normal candlestick chart (bottom).

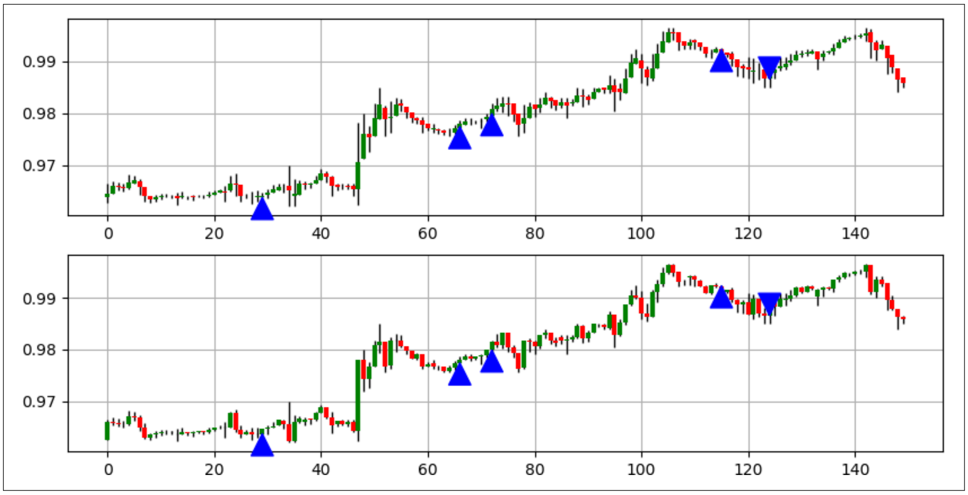


Figure 8-3. On the top, signals (estimated) on Heikin-Ashi candlesticks; on the bottom, the same signals on real candlesticks. The currency pair is USDCHF.

The bottom panel in Figure 8-3 shows where the real signals are relative to real OHLC data. In reality, you can see that there is little difference between the position of estimated and real signals.

Figure 8-4 shows a signal chart on USDCAD. Don't forget to round the values according to the analyzed asset.

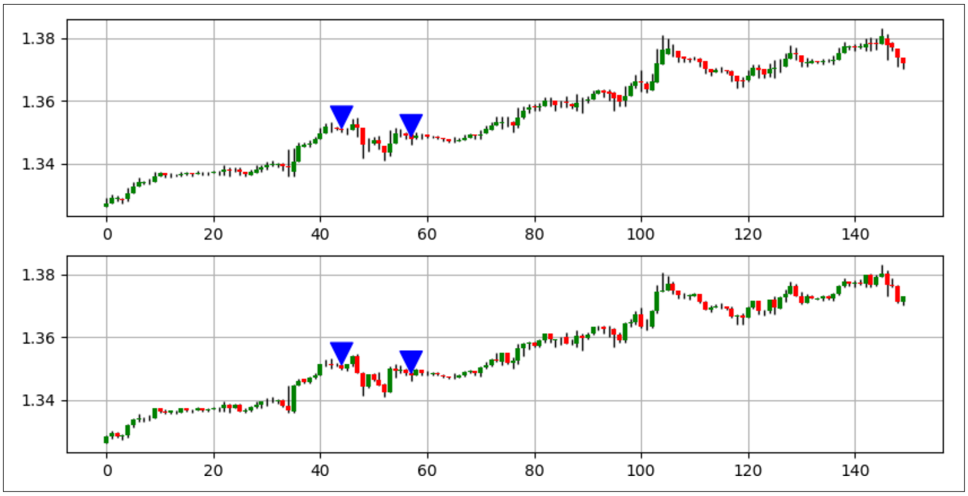


Figure 8-4. On the top, signals (estimated) on Heikin-Ashi candlesticks; on the bottom, the same signals on real candlesticks. The asset is USDCAD.

Let's check the performance of the pattern (see Table 8-1). It is important to calculate the performance metrics using real OHLC data and not the transformed values so that you do not bias your system.

Table 8-1. Heikin-Ashi chart with Doji pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	49,67%	1,07	1,08	2126
USDCHF	47,30%	1,07	1,19	2359
GBPUSD	48,73%	1,03	1,08	1660
USDCAD	48,18%	0,97	1,04	1982
BTCUSD	46,54%	0,84	0,97	434
ETHUSD	44,91%	0,81	0,99	1534
GOLD	44,93%	1,05	1,29	3394
S&P500	50,12%	0,85	0,84	401
FTSE100	52,10%	1,09	1,00	261

Detecting the Tasuki Pattern

As a reminder, the Tasuki pattern is a trend-following configuration characterized by the following conditions:

- If the close price from two periods ago is greater than the open price from two periods ago, the open price from one period ago is greater than the close two periods ago, the close price from one period ago is greater than the open price from one period ago, and the current close price is greater than the close price two periods ago, then a bullish Tasuki has been printed.

- If the close price from two periods ago is lower than the open price from two periods ago, the open price from one period ago is lower than the close two periods ago, the close price from one period ago is lower than the open price from one period ago, and the current close price is lower than the close price two periods ago, then a bearish Tasuki has been printed.

Figure 8-5 shows a signal chart on GBPAUD.

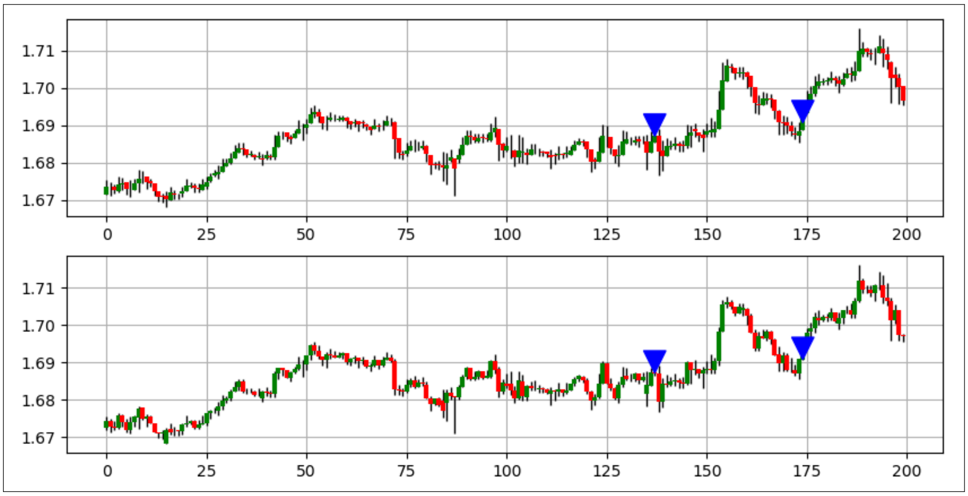


Figure 8-5. On the top, signals (estimated) on Heikin-Ashi candlesticks; on the bottom, the same signals on real candlesticks. The currency pair is GBPAUD.

To create the double-plot signal chart, use the following code:

```
def candlestick_double_plot(data, buy_column, sell_column, window = 250):

    fig, ax = plt.subplots(2, figsize = (10, 5))

    sample = data[-window:, ]

    for i in range(len(sample)):

        ax[0].vlines(x = i, ymin = sample[i, 6], ymax = sample[i, 5],
                     color = 'black', linewidth = 1)

        if sample[i, 7] > sample[i, 4]:

            ax[0].vlines(x = i, ymin = sample[i, 4], ymax = sample[i, 7],
                         color = 'mediumseagreen', linewidth = 3)

        if sample[i, 7] < sample[i, 4]:
            ax[0].vlines(x = i, ymin = sample[i, 7], ymax = sample[i, 4],
                         color = 'maroon', linewidth = 3)

        if sample[i, 7] == sample[i, 4]:

            ax[0].vlines(x = i, ymin = sample[i, 7], ymax = sample[i, 4] +
                         0.00005, color = 'black', linewidth = 1.00)

        if sample[i, buy_column] == 1:

            x = i
            y = sample[i, 0]

            ax[0].annotate(' ', xy = (x, y),
                        arrowprops = dict(width = 9, headlength = 11,
                                          headwidth = 11, facecolor =
                                          'green', color = 'green'))

        elif sample[i, sell_column] == -1:

            x = i
            y = sample[i, 0]

            ax[0].annotate(' ', xy = (x, y),
                        arrowprops = dict(width = 9, headlength = -11,
                                          headwidth = -11, facecolor =
                                          'red', color = 'red'))

    ax[0].grid()

    for i in range(len(sample)):

        ax[1].vlines(x = i, ymin = sample[i, 2], ymax = sample[i, 1],
                     color = 'black', linewidth = 1)

        if sample[i, 3] > sample[i, 0]:

            ax[1].vlines(x = i, ymin = sample[i, 0], ymax = sample[i, 3],
                         color = 'mediumseagreen', linewidth = 3)

        if sample[i, 3] < sample[i, 0]:

            ax[1].vlines(x = i, ymin = sample[i, 3], ymax = sample[i, 0],
                         color = 'maroon', linewidth = 3)

        if sample[i, 3] == sample[i, 0]:

            ax[1].vlines(x = i, ymin = sample[i, 3], ymax = sample[i, 0] +
                         0.00005, color = 'black', linewidth = 1.00)

        if sample[i, buy_column] == 1:
            x = i
            y = sample[i, 0]

            ax[1].annotate(' ', xy = (x, y),
                        arrowprops = dict(width = 9, headlength = 11,
                                          headwidth = 11, facecolor =
                                          'green', color = 'green'))

        elif sample[i, sell_column] == -1:

            x = i
            y = sample[i, 0]

            ax[1].annotate(' ', xy = (x, y),
                        arrowprops = dict(width = 9, headlength = -11,
                                          headwidth = -11, facecolor =
                                          'red', color = 'red'))


    ax[1].grid()
```

Figure 8-6 shows a signal chart on EURCHF.

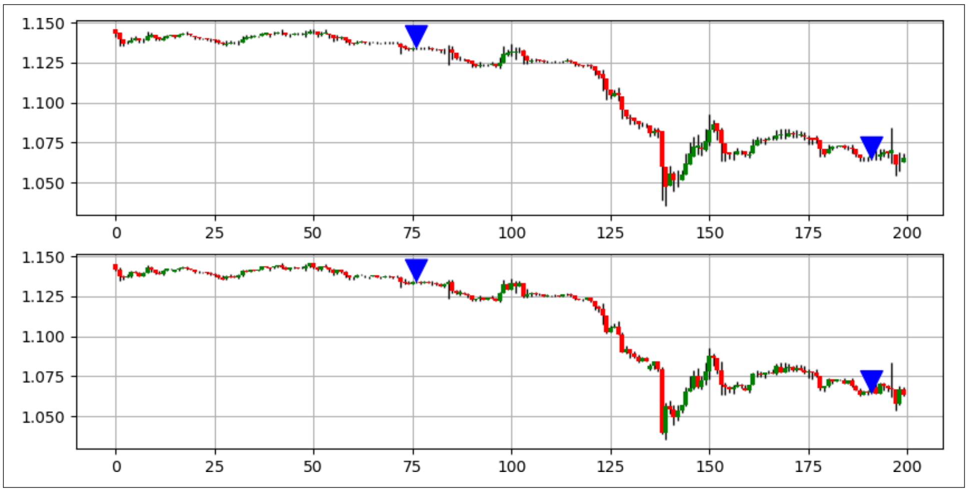


Figure 8-6. On the top, signals (estimated) on Heikin-Ashi candlesticks; on the bottom, the same signals on real candlesticks. The currency pair is EURCHF.

Table 8-2 summarizes the performance metrics of an algorithm that scans and trades the Tasuki pattern on Heikin-Ashi charts.

Table 8-2. Heikin-Ashi chart with Tasuki pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	51.04%	0.83	0.80	384
USDCHF	50.00%	0.91	0.91	468
GBPUSD	50.11%	0.98	0.97	441
USDCAD	50.90%	0.90	0.87	442
BTCUSD	50.00%	0.94	0.93	248
ETHUSD	52.14%	0.76	0.69	257
GOLD	47.04%	0.92	1.04	372
S&P500	55.76%	1.12	0.88	52
FTSE100	44.64%	1.20	1.49	56

Detecting the Euphoria Pattern

As a reminder, the Euphoria pattern is a contrarian configuration characterized by the following conditions:

- If the current bearish candlestick is greater in real size than the previous bearish candlestick and the previous bearish candlestick is greater in real size than the the prior bearish candlestick, then a bullish Euphoria is printed.
- If the current bullish candlestick is greater in real size than the previous bullish candlestick and the previous bullish candlestick is greater in real size than the the prior bullish candlestick, then a bullish Euphoria is printed.

Figure 8-7 shows a signal chart on AUDNZD.

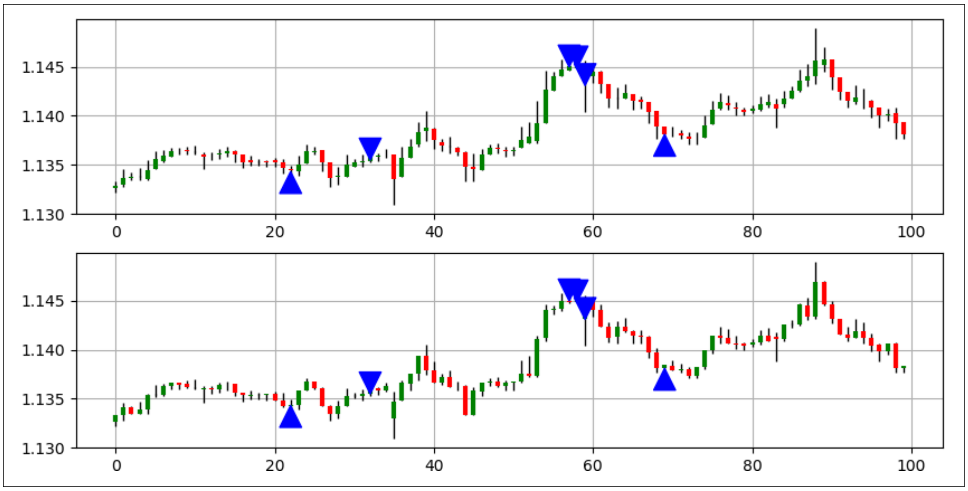


Figure 8-7. On the top, signals (estimated) on Heikin-Ashi candlesticks; on the bottom, the same signals on real candlesticks. The currency pair is AUDNZD.

Using the `rounding()` function with the Euphoria pattern may help filter out some signals if you are looking to decrease the frequency.

Figure 8-8 shows a signal chart on EURGBP. It is always interesting to keep the trend on your side. This is a technique that you will see in the chapter dealing with strategies. The main idea is that it is better to take into consideration a bullish reversal pattern during a bullish trend than during a bearish trend. This is because of the invisible trend hand, which helps push the prices toward the aggregate direction.

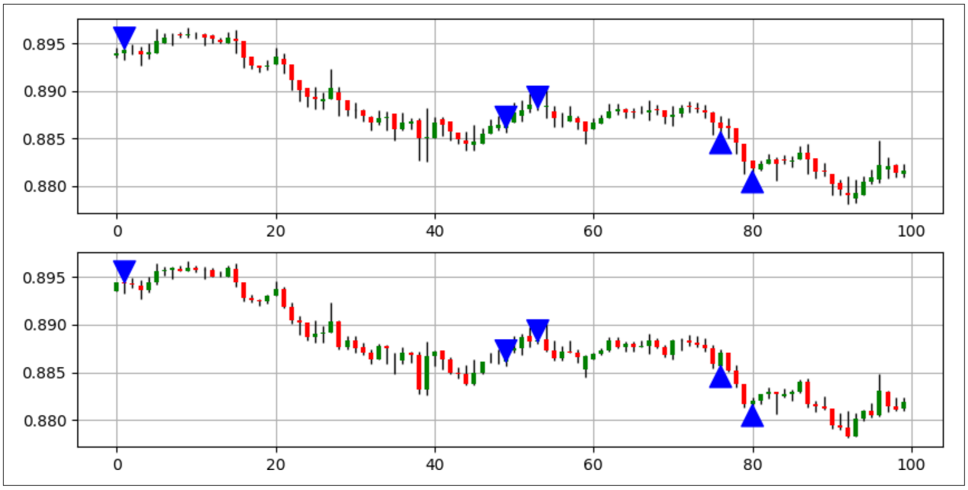


Figure 8-8. On the top, signals (estimated) on Heikin-Ashi candlesticks; on the bottom, the same signals on real candlesticks. The currency pair is EURGBP.

Table 8-3 summarizes the performance metrics of an algorithm that scans and trades the Euphoria pattern on Heikin-Ashi charts.

Table 8-3. Heikin-Ashi chart with Euphoria pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	47.74%	1.01	1.11	2572
USDCHF	45.64%	0.91	1.09	2583
GBPUSD	46.52%	1.00	1.15	2990
USDCAD	48.25%	1.08	1.16	2918
BTCUSD	48.61%	1.04	1.1	2055
ETHUSD	43.83%	0.8	1.02	933
GOLD	49.17%	1.06	1.09	543
S&P500	46.28%	1.08	1.25	417
FTSE100	44.46%	0.94	1.17	497

Detecting the Double Trouble Pattern

As a reminder, the Double Trouble pattern is a trend-following configuration characterized by the following conditions:

- The second bullish candlestick must be at least double the size (from high to low) of the previous candlestick's 10-period ATR.
- The second bearish candlestick must be at least double the size (from high to low) of the previous candlestick's 10-period ATR.

Figure 8-9 shows a signal chart on USDCHF.

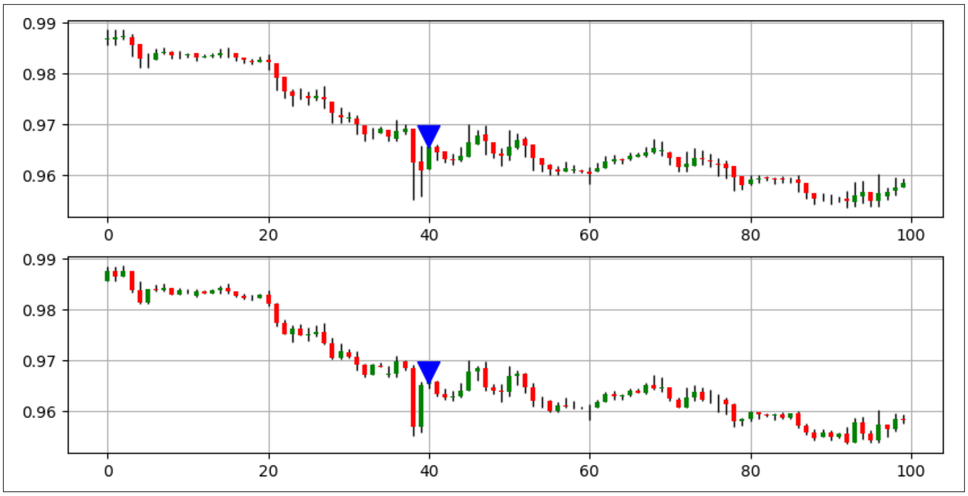


Figure 8-9. On the top, signals (estimated) on Heikin-Ashi candlesticks; on the bottom, the same signals on real candlesticks. The currency pair is USDCHF.

Figure 8-10 shows a signal chart on the USDJPY.

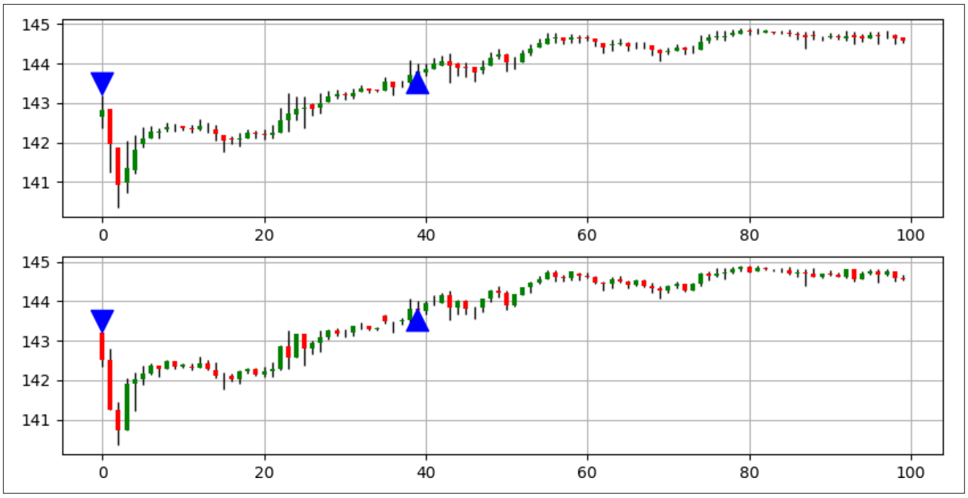


Figure 8-10. On the top, signals (estimated) on Heikin-Ashi candlesticks; on the bottom, the same signals on real candlesticks. The asset is USDJPY.

Table 8-4 summarizes the performance metrics of an algorithm that scans and trades the Double Trouble pattern on Heikin-Ashi charts.

Table 8-4. Heikin-Ashi chart with Double Trouble pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	57.62%	0.92	0.67	1135
USDCHF	56.49%	1.08	0.84	110
GBPUSD	53.72%	0.89	0.77	1167
USDCAD	56.14%	1.02	0.80	1058
BTCUSD	62.84%	1.19	0.71	716
ETHUSD	61.00%	1.04	0.67	659
GOLD	57.00%	0.96	0.73	1021
S&P500	57.40%	1.36	1.00	162
FTSE100	54.13%	0.95	0.80	133

K's Candlesticks System

K's candlesticks aim to further smooth out the OHLC values to gain a better understanding of the underlying trend while retaining the four elementary pieces of data.[1] Therefore, K's candlestick system calculates a simple moving average of OHLC data using a default lookback period of three.

A period of three represents three hours in hourly charts and three days in daily charts.

To calculate the K's open price, use the following formula:

$$\text{K's open price}_i = \frac{\text{Open price}_i + \text{Open price}_{i-1} + \text{Open price}_{i-2}}{3}$$

To calculate the K's high price, use the following formula:

$$\text{K's high price}_i = \frac{\text{High price}_i + \text{High price}_{i-1} + \text{High price}_{i-2}}{3}$$

To calculate the K's low price, use the following formula:

$$\text{K's low price}_i = \frac{\text{Low price}_i + \text{Low price}_{i-1} + \text{Low price}_{i-2}}{3}$$

To calculate the K's close price, use the following formula:

$$\text{K's close price}_i = \frac{\text{Close price}_i + \text{Close price}_{i-1} + \text{Close price}_{i-2}}{3}$$

Figure 8-11 shows the difference between a regular candlestick chart and a K's candlesticks chart.

1 The OHLC data.

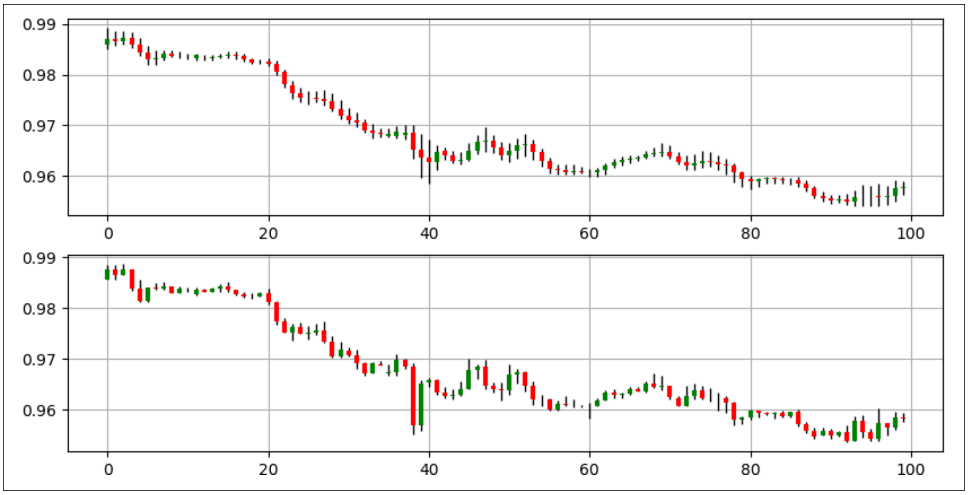


Figure 8-11. On the top, a K's candlesticks chart of EURUSD; on the bottom, a normal candlestick chart of EURUSD

You should notice that K's candlesticks are even smoother than Heikin-Ashi candlesticks. This is because the smoothing period is 3 in K's candlesticks while Heikin-Ashi has a smoothing period of 1.

Figure 8-12 shows the difference between a regular AUDNZD candlestick chart and a K's candlesticks chart on the same currency pair.

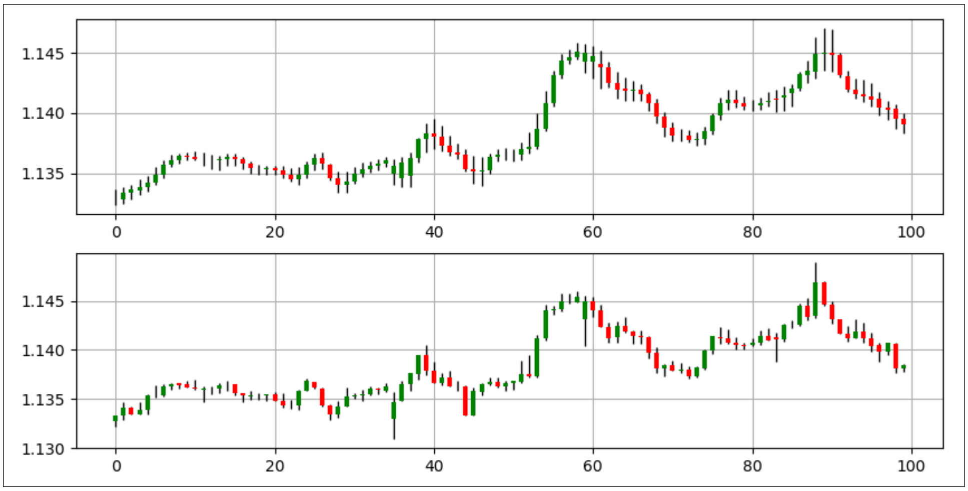


Figure 8-12. On the top, a K's candlesticks chart of AUDNZD; on the bottom, a normal candlestick chart of AUDNZD

The limitations of K's candlesticks charts are as follows:

1. The OHLC K's candlesticks values are not real prices as they are transformed.
2. In flat markets, K's candlesticks will alternate in color, which hinders their ability to generate a signal.
3. The lag effect is even bigger than the one related to Heikin-Ashi candlesticks. However, it is not big enough to completely offset the benefits of the view.

The function you can use to code K's candlesticks is as follows:

```python
def k_candlesticks(data, open_column, high_column, low_column, 
                   close_column, lookback, position):
    
    data = add_column(data, 4)
    
    # Averaging the open price
    data = ma(data, lookback, open_column, position)
    
    # Averaging the high price
    data = ma(data, lookback, high_column, position + 1)
    
    # Averaging the low price
    data = ma(data, lookback, low_column, position + 2)
    
    # Averaging the close price
    data = ma(data, lookback, close_column, position + 3)
    
    return data
```

Let's see now how to apply some of the previously seen candlesticks on K's candlesticks charts.

Detecting the Doji Pattern

The Doji pattern is probably one of the best patterns to detect with K's candlesticks as it is correlated with many forms of short-term and long-term reversals. Figure 8-13 shows a signal chart on GBPUSD.

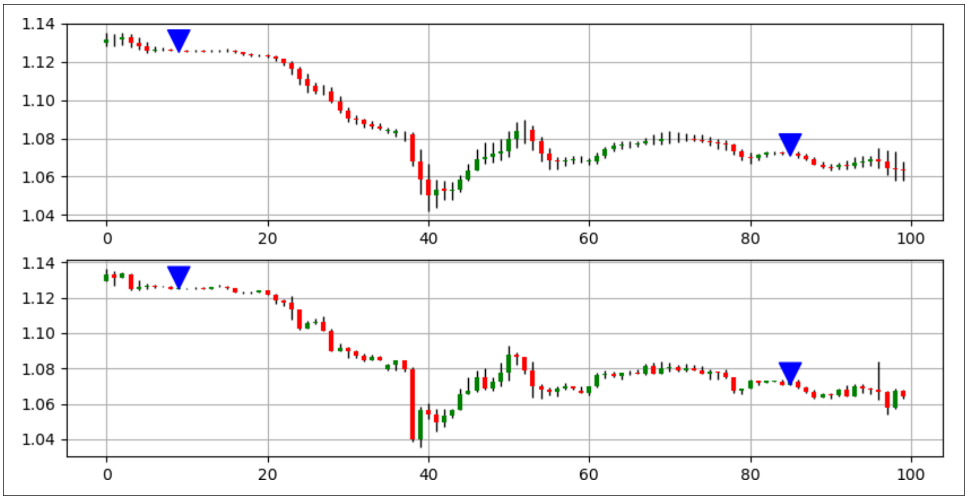


Figure 8-13. On the top, signals (estimated) on K's candlesticks; on the bottom, the same signals on real candlesticks. The currency pair is GBPUSD.

Figure 8-14 shows a signal chart on USDCAD. On ranging markets, reversal patterns work relatively better.

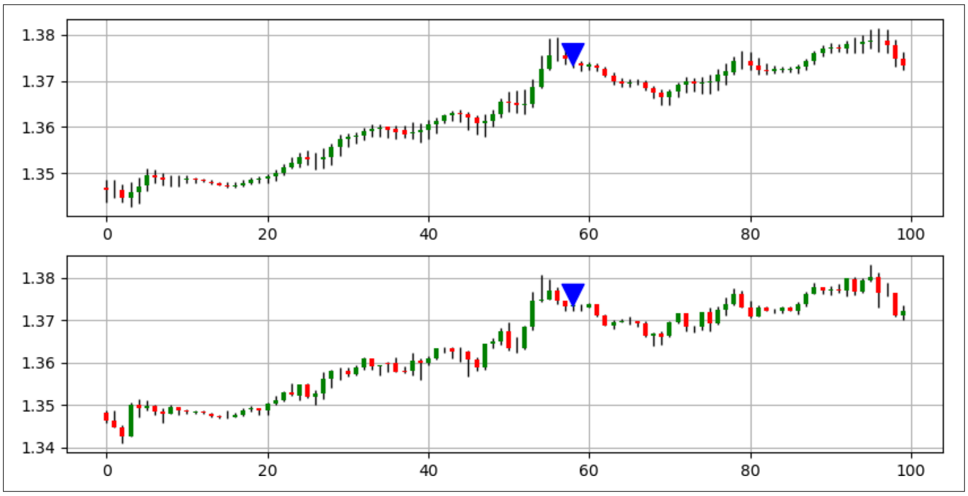


Figure 8-14. On the top, signals (estimated) on K's candlesticks; on the bottom, the same signals on real candlesticks. The currency pair is USDCAD.

Table 8-5 summarizes the performance metrics of an algorithm that scans and trades the Doji pattern on K's candlesticks charts. Remember that one of the keys to improving performance is optimizing the entry and exit techniques.

Table 8-5. K's candlesticks with Doji pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	43.66%	0.81	1.05	2352
USDCHF	43.80%	0.90	1.15	2479
GBPUSD	46.19%	1.07	1.24	1840
USDCAD	45.07%	1.02	1.24	2143
BTCUSD	44.23%	0.90	1.13	486
ETHUSD	45.93%	1.20	1.41	1132
GOLD	45.60%	0.89	1.06	2445
S&P500	42.70%	0.89	1.19	384
FTSE100	46.44%	1.19	1.37	267

Detecting the Tasuki Pattern

Figure 8-15 shows a signal chart on AUDNZD.

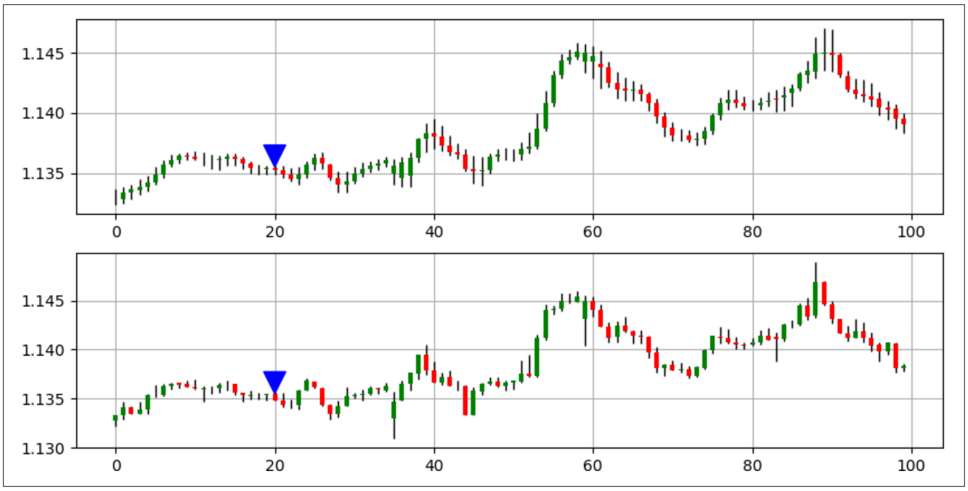


Figure 8-15. On the top, signals (estimated) on K's candlesticks; on the bottom, the same signals on real candlesticks. The currency pair is AUDNZD.

Table 8-6 summarizes the performance metrics of an algorithm that scans and trades the Tasuki pattern on K's candlesticks charts.

Table 8-6. K's candlesticks with Tasuki pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	50.00%	1.11	1.11	50
USDCHF	51.68%	0.83	0.78	89
GBPUSD	56.66%	1.65	1.26	60
USDCAD	50.79%	1.19	1.15	63
BTCUSD	43.24%	0.78	1.02	74
ETHUSD	60.38%	0.81	0.53	53
GOLD	55.38%	1.01	0.81	130
S&P500	47.05%	0.77	0.86	17
FTSE100	50.00%	1.80	1.8	24

Detecting the Euphoria Pattern

Figure 8-16 shows a signal chart on AUDNZD.

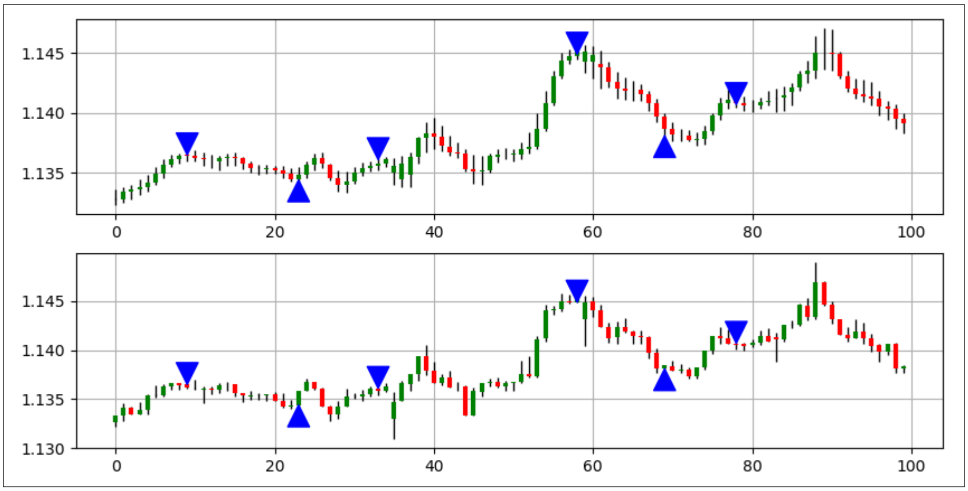


Figure 8-16. On the top, signals (estimated) on K's candlesticks; on the bottom, the same signals on real candlesticks. The currency pair is AUDNZD.

Figure 8-17 shows a signal chart on AUDJPY.

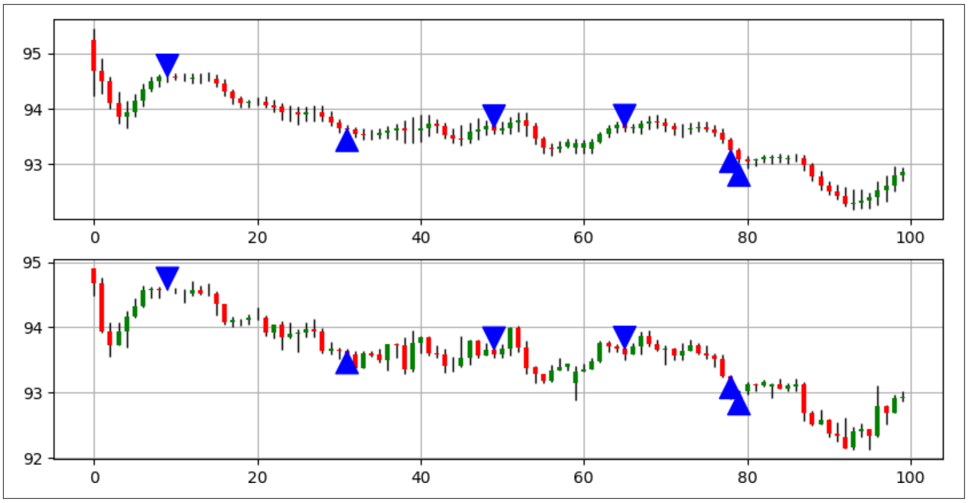


Figure 8-17. On the top, signals (estimated) on K's candlesticks; on the bottom, the same signals on real candlesticks. The currency pair is AUDJPY.

Table 8-7 summarizes the performance metrics of an algorithm that scans and trades the Euphoria pattern on K's candlesticks charts.

Table 8-7. K's candlesticks with Euphoria pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	46.29%	1.02	1.18	3916
USDCHF	44.73%	0.91	1.13	3798
GBPUSD	46.15%	1.00	1.16	4446
USDCAD	45.64%	0.99	1.18	4296
BTCUSD	46.71%	1.01	1.15	3303
ETHUSD	45.14%	0.98	1.19	1327
GOLD	47.40%	1.03	1.14	597
S&P500	48.08%	1.27	1.37	522
FTSE100	48.13%	1.11	1.2	698

Detecting the Double Trouble Pattern

Figure 8-18 shows a signal chart on USDCHF.

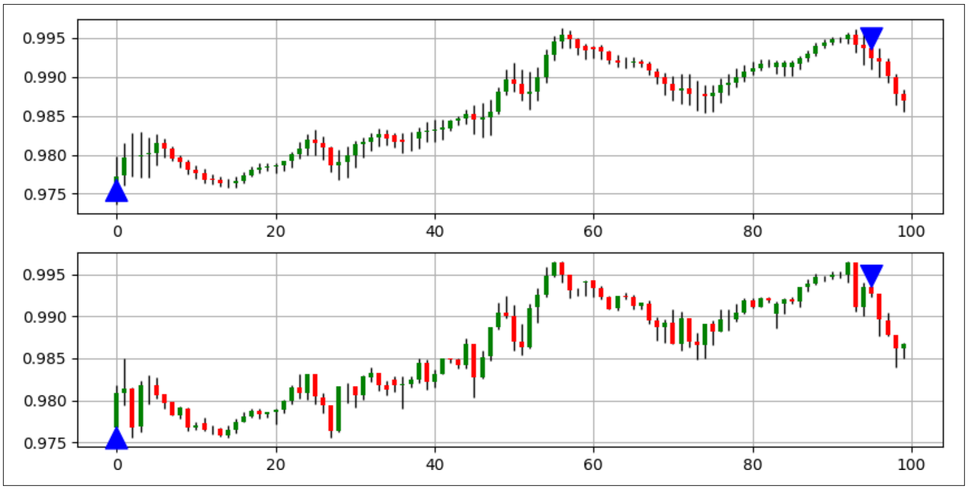


Figure 8-18. On the top, signals (estimated) on K's candlesticks; on the bottom, the same signals on real candlesticks. The currency pair is USDCHF.

Figure 8-19 shows a signal chart on EURGBP.

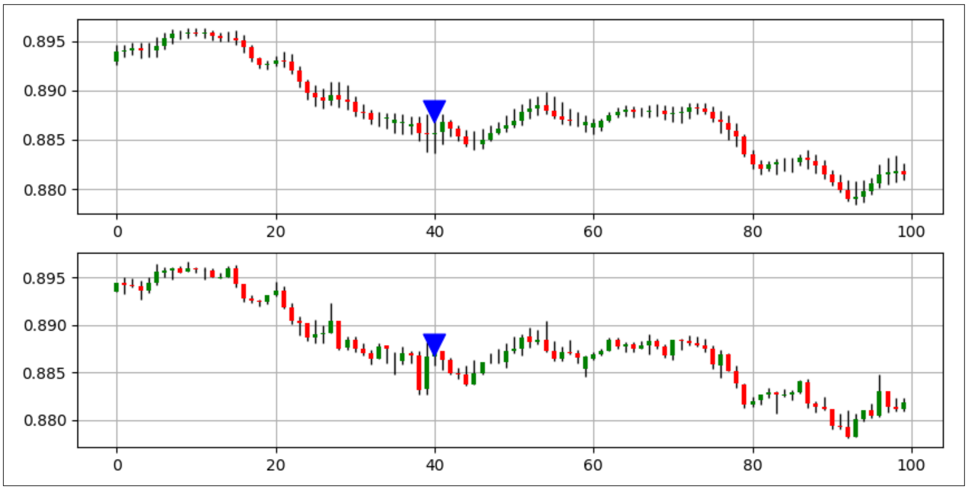


Figure 8-19. On the top, signals (estimated) on K's candlesticks; on the bottom, the same signals on real candlesticks. The asset is EURGBP.

Table 8-8 summarizes the performance metrics of an algorithm that scans and trades the Double Trouble pattern on K's candlesticks charts.

Table 8-8. K's candlesticks with Double Trouble pattern: performance summary table

Asset	Hit Ratio	Profit Factor	Risk-Reward Ratio	Signals
EURUSD	50.35%	0.9	0.89	709
USDCHF	53.36%	1.06	0.93	624
GBPUSD	52.20%	0.98	0.9	793
USDCAD	50.15%	0.93	0.92	664
BTCUSD	54.67%	1.49	1.24	406
ETHUSD	61.37%	2.94	1.85	334
GOLD	51.68%	1.09	1.02	743
S&P500	55.35%	1.07	0.86	112
FTSE100	40.25%	0.42	0.62	77

To summarize, different charting systems are useful in offering different views of the price action. Each system has its advantages and disadvantages. Table 8-9 summarizes some key points about charting systems.

Table 8-9. Comparison of different candlestick charting systems

Charting system	Advantages	Disadvantages
Candlestick chart	Real prices and ease of interpretation	Noisy
Heikin-Ashi chart	Better trend interpretability due to smoothing	Small lag and unreal prices
K's candlestick chart	Better trend interpretability due to extra smoothing	Bigger lag and unreal prices

CHAPTER 9

Candlestick Patterns Exit Techniques

Whenever a candlestick pattern appears, you have to think about three types of techniques (events):

The entry technique
: This technique controls the buy or sell price you use upon validating the pattern. In other words, would you buy at the next open price after the appearance of the pattern, or would you use another price?

The target technique
: This technique controls where to liquidate the position profitably. It can be referred to as the pattern's *potential* or its *expected reaction range*.

The stop technique
: This technique controls where to liquidate the position at a loss. It can be referred to as the pattern's *invalidation point* or the *stop-loss point*.

This chapter discusses the two exit techniques, which are the target and stop techniques.

> Note that the *entry technique* assumes initiating a trade (long or short) on the open of the candlestick that follows the one that validates the pattern.

The Symmetrical Exit Technique

This technique gives an easy target for any candlestick pattern based on the size of a key candlestick inside the pattern. The *symmetrical exit* takes the distance of the high and low of the key candlestick and projects it from one of the extremities (depending on whether it is a bullish or bearish pattern).

The *key candlestick* is the one used to determine the symmetrical target. Generally, it is the candlestick that either confirms the pattern or defines it. For instance, the key candlestick of a Doji pattern is the middle candlestick that resembles a plus sign, while the key candlestick of a Euphoria pattern is the third (and biggest) candlestick.

Figure 9-1 shows an example of a bullish target calculated from the key candlestick (the first one). The example shows a hypothetical pattern. The two arrows are the same size.

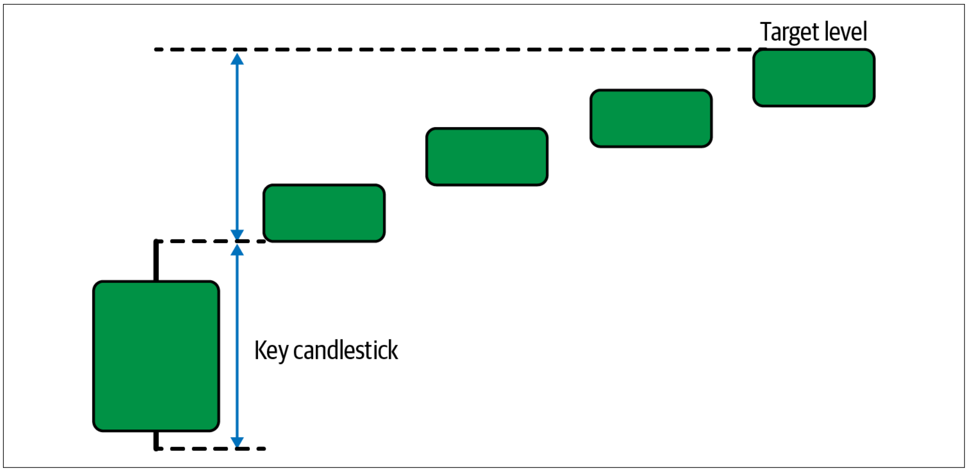


Figure 9-1. The calculation of a bullish target using the symmetrical exit technique

Consider a bullish Engulfing pattern with the following characteristics:

- Open price of the Engulfing candlestick: $100
- High price of the Engulfing candlestick: $105
- Low price of the Engulfing candlestick: $95
- Close price of the Engulfing candlestick: $102

The theoretical bullish target you should be aiming for is ($105 – $95) + $105 = $115.

Figure 9-2 shows the potential of a bullish Engulfing pattern using the technique.

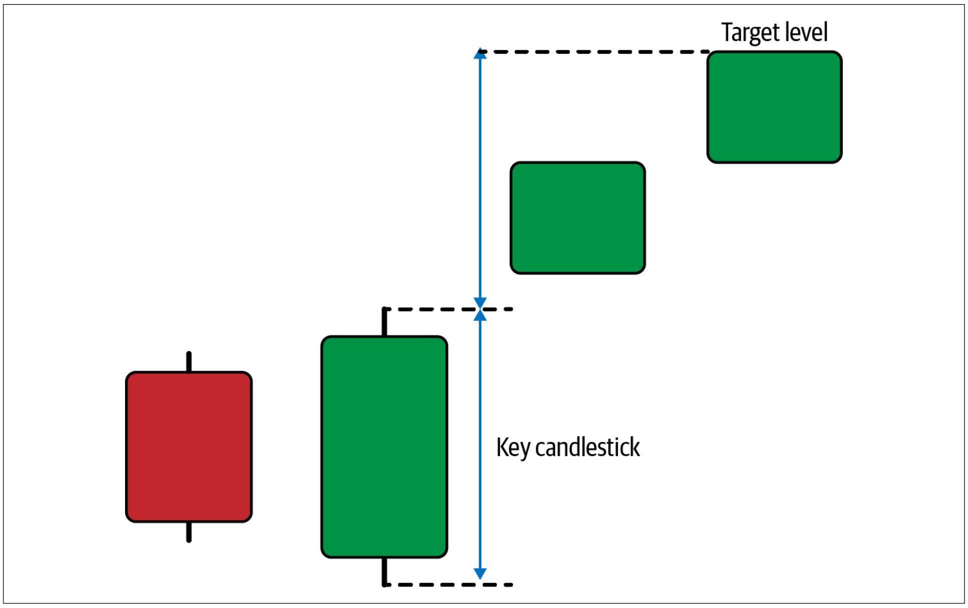


Figure 9-2. The calculation of a bullish target on an Engulfing pattern using the symmetrical exit technique

Figure 9-3 shows an example of a bearish target calculated from the key candlestick. The example shows a hypothetical pattern.

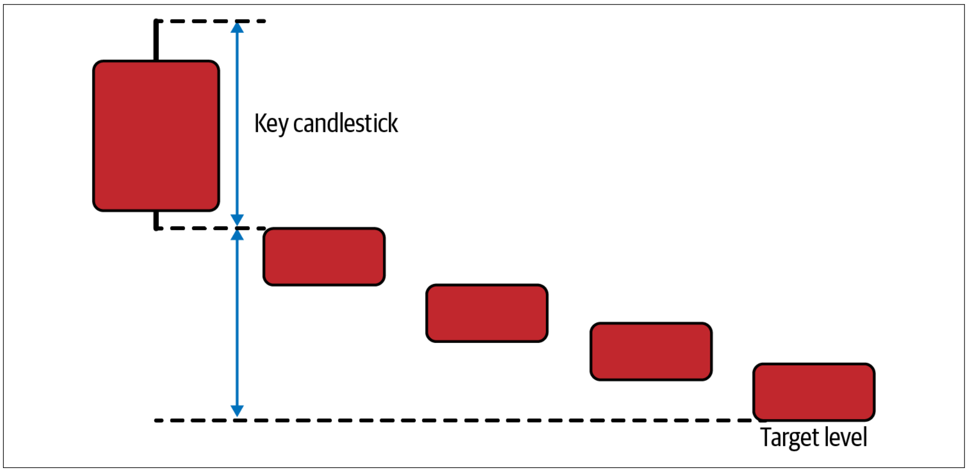


Figure 9-3. The calculation of a bearish target using the symmetrical exit technique

Consider a bearish Piercing (Dark Cloud) pattern with the following characteristics:

- Open price of the Piercing candlestick: $50
- High price of the Piercing candlestick: $55
- Low price of the Piercing candlestick: $45
- Close price of the Piercing candlestick: $52

The theoretical bearish target you should be aiming for is $55 – ($55 – $45) = $45.

Figure 9-4 visualizes the potential of a bullish Engulfing pattern using the technique.

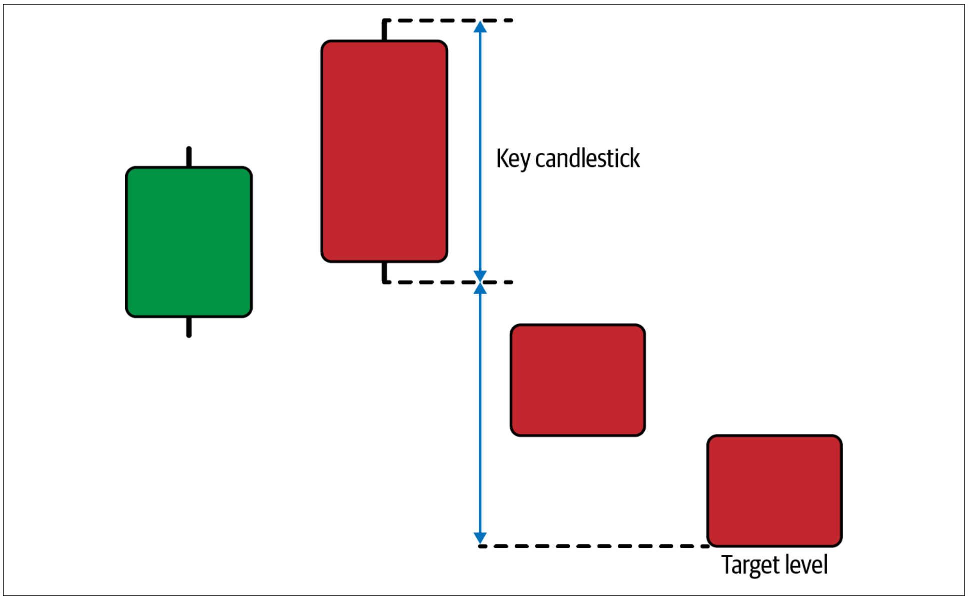


Figure 9-4. The calculation of a bearish target on a Piercing pattern using the symmetrical exit technique

The advantages of the symmetrical exit technique are its ease of use and the simple intuition of simply projecting the distance between the highs and lows, which is in itself a volatility-weighted target measure.

Remember that Engulfing and Piercing patterns are classic contrarian patterns that I discussed in Chapter 6.

The Fixed Holding Period Exit Technique

The fixed holding period assumes that you must exit a pattern after a prespecified number of time periods. It is a relatively inflexible technique and assumes that the reaction must happen within a certain period of time.

The main idea is that when a position is initiated after the validation of a candlestick pattern, a time counter begins and runs until the exit, regardless of the market price relative to the entry price.

For example, a bullish Tweezers pattern is validated, and a position is initiated. Following a fixed holding period of 5, the position must be exited after five time periods. The fixed holding period technique assumes a time-dependent pattern, which is valid for a prespecified window.

The Variable Holding Period Exit Technique

The variable holding period assumes that you must exit a pattern after encountering another pattern. It is a relatively inflexible technique.

The main idea is that when a position is initiated after the validation of a candlestick pattern, you exit it only if you encounter another pattern, whether bullish or bearish. This means that the pattern is valid until the market shows you otherwise.

For example, a bearish Harami pattern is validated, and a position is initiated. Following a variable holding period, the position is exited only if another pattern is encountered.

This technique can be dangerous with rare patterns as the position can remain open for extended periods of time, thus rendering the trades obsolete and meaningless.

This technique assumes that you exit only when you encounter the same type of pattern or any type of other pattern (bullish or bearish). It is worth mentioning that this is the technique used in the back-tests presented throughout the book.

The Hybrid Exit Technique

The hybrid exit technique is probably the best choice among the discussed exit techniques as it combines the attributes of the three exit techniques so as to limit the risks. The hybrid exit technique uses a first-come, first-served method to determine the exit of a candlestick pattern using the following rules:

- The symmetrical projection is calculated and is given a certain weight between 1% and 99%.
- The variable holding period is monitored for every candlestick and is also given a certain weight, which is the remainder of the first given weight.
- The fixed holding period is used to frame the whole trade in a maximum duration scenario.

What the preceding bullet points mean is that you will be exiting the pattern on two occasions, whichever comes first (symmetrical projection or appearance of another pattern). When you exit a position, you have a certain weight, for example 50%. This means that you will exit 50% of the position upon validating the symmetrical projection or the appearance of another pattern.

If you do not meet these two conditions before a prespecified time window elapses, the position is closed (the fixed holding period enters).

Let's try an example. Suppose the market has validated a bullish Mirror pattern on EURUSD with the following trade details:

- Initiate the buy order at $1.0000.
- The calculation of the symmetrical projection gives a target at $1.0100.
- The weighting is set at 50%.
- The fixed holding period is set at five periods.

Here's a hypothetical time line that shows how the trade could go if the trader uses the hybrid approach:

- *H-1*: The market reaches a high of $1.0110; half the trade is closed because the pattern reached its symmetrical projection.
- *H-2*: The market trades and closes at $1.0000.
- *H-3*: The market trades and closes at $0.9995.
- *H-4*: The market trades and closes at $1.0150.
- *H-5*: The market closes at $1.0120, and no pattern has emerged during the five periods. The remainder of the position is liquidated.

Each market may have a unique exit approach that works well on it. Generally, the hybrid technique tries to get the best of all worlds by incorporating multiple exit techniques. Table 9-1 shows the key differences among the discussed exit techniques.

Table 9-1. Summary of exit techniques

Exit technique	Duration	Dependency
Symmetrical exit technique	Medium to long	Price
Fixed holding period	Depends on the user	Time
Variable holding period	Medium to long	Price
Hybrid exit technique	Medium to long	Price and time

Pattern Invalidation

There comes a time when a pattern does not produce the expected reaction, and as you may have seen from the back-testing results, this is not uncommon. In such instances, you must be ready to limit the damage, which is done by creating *invalidation rules*. These rules, depending on the trader, can be objective or subjective.

Patterns can be invalidated by risk management techniques such as stop-losses that are fixed by traders themselves or by volatility measures such as the ATR, which I previously discussed in Chapter 5.

Note that pattern invalidation is similar to an exit technique except it deals with exiting at a loss. However, the fixed holding period technique can be seen as both a target or an invalidation technique due to it being a risk-limiting approach (exiting after a certain amount of time) and a target approach (exiting after believing that the pattern could cause a reaction for a limited period of time).

Pattern invalidation can be twofold:

Fixed stop-losses
: This invalidation method limits risk by imposing a fixed level that liquidates the position if the market reaches it. This method is not recommended as the volatility of the market changes all the time.

ATR-based stop-losses
: This invalidation method limits risk by imposing a level based on the values of the ATR. It is recommended to weight the risk according to volatility.

For example, a trader can always use 20 pips as a stop-loss for their EURUSD short-term trades. This means that if a pattern emerges and a buy order is initiated at $1.0000, whenever the market touches or breaks $0.9980, the position is automatically liquidated at a loss. In contrast, if the same trader uses the ATR-based technique to invalidate the pattern, they would exit if the market breaks the open price minus the ATR reading at initiation multiplied by a constant such as 2.

To be clearer, suppose that the ATR has a value of 20. Then, the stop-loss would be at \$0.9960 ($20 \times 2 = 40$).

CHAPTER 10

Candlestick-Based Trend-Following Strategies

Pattern recognition is only one part of the equation. When you a find a pattern, you are most likely to use it within a wider trading framework, as trading systems relying purely on candlestick patterns are unlikely to yield consistent and stable positive returns.

This chapter discusses some examples of strategies you can use to filter the patterns using specific technical rules when dealing with trend following. Each market has its own characteristics and must have optimized strategy parameters, which is why I recommend you focus on understanding the main ideas and concepts rather than take away the exact parameters of the strategies presented in this chapter.

Ideally, the strategies should help you understand how to combine the different candlestick patterns (classic and modern) with some technical indicators. I also discuss the indicators in this chapter so that you understand their mechanisms and their weaknesses.

Filtering refers to the concept of choosing the signals that have a better probability of providing a positive return.

Combining the Double Trouble Pattern with the RSI

As a reminder, the Double Trouble pattern is a modern trend-following candlestick configuration that incorporates volatility in its characteristics (calculated using the ATR indicator). The RSI is an indicator that, as discussed previously in the book, reinterprets the market's momentum into values between 0 and 100 to better understand the current dynamic.

Even though the RSI is a contrarian indicator, I will show you a technique for using it within a trend-following framework.

This strategy uses the RSI as a filter for the detected Double Trouble patterns. This means that whenever a pattern is found, it is run through an RSI filter to validate the signal.

The RSI can also be a momentum gauge. Therefore, values above 50 represent bullish momentum (uptrend) while values below 50 represent bearish momentum (downtrend). This is one way to use the RSI as a trend-following indicator.

The trading conditions of the strategy are as follows:

- A long signal is generated whenever a bullish Double Trouble pattern appears while the 14-period RSI is above 50.
- A short signal is generated whenever a bearish Double Trouble pattern appears while the 14-period RSI is below 50.

The filter is therefore the value of the RSI relative to the neutrality level of 50. This is used to increase the hit ratio of the trades using the invisible hand mechanism.[1] The basic intuition of the strategy is to take only bullish signals in a bullish regime and only bearish signals in a bearish regime.

The following code snippet shows how to code the signal function of the strategy:

```
def signal(data, open_column, high_column, low_column, close_column,
           atr_column, rsi_column, buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:
```

1 The *invisible hand mechanism* involves using trades in tandem with the trend, for example, only taking buy signals in a bullish trend while disregarding any sell signals.

```
            # Bullish setup
            if data[i, close_column] > data[i, open_column] and \
               data[i, close_column] > data[i - 1, close_column] and \
               data[i - 1, close_column] > data[i - 1, open_column] and \
               data[i, high_column] - data[i, low_column] > \
               (2 * data[i - 1, atr_column]) and \
               data[i, close_column] - data[i, open_column] > \
               data[i - 1, close_column] - data[i - 1, open_column] and \
               data[i, buy_column] == 0 and \
               data[i, rsi_column] > 50:

                     data[i + 1, buy_column] = 1

            # Bearish setup
            elif data[i, close_column] < data[i, open_column] and \
               data[i, close_column] < data[i - 1, close_column] and \
               data[i - 1, close_column] < data[i - 1, open_column] and \
               data[i, high_column] - data[i, low_column] > \
               (2 * data[i - 1, atr_column]) and \
               data[i, open_column] - data[i, close_column] > \
               data[i - 1, open_column] - data[i - 1, close_column] and \
               data[i, sell_column] == 0 and \
               data[i, rsi_column] < 50:

                     data[i + 1, sell_column] = -1

        except IndexError:

             pass

    return data
```

Notice the addition of the filter through the RSI line of code stating its value relative to the neutrality level 50. Figure 10-1 shows a signal chart on USDCAD.

You can always tweak the strategy so that it better fits your profile and the market's characteristics. However, you should keep in mind that the RSI is not a miraculous predictor of trend as it is price-derived and therefore lagging (that is, it does not predict the future but simply tells the story of the past). Also, the RSI tends to break and reintegrate the 50 level many times, which may give faulty signals.

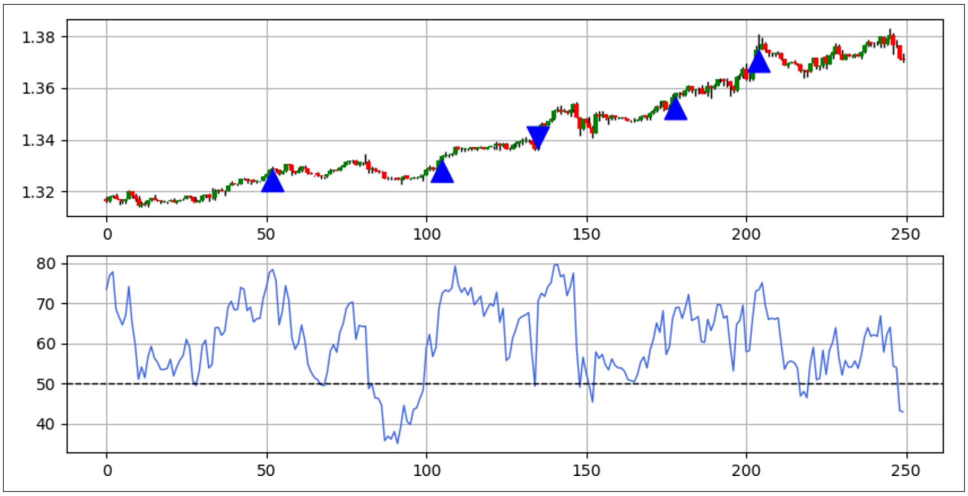


Figure 10-1. Signal chart of the strategy using the Double Trouble pattern and the RSI

Combining the Three Candles Pattern with Moving Averages

Moving averages are great trend filters and help in determining whether to take the signal or not. Remember, a *moving average* is a mean that follows the market price using a rolling window. This strategy uses two trend-following elements from different fields[2] in order to give a signal.

The Three Candles pattern is a trend-following configuration composed of big-bodied homogenous successive candlesticks that confirm the underlying trend. Generally, a bullish Three Candles pattern is called Three White Soldiers, while the bearish version is called Three Black Crows.

Similar to using the RSI as a trend follower, using moving averages is extremely useful in determining the underlying trend.

When you see a market above its moving average (for example, a 100-period moving average), it is generally a sign that a bullish regime is in progress, and when you see a

2 The Three Candles pattern comes from the field of pattern recognition, while moving averages come from the field of statistics and trend-following technical indicators.

market below its moving average, it is generally a sign that a bearish regime is in progress, thus indicating which type of signal to prefer.

The trading conditions of the strategy are as follows:

- A long signal is generated whenever a Three White Soldiers pattern appears while the market price is above its 100-period moving average.
- A short signal is generated whenever a Three Black Crows pattern appears while the market price is below its 100-period moving average.

The trend-following strategies use a combination of a candlestick pattern and a trend filter that increases the conviction in the trade. Although historical back-testing shows that the hit ratio is not always increased with the filter, for some markets, the ratio is significantly enhanced.

The following code snippet shows how to code the signal function of the strategy:

```
def signal(data, open_column, close_column, ma_column, buy_column,
           sell_column):

    data = add_column(data, 10)

    for i in range(len(data)):

       try:

           # Bullish setup
           if data[i, close_column] - data[i, open_column] > body and \
              data[i - 1, close_column] - data[i - 1, open_column] > \
              body and data[i - 2, close_column] - \
              data[i - 2, open_column] > body and data[i, close_column] \
              > data[i - 1, close_column] and data[i - 1, close_column] \
              > data[i - 2, close_column] and data[i - 2, close_column] \
              > data[i - 3, close_column] and data[i, close_column] > \
              data[i, ma_column] and data[i, buy_column] == 0:

                    data[i + 1, buy_column] = 1

           # Bearish setup
           elif data[i, open_column] - data[i, close_column] > body and \
                data[i - 1, open_column] - data[i - 1, close_column] > body \
                and data[i - 2, open_column] - data[i - 2, close_column] \
                > body and data[i, close_column] < \
                data[i - 1, close_column] and data[i - 1, close_column] \
                < data[i - 2, close_column] and data[i - 2, close_column] \
                < data[i - 3, close_column] and data[i, close_column] < \
                data[i, ma_column] and data[i, sell_column] == 0:
                    data[i + 1, sell_column] = -1
            except IndexError:
                pass
    return data
```

Figure 10-2 shows an example of a signal chart with a 100-period moving average (the swooping line starting at the upper left), which acts as a filter. Notice how you see only bullish signals when the market is above its moving average and see only bearish signals when the market is below its moving average.

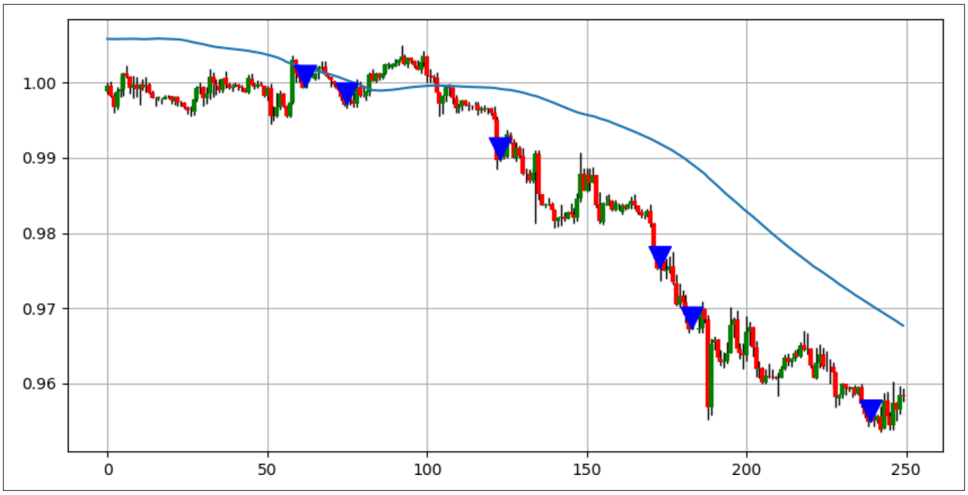


Figure 10-2. Signal chart of the strategy using the Three Candles pattern and moving averages

The strategy is best used in confirmation of other trend-following strategies. For example, a fundamental trader has a carry trade in USDJPY and sees that a Three White Soldiers pattern has emerged while the market is above its 100-period moving average. This observation can serve as a conviction enhancer or a confirmation to add more to the position. After all, trading is a numbers game, and stacking up the odds on your side increases the probability of gain.

A *fundamental trader* is a trader who relies on economic and financial analysis instead of technical analysis to make their decisions.

A *carry trade* is a currency position that entails going long on (buying) the currency with the higher interest rate and shorting (selling) the currency with the lower interest rate to benefit from the interest rate differential.

Combining the Bottle Pattern with the Stochastic Oscillator

Similar to the RSI, the stochastic oscillator is a momentum indicator heavily used in technical trading and known both to the retail and professional communities. The *stochastic oscillator* uses a basic normalization function to trap the values between 0 and 100. It is easier to calculate than the RSI. Before discussing the oscillator, let's look at the concept of *normalization*.

Whenever you have an array of different unbounded values such as the market price (or any other random set of time series), you can normalize the values between 0 and 1, with 0 being the lowest value for a specific time window and 1 being the highest value for a specific time window. Take a look at this table:

Time step	1	2	3	4	5
Value	10	40	5	90	25
Normalized value	0.06	0.41	**0.00**	**1.00**	0.24

The table shows that when the variable moves in time from period 1 to period 5, the normalized values show that they can be bounded between 0 and 1, with 0 referring to the lowest value (5) and 1 referring to the highest value (90). Similarly, notice how the value that lies approximately between 0 and 1 has a normalized value of 0.41, which means it is probably the middle value. This is normal as 40 is around half the distance between 5 and 90.

The following formula shows how to normalize any value between 0 and 1 given a specific time window:

$$x_{\text{Normalized}} = \frac{x_{\text{Original}} - x_{\text{Low}}}{x_{\text{High}} - x_{\text{Low}}}$$

Let's try an example using the previous table. Take the value 10 and try to normalize it using the formula. You should have the following calculation:

$$x_{\text{Normalized}} = \frac{10 - 5}{90 - 5} = 0.06$$

The stochastic oscillator normalizes the market price in a modified way by incorporating the highs and lows in the formula so that it becomes as follows:

$$\text{Stochastic value}_i = \frac{\text{Close}_i - \text{Low}_{i-n:i}}{\text{High}_{i-n:i} - \text{Low}_{i-n:i}}$$

The formula means that the current value of the stochastic is the difference between the current close price minus the lowest low price for a set lookback period divided by the difference between the highest high price and the lowest low price for the same lookback period.

The difference between the simple normalization function and the stochastic oscillator's function is the addition of the high price and low price to the latter.

The stochastic oscillator can be described as the smoothed version of the previous formula, and it is generally charted with a short-term moving average calculated on its value, called the *signal line*. To create the default stochastic oscillator, follow these steps:

1. Normalize the values using the stochastic oscillator's function using a rolling window of 14 periods.
2. Smooth out the result from the first step with a three-period moving average. This is the stochastic oscillator.
3. Calculate the signal line, which is another three-period moving average calculated on the values of the second step. This is the signal line.

The first moving average calculated on the stochastic oscillator is referred to as *smoothing*, while the signal line is referred to as *slowing*.

Similar to the RSI, the stochastic oscillator is bounded between 0 and 100 and has an oversold zone below 20 and an overbought zone above 80. Because of its formula, it is relatively more volatile than the RSI and tends to move from one extreme to another faster.

Many techniques are possible with the stochastic oscillator, but the one that interests us is the cross between it and its signal line. This is referred to as the *cross technique* and is famous in contrarian strategies. (However, I am using it with a trend-following candlestick pattern and therefore turning it into a trend-following technique.)

You must be careful with the stochastic oscillator as it tends to stick to the extremes from time to time due to the nature of the normalization function, thus providing false signals. The stickiness effect is manifested when the oscillator remains in the oversold and overbought zones for extended periods of time.

The following snippet shows how to code the stochastic oscillator:

```
def stochastic_oscillator(data,
                          lookback,
                          high,
                          low,
                          close,
                          position,
                          slowing = False,
                          smoothing = False,
                          slowing_period = 1,
                          smoothing_period = 1):

    data = add_column(data, 1)

    for i in range(len(data)):

        try:

            data[i, position] = (data[i, close] - min(data[i - lookback \
                                + 1:i + 1, low])) / (max(data[i - lookback\
                                + 1:i + 1, high]) - min(data[i - lookback \
                                + 1:i + 1, low]))

        except ValueError:

            pass

    data[:, position] = data[:, position] * 100

    if slowing == True and smoothing == False:

        data = ma(data, slowing_period, position, position + 1)

    if smoothing == True and slowing == False:

        data = ma(data, smoothing_period, position, position + 1)

    if smoothing == True and slowing == True:

        data = ma(data, slowing_period, position,   position + 1)
```

```
            data = ma(data, smoothing_period, position + 1, position + 2)

        data = delete_row(data, lookback)

        return data
```

The trading conditions of the strategy are as follows:

- A long signal is generated whenever a bullish Bottle pattern appears while the stochastic oscillator is above its signal line.
- A short signal is generated whenever a bearish Bottle pattern appears while the stochastic oscillator is below its signal line.

The following code snippet shows how to code the signal function of the strategy:

```
    def signal(data, open_column, high_column, low_column, close_column,
               stochastic_column, signal_column, buy_column, sell_column):

        data = add_column(data, 5)

        for i in range(len(data)):

           try:

               # Bullish setup
               if data[i, close_column] > data[i, open_column] and \
                  data[i, open_column] == data[i, low_column] and \
                  data[i - 1, close_column] > data[i - 1, open_column] and \
                  data[i, open_column] < data[i - 1, close_column] and \
                  data[i, stochastic_column] > data[i, signal_column] and \
                  data[i, buy_column] == 0:

                     data[i + 1, buy_column] = 1

               # Bearish setup
               elif data[i, close_column] < data[i, open_column] and \
                    data[i, open_column] == data[i, high_column] and \
                    data[i - 1, close_column] < data[i - 1, open_column] and \
                    data[i, open_column] > data[i - 1, close_column] and \
                    data[i, stochastic_column] > data[i, signal_column] and \
                    data[i, sell_column] == 0:

                     data[i + 1, sell_column] = -1

           except IndexError:

                pass

        return data
```

Figure 10-3 shows the signal chart on USDCHF.

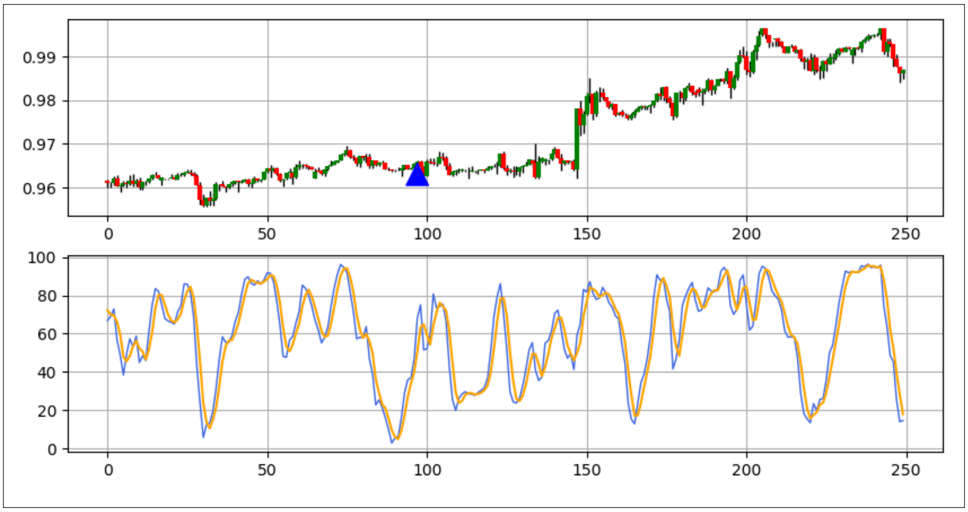


Figure 10-3. Signal chart of the strategy using the Bottle pattern and the stochastic oscillator

Combining the Marubozu Pattern with K's Volatility Bands

Let's get to square one with the first candlestick pattern covered in this book. The Marubozu pattern can be considered the most powerful candlestick since it has no wicks, meaning that the market went straight from one point to another without hesitation.

This strategy uses a concept known as a *volatility band*, which is a framing technique that envelops the market price to deliver dynamic support and resistance levels.

There are many types of volatility bands. The most well-known are Bollinger bands. The reliability of different types of volatility bands depends on the underlying markets and their parameters.

Before you can grasp K's volatility bands, you have to understand the basics of Bollinger bands. Developed by John Bollinger, the bands are more statistical in nature than technical.

Consider the following list: {11, 4, 5, 20}. Given the four values, how would you describe the elements? Generally, the best measure that describes elements in a list is the mean. It is also the best estimate of the next expected value (if you were to add a new element chronologically). To calculate the mean of a list, follow this formula:

$$\chi = \frac{1}{n}\left(\Sigma_{i=1}^{n} x_i\right) = \frac{x_1 + x_2 + \ldots + x_n}{n}$$

Therefore, according to the formula, you must sum them and divide the result by their quantity:

$$\chi = \frac{11 + 4 + 5 + 20}{4} = 10$$

Hence, the mean of the list is 10. Remember the concept of volatility from previous chapters where you use the ATR to approximate the fluctuations of the price relative to a past period. The bands use another technique to calculate volatility, which is the one used in descriptive statistics, that is, standard deviation.

Standard deviation is the square root of the squared deviations of each variable from the group's mean. The concept may sound complicated, but let's simplify it in a few steps:

- Calculate the distance of each variable (the close price) from the mean of the lookback period at the same time step.
- Square the distances so that you do not get negative values.
- Calculate the mean of these squared distances. The result is called the *variance*.
- Calculate the square root of the variance. The result is called the *standard deviation*.

Taking the square root in the last step allows you to compare apples to apples with the mean.

Mathematically speaking, the formula of the standard deviation is expressed as follows:

$$\sigma = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(x_i - \chi\right)^2}$$

Now that you know how to calculate a moving average, you can simply apply a rolling standard deviation measure, which is the same as a moving average except you are calculating a rolling volatility. Bollinger bands are found as follows:

- The upper Bollinger band is the sum of the current 20-period moving average and the current standard deviation multiplied by 2.
- The lower Bollinger band is the difference between the current 20-period moving average and the current standard deviation multiplied by 2.

The constant (2 by default) is multiplied by the standard deviation's value, and then the product is summed or subtracted from the current value of the moving average.

Figure 10-4 shows an example with Bollinger bands applied on USDCHF. Some traders like to keep the middle line (which is simply the 20-period moving average). However, with Bollinger bands, the line does not provide as much value as the bands themselves.

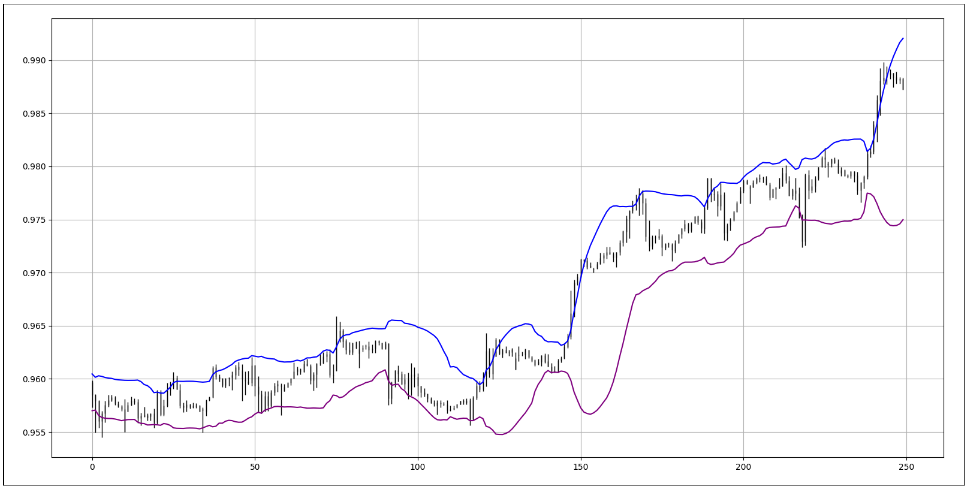


Figure 10-4. An example of Bollinger bands applied on USDCHF

Generally, when the market reaches the lower Bollinger band, it is believed to be oversold, and a bullish reacted is expected. Similarly, when the market reaches the upper Bollinger band, it is believed to be overbought, and a bearish reacted is expected.

Let's now look at K's volatility bands, which are inspired by Bollinger bands. They follow these conditions:

- Calculate the mean between the highest highs and the lowest lows of the last 20 periods.
- Calculate the highest standard deviation measure of the last 20 periods.
- The upper volatility band is the sum of the result from the first step and the second step multiplied by 2.
- The lower volatility band is the difference between the result from the first step and the second step multiplied by 2.

Similarly, the constant (2 by default) is multiplied by the standard deviation's maximum value, and then the product is summed or subtracted from the current value of the mean of the maximum highs and lows.

The following code snippet shows the function to code K's volatility bands:

```python
# Defining the standard deviation function
def volatility(data, lookback, close, position):

    data = add_column(data, 1)

    for i in range(len(data)):

        try:

            data[i, position] = (data[i -lookback + 1:i + 1, close].std())

        except IndexError:

            pass

    data = delete_row(data, lookback)

    return data

def k_volatility_band(data, lookback, multiplier, high, low, close,
                      position):

    data = add_column(data, 6)

    # Calculating the median line
    for i in range(len(data)):

        try:
            data[i, position] = max(data[i - lookback + 1:i + 1, high])
            data[i, position + 1] = min(data[i - lookback + 1:i + 1, low])
            data[i, position + 2] = (data[i, position] + data[i, position \
                                    + 1]) \/ 2

        except ValueError:

            pass

    data = delete_column(data, position, 2)

    # Calculating maximum volatility
    data = volatility(data, lookback, close, position + 1)

    for i in range(len(data)):

        try:

            data[i, position + 2] = max(data[i - lookback + 1:i + 1, \
                                    position + 1])

        except ValueError:

            pass

    data = delete_column(data, position + 1, 1)

    # Calculating the bands
    data[:, position + 2] = data[:, position] + (multiplier * data[:, \
                            position + 1])
    data[:, position + 3] = data[:, position] - (multiplier * data[:, \
                            position + 1])

    data = delete_column(data, position + 1, 1)

    return data
```

Keeping the middle line is essential to the strategy as it is the main filter. Therefore, with K's volatility bands, the middle line is present in the chart.

Note that the words *volatility* and *standard deviation* are used interchangeably. Figure 10-5 shows an example with K's volatility bands applied on USDCHF.

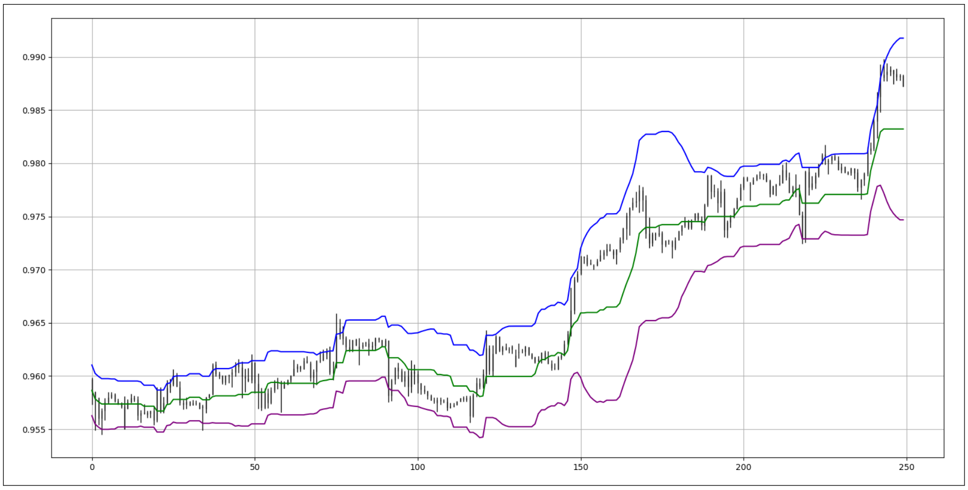


Figure 10-5. An example of K's volatility bands applied on USDCHF

Let's now discuss the strategy. The trading conditions are as follows:

- A long signal is generated whenever a bullish Marubozu pattern appears while the market price is below the middle line.
- A short signal is generated whenever a bearish Marubozu pattern appears while the market price is above the middle line.

The following code snippet shows how to code the signal function of the strategy:

```
def signal(data, open_column, high_column, low_column, close_column,
           middle_band, buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish setup
           if data[i, close_column] > data[i, open_column] and \
              data[i, high_column] == data[i, close_column] and \
              data[i, low_column] == data[i, open_column] and \
              data[i, close_column] < data[i, middle_band] and \
              data[i, buy_column] == 0:

                    data[i + 1, buy_column] = 1

           # Bearish setup
           elif data[i, close_column] < data[i, open_column] and \
                data[i, high_column] == data[i, open_column] and \
                   data[i, low_column] == data[i, close_column] and \
                   data[i, close_column] > data[i, middle_band] and \
                   data[i, sell_column] == 0:

                    data[i + 1, sell_column] = -1

            except IndexError:

                pass

        return data
```

Figure 10-6 shows the signal chart on AUDNZD.

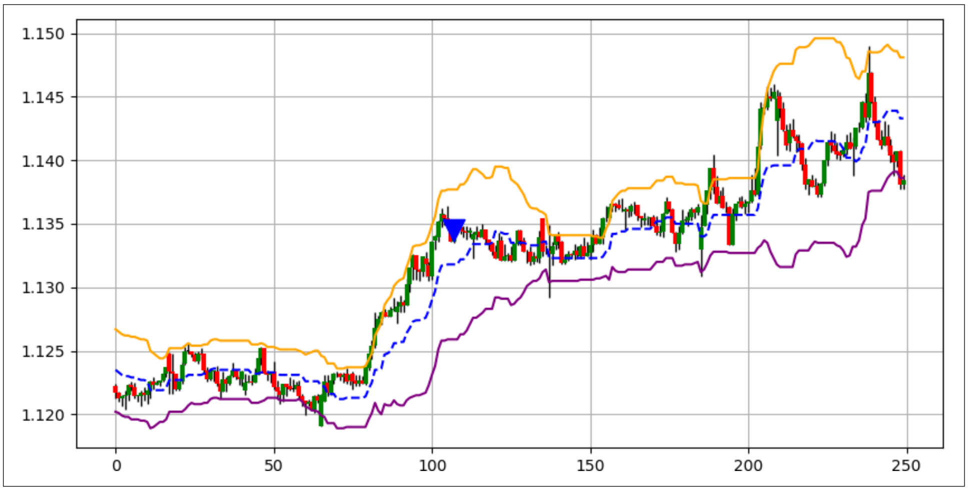


Figure 10-6. Signal chart of the strategy using the Marubozu pattern and K's volatility bands

Optionally, you can consider the following more restraining conditions (however, the signals will be less frequent):

- A long signal is generated whenever a bullish Marubozu pattern appears while the market price is below the lower volatility band.
- A short signal is generated whenever a bearish Marubozu pattern appears while the market price is above the upper volatility band.

To sum up, the strategy can have fewer signals, but intuitively, it has a strong foundation pertaining to the most powerful candlestick (in terms of amplitude) and the statistical extreme.

Combining the H Pattern with the Trend Intensity Index

The *trend intensity index* (TII) measures the strength of the trend. It is a simple calculation created out of a moving average and price deviations around it to show, in a bounded appearance, the intensity of the trend. To build the indicator, follow these steps:

1. Calculate a 20-period moving average on the market price.
2. Calculate the deviations of the market price from the moving average. This is done by doing so on two columns. If the close price is greater than its moving average, then the first column is filled by the difference between the two (up variable), and if the current market price is lower than its moving average, then the second column is filled by the difference between the two (down variable).
3. Count the values where the market was above its moving average and where it was below it.

Counting the values with certain conditions can be accomplished with the `numpy` function `count_nonzero()`.

- Apply the following formula to find the TII (by default, a 20-period lookback):

$$TII_i = \left(\frac{\text{Number of up}}{\text{Number of up} + \text{Number of down}}\right)x100$$

To code the TII in Python, use the following function:

```
def trend_intensity_indicator(data, lookback, close_column, position):

    data = add_column(data, 5)

    # Calculating the moving average
    data = ma(data, lookback, close_column, position)

    # Deviations
    for i in range(len(data)):

        if data[i, close_column] > data[i, position]:
           data[i, position + 1] = data[i, close_column] - \
           data[i, position]

        if data[i, close_column] < data[i, position]:
           data[i, position + 2] = data[i, position] - \
           data[i, close_column]

    # Trend intensity index
    for i in range(len(data)):

        data[i, position + 3] = np.count_nonzero(data[i - lookback + 1:i \
                                + 1, position + 1])

    for i in range(len(data)):

        data[i, position + 4] = np.count_nonzero(data[i - lookback + 1:i \
                                + 1, position + 2])

    for i in range(len(data)):

        data[i, position + 5] = ((data[i, position + 3]) / (data[i, \
                                position + 3] + data[i, position + 4])) \
                                * 100

    data = delete_column(data, position, 5)

    return data
```

Typically, a strong bullish momentum is in progress when the TII is above 50, and a strong bearish momentum is in progress when the TII is below 50.

The trading conditions of the strategy are as follows:

- A long signal is generated whenever a bullish H pattern appears while the 20-period TII is above 50.
- A short signal is generated whenever a bearish H pattern appears while the 20-period TII is below 50.

It is preferable to tweak the lookback of the TII to find the right adjustment. The H pattern, being a reversal-invalidation pattern (thus, a trend confirmation pattern), could work well with an indicator that shows the strength of the underlying trend.

The following code snippet shows how to code the signal function of the strategy:

```
def signal(data, open_column, high_column, low_column, close_column,
           tii_column, buy_column, sell_column):

    data = add_column(data, 5)

    data = rounding(data, 4) # Put 0 instead of 4 as of pair 4

    for i in range(len(data)):

       try:

           # Bullish setup
           if data[i, close_column] > data[i, open_column] and \
              data[i, close_column] > data[i - 1, close_column] and \
              data[i, low_column] > data[i - 1, low_column] and \
              data[i - 1, close_column] == data[i - 1, open_column] and \
              data[i - 2, close_column] > data[i - 2, open_column] and \
              data[i - 2, high_column] < data[i - 1, high_column] and \
              data[i, tii_column] > 50:

                    data[i + 1, buy_column] = 1

           # Bearish setup
           elif data[i, close_column] < data[i, open_column] and \
                data[i, close_column] < data[i - 1, close_column] and \
                data[i, low_column] < data[i - 1, low_column] and \
                data[i - 1, close_column] == data[i - 1, open_column] and \
                data[i - 2, close_column] < data[i - 2, open_column] and \
                data[i - 2, low_column] > data[i - 1, low_column] and \
                data[i, tii_column] < 50:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 10-7 shows the signal chart on EURGBP.

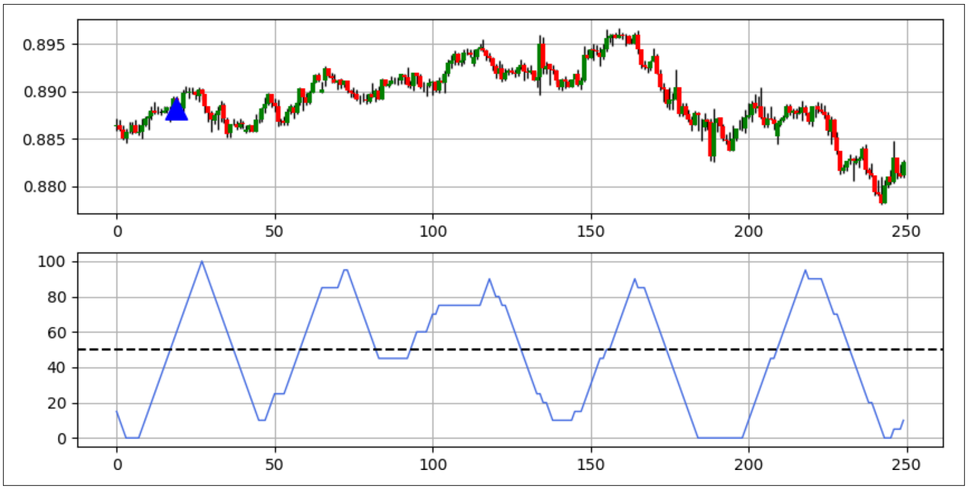


Figure 10-7. Signal chart of the strategy using the H pattern and the trend intensity index

The graph shows that there was one signal that coincided with the bullish conditions.

To sum up this chapter, it is important to combine the right patterns with the right indicators, and this is done through back-testing. Naturally, the combinations presented are merely examples, and more powerful combinations can be created by fusing other indicators with other patterns. It is also interesting to know that you can include multiple patterns in your strategies rather than combine just one pattern with one indicator.

CHAPTER 11

Candlestick-Based Contrarian Strategies

This chapter covers the contrarian part of candlestick pattern strategies. Remember that candlestick patterns on their own are unlikely to provide stable returns, as they must join forces with more sophisticated techniques and indicators in order to transform simple ideas and observations into actionable trade setups.

You should recognize some of the indicators used in this chapter as they have already shown that they can be useful in trend following. Notably, the RSI can provide not only the current market state (helpful in determining the trend) but also extreme levels that could point to a reversal (helpful in determining contrarian trades). In other words, depending on how you use it, the RSI can be a trend-following or a contrarian indicator.

Make sure to focus on the intuition of combining patterns with indicators so that you can create your own combination of strategies. After all, the point of Chapter 10 and Chapter 11 is to help you design your own strategy.

Combining the Doji Pattern with the RSI

This strategy may be the most familiar one that combines a candlestick pattern and a technical indicator. With the Doji being the simplest contrarian configuration and the RSI being the most commonly used and researched indicator, both can give a good confirmation for the expected contrarian move.

The RSI filter makes it possible to transform the Doji into a one-candlestick pattern instead of three. This is because the first and third candlesticks in the Doji pattern are used only to differentiate bullish from bearish Doji patterns. Therefore, the Doji is just the plus sign–looking candlestick.

The strategy uses an aggressive technique on the RSI that awaits for its reading to be below the oversold level or above the overbought level. Oversold and overbought levels are variable and depend on the underlying market and on the lookback of the RSI.

Generally, with a 14-period RSI (the default version), traders tend to use 30 as the oversold level and 70 as the overbought level. This strategy uses a 3-period RSI with 20 as the oversold level and 80 as the overbought level. This aims to increase the frequency of the signals as it makes it more likely to have a simultaneous Doji pattern with a signal on the RSI.

The trading conditions are as follows:

- A long signal is generated whenever a bullish Doji pattern appears while the 3-period RSI is below 20.
- A short signal is generated whenever a bearish Doji pattern appears while the 3-period RSI is above 80.

The following code snippet shows how to code the signal function of the strategy:

```python
lower_barrier = 20
upper_barrier = 80

def signal(data, open_column, close_column, indicator_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    data = rounding(data, 0)

    for i in range(len(data)):

       try:

           # Bullish setup
           if data[i, close_column] == data[i, open_column] and \
              data[i, indicator_column] < lower_barrier:

                    data[i + 1, buy_column] = 1

           # Bearish setup
           elif data[i, close_column] == data[i, open_column] and \
                data[i, indicator_column] > upper_barrier:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 11-1 shows a signal chart on AUDNZD showing the strategy in action. As you may have gathered from previous chapters, contrarian techniques work better in a flat (sideways) market, as there is an implied equilibrium between supply and demand and an excess in either is bound to return to normality. This excess is measured through contrarian tools.

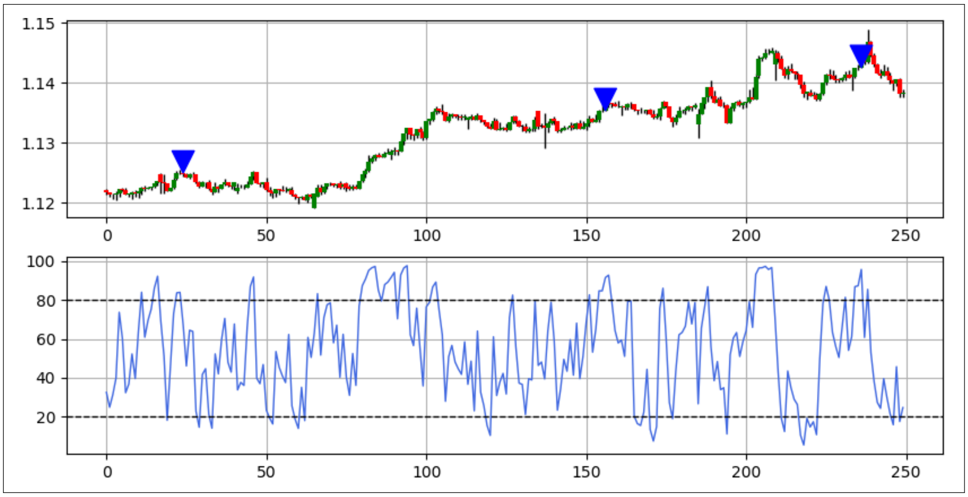


Figure 11-1. Signal chart of the strategy using the Doji pattern and the RSI

To sum up, the Doji pattern is versatile and is easily integrated with technical indicators due to its simplicity. The strategy is easy to grasp, and many other variations exist that tweak it to improve its frequency and profitability.

Combining the Engulfing Pattern with Bollinger Bands

Chapter 10 mentioned that Bollinger bands are an envelopment technique that move with the market price and provide dynamic support and resistance levels using a statistical approach. This strategy couples the Engulfing pattern with the position of the market price relative to the lower or upper Bollinger band.

The trading conditions of the strategy are as follows:

- A long signal is generated whenever a bullish Engulfing pattern appears while the market price is below the lower Bollinger band.
- A short signal is generated whenever a bearish Engulfing pattern appears while the market price is above the upper Bollinger band.

A market price below its lower Bollinger band implies a statistically oversold event and suggests a bullish reaction, whereas a market price above its upper Bollinger band implies a statistically overbought event and suggests a bearish reaction.

The following code snippet shows how to code the signal function of the strategy:

```python
def signal(data, open_column, close_column, upper_band_column,
           lower_band_column, buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish setup
           if data[i, close_column] > data[i, open_column] and \
              data[i, open_column] < data[i - 1, close_column] and \
              data[i, close_column] > data[i - 1, open_column] and \
              data[i - 1, close_column] < data[i - 1, open_column] and \
              data[i - 2, close_column] < data[i - 2, open_column] and \
              data[i, close_column] < data[i, lower_band_column]:

                    data[i + 1, buy_column] = 1

           # Bearish setup
           elif data[i, close_column] < data[i, open_column] and \
                data[i, open_column] > data[i - 1, close_column] and \
                data[i, close_column] < data[i - 1, open_column] and \
                data[i - 1, close_column] > data[i - 1, open_column] and \
                data[i - 2, close_column] > data[i - 2, open_column] and \
                data[i, close_column] > data[i, upper_band_column]:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Generally, Bollinger bands use a lookback period of 20 and a standard deviation of 2. But let's review what this means one more time so you thoroughly understand how this great indicator is designed:

- The 20-period lookback period refers to a 20-period moving average calculated on the market price (close price). Similarly, a 20-period moving standard deviation measure is calculated on the market price (close price).

- The standard deviation of 2 means that the rolling standard deviation measure is multiplied by two before adding it to the moving average or subtracting it.

The following code snippet shows the function to calculate the Bollinger bands:

```
def bollinger_bands(data, lookback, standard_deviation, close, position):

    data = add_column(data, 2)

    # Calculating the moving average
    data = ma(data, lookback, close, position)

    # Calculating the standard deviation
    data = volatility(data, lookback, close, position + 1)

    data[:, position + 2] = data[:, position] + (standard_deviation * 
                            data[:, position + 1])
    data[:, position + 3] = data[:, position] - (standard_deviation * 
                            data[:, position + 1])

    data = delete_row(data, lookback)

    data = delete_column(data, position + 1, 1)

    return data
```

The words *volatility* and *standard deviation* are used interchangeably.

The strategy uses 20-period Bollinger bands with a standard deviation of 1. This is to increase the frequency of signals just as you have seen with the previous strategy. Figure 11-2 shows a signal chart in USDCAD so you can see the strategy in action.

Notice how the bands are closer to the market price and are regularly broken or surpassed. Statistically speaking, if we assume that financial time series follow a normal distribution, then 68% of the market price should be contained within one standard deviation of the moving average (represented by the upper and lower bands).

The Engulfing pattern is not an abundant pattern on its own, and thus by adding an extra filter (Bollinger bands), the frequency goes down even further, which is why tweaking the parameters is important.

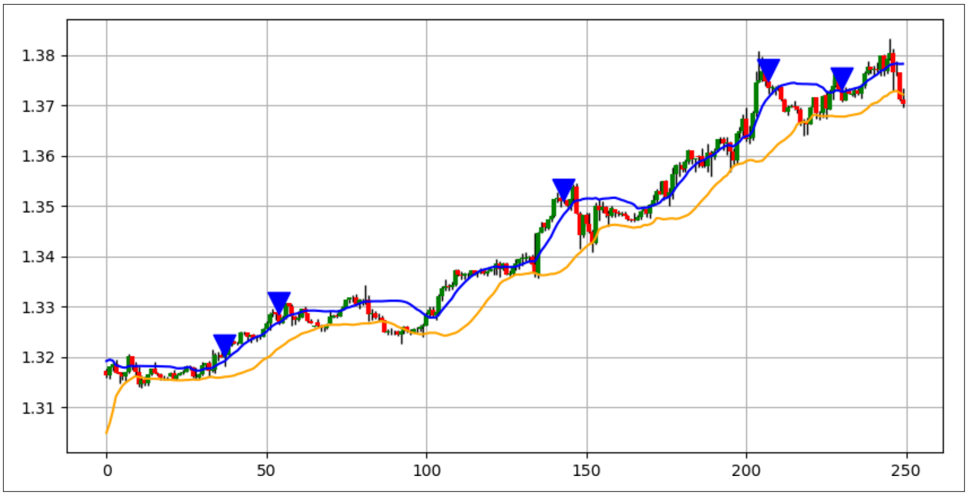


Figure 11-2. Signal chart of the strategy using the Engulfing pattern and Bollinger bands

Combining the Piercing Pattern with the Stochastic Oscillator

This strategy combines the Piercing pattern, a classic candlestick contrarian configuration, with the stochastic oscillator, a technical indicator discussed in Chapter 10.

Remember that the stochastic oscillator uses the normalization function to create a rolling calculation trapped between 0 and 100 while taking into account the high and low prices. It resembles the RSI in the way it is interpreted and used.

The trading conditions of the strategy are as follows:

- A long signal is generated whenever a bullish Piercing pattern appears while the 14-period stochastic oscillator is below 20.
- A short signal is generated whenever a bearish Piercing pattern appears while the 14-period stochastic oscillator is above 80.

The following code snippet shows how to code the signal function of the strategy:

```
lower_barrier = 20
upper_barrier = 80

def signal(data, open_column, close_column, indicator_column,
           buy_column, sell_column):

    data = add_column(data, 5)
    for i in range(len(data)):

       try:

           # Bullish setup
           if data[i, close_column] > data[i, open_column] and \
              data[i, close_column] < data[i - 1, open_column] and \
              data[i, close_column] > data[i - 1, close_column] and \
              data[i, open_column] < data[i - 1, close_column] and \
              data[i - 1, close_column] < data[i - 1, open_column] and \
              data[i - 2, close_column] < data[i - 2, open_column] and \
              data[i, indicator_column] < lower_barrier:

                    data[i + 1, buy_column] = 1

           # Bearish setup
           elif data[i, close_column] < data[i, open_column] and \
                data[i, close_column] > data[i - 1, open_column] and \
                data[i, close_column] < data[i - 1, close_column] and \
                data[i, open_column] > data[i - 1, close_column] and \
                data[i - 1, close_column] > data[i - 1, open_column] and \
                data[i - 2, close_column] > data[i - 2, open_column] and \
                data[i, indicator_column] > upper_barrier:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 11-3 shows a signal chart on USDCHF with bullish and bearish signals represented by upward-pointing arrows and downward-pointing arrows, respectively.

To sum up, combining the Piercing pattern with the stochastic oscillator offers an extra confirmation factor to the expected reversal. The stochastic oscillator can be tweaked as much as needed to calibrate the strategy to the desired frequency and profitability. Of course, no strategy works on all assets, but that's the nature of these semi-random environments we call financial markets.

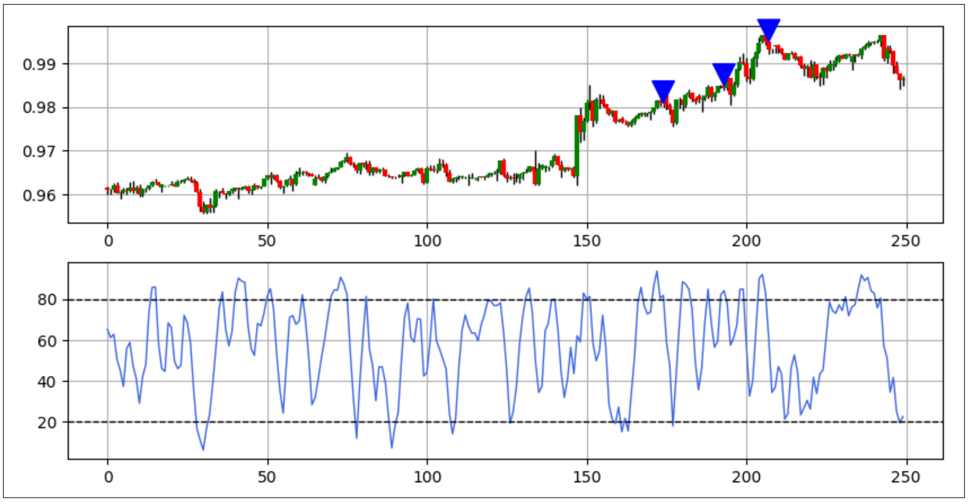


Figure 11-3. Signal chart of the strategy using the Piercing pattern and the stochastic oscillator

Combining the Euphoria Pattern with K's Envelopes

K's envelopes are a couple of simple 800-period moving averages where one is applied on the highs of the price and the other is applied on the lows of the price, thus creating a zone. Therefore, it is a dynamic support and resistance area that moves with the market price. An important characteristic of K's envelopes is stability, which comes from the fact that the lookback period is significantly high, therefore making it relatively immune to short-term fluctuations or noise.

The indicator is extremely simple and is used in determining the current regime (bullish or bearish) or to find demand and supply zones whenever the market enters the zone. The following block of code shows the function of K's envelopes:

```
def k_envelopes(data, lookback, high, low, position):

    # Calculating the upper moving average
    data = ma(data, lookback, high, position)

    # Calculating the lower moving average
    data = ma(data, lookback, low, position + 1)

    return data
```

Figure 11-4 shows USDCHF with K's envelopes applied. Notice how the envelopes provide a dynamic support and resistance zone.

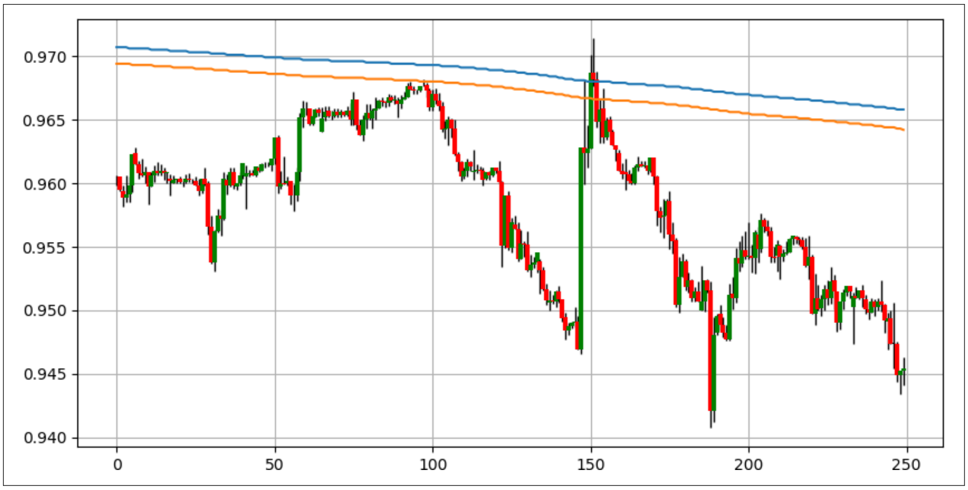


Figure 11-4. USDCHF with K's envelopes

Now, the idea is to combine K's envelopes with a candlestick pattern. The trading conditions of the strategy are as follows:

- A long signal is generated whenever a bullish Euphoria pattern appears while the current market price is inside K's envelopes.
- A short signal is generated whenever a bearish Euphoria pattern appears while the current market price is inside K's envelopes.

By being inside K's envelopes, the market price is therefore pending an expecting reaction in the direction that is determined by the Euphoria pattern. In other words, the trigger and the direction are given by the Euphoria pattern, while the confirmation is given by K's envelopes.

The following code snippet shows how to code the signal function of the strategy:

```
def signal(data, open_column, close_column, upper_k_envelope,
           lower_k_envelope, buy_column, sell_column):

    data = add_column(data, 5)

    data = rounding(data, 4) # Put 0 instead of 4 as of pair 4

    for i in range(len(data)):

       try:

           # Bullish setup
           if data[i, open_column] > data[i, close_column] and \
              data[i - 1, open_column] > data[i - 1, close_column] and \
              data[i - 2, open_column] > data[i - 2, close_column] and \
              data[i, close_column] < data[i - 1, close_column] and \
              data[i - 1, close_column] < data[i - 2, close_column] and \
              (data[i, open_column] - data[i, close_column]) > (data[i - 1,
              open_column] - data[i - 1, close_column]) and \
              (data[i - 1, open_column] - data[i - 1, close_column]) >
              (data[i - 2, open_column] - data[i - 2, close_column]) and \
              data[i, close_column] > data[i, lower_k_envelope] and \
              data[i, close_column] < data[i, upper_k_envelope]:

                    data[i + 1, buy_column] = 1

           # Bearish setup
           elif data[i, open_column] < data[i, close_column] and \
                data[i - 1, open_column] < data[i - 1, close_column] and \
                data[i - 2, open_column] < data[i - 2, close_column] and \
                data[i, close_column] > data[i - 1, close_column] and \
                data[i - 1, close_column] > data[i - 2, close_column] and \
                (data[i, open_column] - data[i, close_column]) > (data[i \
                - 1, open_column] - data[i - 1, close_column]) and \
                (data[i - 1, open_column] - data[i - 1, close_column]) > \
                (data[i - 2, open_column] - data[i - 2, close_column]) and \
                data[i, close_column] > data[i, lower_k_envelope] and \
                data[i, close_column] < data[i, upper_k_envelope]:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 11-5 shows a signal chart with K's envelopes visible on the chart. The signal has been generated due to the appearance of the Euphoria pattern within the envelopes.

The strategy borrows from the field of graphical analysis (support and resistance levels) and applies the Euphoria modern candlestick pattern to confirm the demand zone or the supply zone.

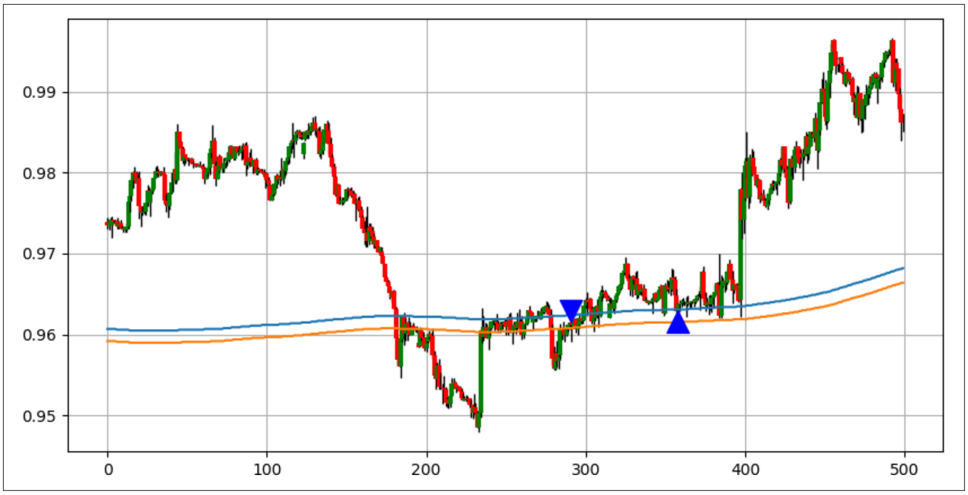


Figure 11-5. Signal chart of the strategy using K's envelopes and the Euphoria pattern

Combining the Barrier Pattern with the RSI-ATR

What exactly is the RSI-ATR? Well, you already know that the RSI is a momentum indicator and that the ATR is a volatility indicator. You have also seen how to code them and use them accordingly. However, fusing them into one indicator is something you have not seen yet. The RSI-ATR is a structured indicator composed of the directional elements of the RSI and the deflating/inflating properties of the ATR. It is used in the same way as the RSI.

A *structured indicator* is a technical indicator composed of two or more indicators. Examples of a structured indicator are the RSI-ATR and the stochastic-RSI (an indicator that has the stochastic oscillator and the RSI as building blocks).

To calculate the RSI-ATR (by default, the lookback period is 14), follow these steps:

1. Calculate an RSI on the market price.
2. Calculate an ATR on the market price using the formula as you know it.
3. Divide the RSI from the first step by the ATR from the second step.
4. Calculate an RSI on the results of the previous step.

Hence, the RSI-ATR is an RSI calculated on the ratio between the RSI (applied on the market price) and the ATR. This is why it is a volatility-weighted indicator. The following snippet shows how to create the indicator in Python:

```
def rsi_atr(data, lookback_rsi, lookback_atr, lookback_rsi_atr, high,
            low, close, position):

    data = rsi(data, lookback_rsi, close, position)

    data = atr(data, lookback_atr, high, low, close, position + 1)

    data = add_column(data, 1)

    data[:, position + 2] = data[:, position] / data[:, position + 1]

    data = rsi(data, lookback_rsi_atr, position + 2, position + 3)

    data = delete_column(data, position, 3)

    return data
```

Figure 11-6 shows the five-period RSI-ATR applied on USDCHF.

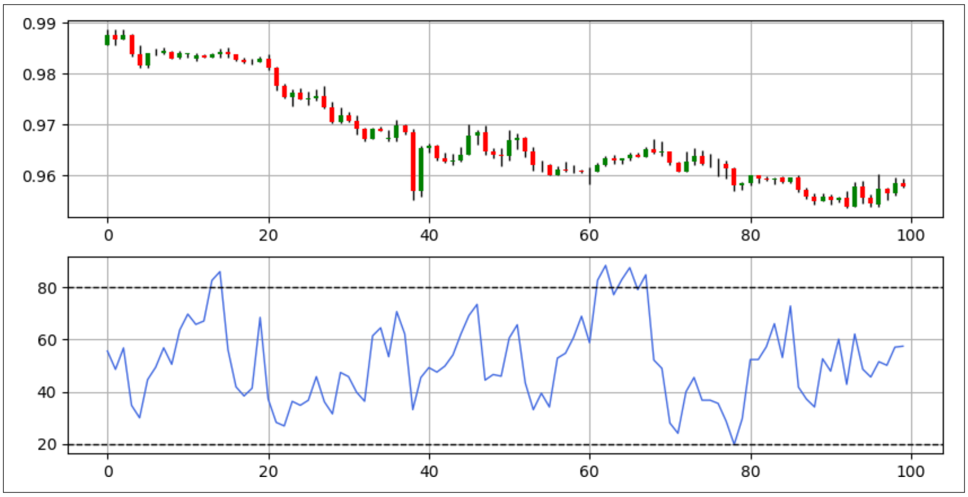


Figure 11-6. USDCHF with the 14-period RSI-ATR

You must have noticed that there are three parameters when dealing with the RSI-ATR as opposed to one parameter with the RSI. The first parameter on the RSI-ATR is the lookback period of the RSI, the second parameter is the lookback of the ATR, and the third parameter is the lookback of the final RSI.

The RSI-ATR is characterized by more volatility and rate of change than the original RSI, thus making it much more reactive to market fluctuations. On the other hand, you use it the same way you would use the RSI.

Remember that the Barrier pattern is a modern configuration (see Chapter 7) and is simply an embodiment of the concept of support and resistance levels.

The trading conditions of the strategy are as follows:

- A long signal is generated whenever a bullish Barrier pattern appears while the 5-period RSI-ATR is below 20.
- A short signal is generated whenever a bearish Barrier pattern appears while the 5-period RSI-ATR is above 80.

The following code snippet shows how to code the signal function of the strategy:

```
def signal(data, open_column, close_column, indicator_column,
           buy_column, sell_column):

    data = add_column(data, 5)

    for i in range(len(data)):

       try:

           # Bullish setup
           if data[i, close_column] > data[i, open_column] and \
              data[i, close_column] < data[i - 1, open_column] and \
              data[i, close_column] > data[i - 1, close_column] and \
              data[i, open_column] < data[i - 1, close_column] and \
              data[i - 1, close_column] < data[i - 1, open_column] and \
              data[i - 2, close_column] < data[i - 2, open_column] and \
              data[i, indicator_column] < lower_barrier:

                    data[i + 1, buy_column] = 1

           # Bearish setup
           elif data[i, close_column] < data[i, open_column] and \
                data[i, close_column] > data[i - 1, open_column] and \
                data[i, close_column] < data[i - 1, close_column] and \
                data[i, open_column] > data[i - 1, close_column] and \
                data[i - 1, close_column] > data[i - 1, open_column] and \
                data[i - 2, close_column] > data[i - 2, open_column]  and \
                data[i, indicator_column] > upper_barrier:

                    data[i + 1, sell_column] = -1

       except IndexError:

            pass

    return data
```

Figure 11-7 shows a signal chart on AUDNZD.

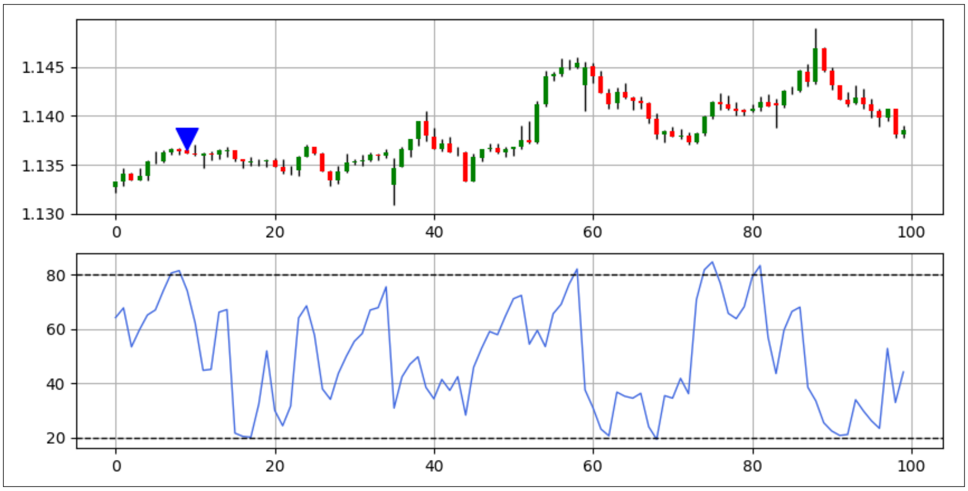


Figure 11-7. Signal chart of the strategy using RSI-ATR with the Barrier pattern

To sum up, the RSI-ATR is an interesting indicator that offers volatility-weighted information of the RSI. Combining the indicator with a candlestick pattern that is based on the concept of support and resistance levels could yield strong trade configurations.

CHAPTER 12

Risk Management and Trading Psychology

Risk management is one of the pillars of any trading system, and in fact it may be the most important pillar. Finding a predictive and profitable trading strategy is what will make you money, but a sound risk management system is what will allow you to keep it.

This chapter discusses the subjective part of the trading process and explains the known biases that you may encounter when trading.

Basics of Risk Management

Risk management does not have to be complex. You can create a solid risk management system by using the few simple rules given in this section.

Stops and Targets

As previously discussed in the book, a *stop-loss order* is an automatic order set during the trade initiation to ensure a minimal predetermined loss. For example, if you buy a few contracts on gold at $1,500 in anticipation that it will go to $2,000, you may place your stop-loss order at $1,250 so that if the price drops to $1,250, you limit your losses to no more than $250. At the same time, your anticipation of selling when the market price reaches $2,000 is known as a *take-profit order* (*target*).

The most basic risk management system for any trade is setting proper stops and targets so that you are framing your expectations and limiting your risks.

One rule that you must never violate: always place stops and targets.

Trailing Stops

A *trailing stop* is a dynamic stop-loss order that follows the market price whenever it goes in the expected direction, thus ensuring less loss and at some point in time ensuring a no-loss trade. The best way to understand the trailing stop order is through an example. Suppose you go long on EURUSD at $1.0000 to target $1.0500. You set your stop-loss order at $0.9900 to limit your losses. Now suppose two days have passed and the current market price is at $1.0230 and you want to ensure your trade never loses money. The solution to this is a trailing stop order where you move your stop from $0.9900 to $1.0000 so that, in the worst case, when the market drops back to $1.0000, you close out and break even. Another way of ensuring at least some profit is to move the stop to $1.0100, thus ensuring a profitable trade no matter what. Of course, the closer you move your stop to the current market price, the more likely you are to be stopped out (of profitably in this case).

It is always a good idea to use trailing stops with trend-following strategies.

Position Sizing

Position sizing is the choice of the number of contracts or lots to allocate for each trade. Some traders use the same size for all their trades, while others prefer to weight them differently depending on one or more factors. There are many ways of sizing your positions, and some use historical data, which takes into account your previous results to optimize the size. The following is a list of the techniques:

Fixed amount
: This is the simplest sizing technique as it does not require any effort. For example, you do not need to choose the number of contracts or how much money to allocate to trades. For example, you can invest $1,000 in each trade regardless of your capital or of the risk. This may be considered a simplistic technique, but some traders prefer to direct their efforts elsewhere and leave position sizing neutral.

Percentage of portfolio
: This technique is a more dynamic version of the fixed-amount technique as the positions the trader takes grow and shrink since they are pegged to the portfolio's size. For example, you have a 3% position size rule. If your portfolio is $100,000, then you should take $3,000 trades. If the market value of the portfolio changes to $120,000, then you should take $3,600 trades.

Conviction-wise

This is a subjective technique of sizing positions that is based on the degree of conviction of the trader. For example, if you have more faith in buying EURUSD than in buying Microsoft stocks, then you might allocate more money into the EURUSD trade. The difference in amounts depends on you.

Kelly criterion

This is an objective mathematical formula that tells you how much to invest by using your historical results. Created by John Kelly, the formula is mainly used in horse betting, but it is also used in trading. The formula is as follows:

$$\text{Kelly weight} = W - \frac{(1 - W)}{R}$$

Here, *Kelly weight* is the percentage size of the trade you should take with regard to your portfolio's overall value, *W* is the historical hit ratio, and *R* is the ratio between the average gain and the average loss. See the following example:

- Historical hit ratio of the past 100 trades = 56%
- Historical loss ratio of the past 100 trades = 44%
- Average gain per trade = $110
- Average loss per trade = $100
- Win-to-loss ratio = 1.1

What is the recommended trade allocation if you find an opportunity?

By using the formula, the answer is 16%. Naturally, the Kelly criterion is not a perfect measure as sometimes it tends to give big weights that are suboptimal with regard to diversification. The user must exercise caution when using the Kelly criterion. You can also use a deflator to decrease the size given by the formula.

The hit ratio technique: This is a personal technique that I use from time to time. It is based on the *clustering of performance*, which means that, generally, profitable trades come together and so do bad trades. This can be true especially because some strategies perform well during certain periods and bad during others. In essence, this is like saying that traders sometimes have good periods and sometimes they are on the losing side, as happens to the most experienced veterans. Table 12-1 illustrates how to use the hit ratio technique to size your positions.

Table 12-1. Chronological hit ratio measures

Element	t	t+1	t+2	t+3	t+4	t+5
Hit ratio	50%	60%	40%	40%	70%	60%
Position size	$5,000	$6,000	$4,000	$4,000	$7,000	$6,000

The table shows the recommended position size with regard to a default size of $10,000 per trade. Basically, when you have a historical hit ratio of 50%, the system recommends 50% allocation of the default size, or $5,000. The system rewards the trader with more allocation and size in case of good results.

Economic Calendar

The economic calendar is the simplest risk management technique as it is a form of risk avoidance. The *economic calendar* shows the important news releases that are expected to have an impact on the markets. Table 12-2 shows a hypothetical example of an economic calendar on a certain day.

Table 12-2. Economic calendar

Time	Country	Event	Impact	Previous value	Expected value	Actual value
9:00 AM	United Kingdom	CPI	High	1.00%	1.20%	1.10%
11:30 AM	Germany	Core CPI	High	0.50%	0.75%	0.75%
4:30 PM	USA	Initial Jobless Claims	Low	232,000	215,000	229,000
7:30 PM	USA	Interest Rate Decision	High	1.50%	1.50%	TBA

Some traders trade based on the news, and therefore they like to trade ahead of news releases or seconds after them in order to profit from volatility. Risk management-wise, this is not recommended as fluctuations are random around news events, which can surprise the markets from time to time.

The best thing to do is to avoid trading around the times of an important release that has historically caused some extreme variations. Examples may include political announcements, GDP numbers, and FOMC meetings.

FOMC stands for Federal Open Market Committee. This committee oversees the country's open market treasury operations and makes decisions about the interest rate.

Behavioral Finance: The Power of Biases

Behavioral finance, a field stemming from behavioral economics, attempts to explain market anomalies and traders' actions. By having a deep understanding of behavioral finance, you will better understand why the market reacts the way it does, especially around certain events and levels. According to behavioral finance, patterns are psychological in nature and are therefore caused by behavioral traits.

Financial markets are composed of the actions and reactions of different human and robotic participants, which makes for a psychological and quantitative stew. This explains the low signal-to-noise ratio—in other words, why it is difficult to accurately forecast the market on a regular basis. These actions and reactions are also referred to as *biases*, and they are human shortcomings in response to certain events.

Biases are the main protagonists of this section, and they are divided into two categories:

Cognitive biases
: These biases come from a lack of knowledge. Cognitive biases generally involve incorrect conclusions based on wrong market assumptions or bad research.

Emotional biases
: These biases are mostly feelings driven and are impulsive in nature. They are not caused by a lack of education but by a lack of self-control and self-management.

Cognitive Biases

This subsection lists some of the most common cognitive biases and their market impact and offers recommendations on how to avoid them:

Conservatism bias
: This occurs when a market participant is slow to react to new information and places too much weight on base rates. It is a type of failure-to-adapt. The way to deal with this bias to force yourself to be skeptical of the basic analysis and to always be dynamic and ready for change. The market does not always behave as it did in the past since it is forward looking.

Confirmation bias
: This occurs when the trader focuses on the type of information that benefits their ongoing position and dismisses the type of information that is not favorable to their position. This is by far one of the most common biases and is actually a normal state of mind that leads to overconfidence over time. The best way to remedy this is to remain impartial and neutral, which is easier said than done. Another way is to automate the decision-making process through scorecards highlighting the attractiveness (or not) of the analyzed underlying assets. Humans in general suffer from this bias—it is not just finance specific.

Illusion of control bias
: This occurs when the trader overestimates their ability to control the trade outcome. Mainly caused by a streak of winning trades, this bias can lead to concentrated positions due to a sense of power over the invested asset. Markets are composed of a huge number of participants with trillions of invested dollars and as such are unlikely to be impacted by any single person (there are a few, very rare exceptions involving small and illiquid assets). The way to remove the

illusion of control is to stay focused and humble and understand that you are facing a semirandom environment that changes its dynamics and drivers every day.

Hindsight bias

This occurs when a trader overestimates their past accuracy and can lead to excessive risk-taking. It is easy to look at past charts and conclude that the subsequent direction was obvious. Most back-tests include a form of hindsight bias since conditions were created at the end of the analyzed period. Market technicians overestimate their abilities when they see that some techniques worked well in the past but fail to take into account the environment of the tested period. Also, some configurations do not look the same when they're happening as when they are complete. Hindsight bias can be hard to cure, but the best way is to take into consideration the variables that were present during the analyzed period to simulate the past environment in a more realistic way.

Anchoring bias

This occurs when the trader's opinion is anchored to a certain base point and fails to change to incorporate new information. As I have mentioned, the analyst or trader must maintain a dynamic and open mindset. The best way to cure this bias is to stay up-to-date with regard to new information and pieces of data.

Mental accounting bias

This occurs when traders give different values to the same amount of money because it is allocated into different funds. For example, people generally treat money earned through work and money earned without work differently. This is a cognitive bias. The best way to remedy this is to treat all money as fungible when allocating it to separate portfolios.

Availability bias

This occurs when the trader selects positions based on how easily the trader can retrieve memories of them. This means that familiar assets are more attractive than nonfamiliar assets, which is a wrong assumption as opportunities may come from any type of market. It is a type of mental shortcut where the trader does not expend much effort in research. To cure this bias, you must practice full due diligence before selecting a universe of investable assets. Do not just trade the EUR-USD because you are familiar with it: expand your horizons.

Loss aversion bias[1]

This occurs when the pain of losing is greater than the joy of winning. It is by far the most common bias. Humans are known to prefer not to lose money than to gain money as was demonstrated in "Prospect Theory: An Analysis of Decision Making Under Risk" by Daniel Kahneman and Amos Tversky (1977). Loss

1 This is also considered as an emotional bias.

aversion could lead to decreased risk taking and therefore decreased expected return. However, the most significant effect is on the stop levels. Loss-averse humans do not want to accept that they are losing money and will see a losing position as an ongoing position and, therefore, prefer to wait until it turns positive. This is highly dangerous as leaving positions without a stop-loss levels could lead to disastrous results. In addition, some people close out of winning positions too fast in fear that they will turn negative (a form of fear of regret). The best way to handle loss aversion is to automate the risk management process and to respect the stop-loss and target orders established at the initiation of the trade.

Whenever there is money to be made, irrationality looms nearby.

Emotional Biases

This subsection lists some of the most common emotional biases and their impact. As a reminder, while cognitive biases are related to a lack of knowledge, emotional biases are related to psychological traits:

Overconfidence bias
: This occurs when a trader enjoys a winning streak and believes it is due to their superior ability to trade the markets, which leads to holding concentrated positions and excessive trading. A good streak will come to an end, and thus the trader must always follow procedures and ensure they do not stray from the strategy.

Regret-aversion bias
: This refers to staying in low-risk investments out of fear. This really is all about the risk profile of the trader. There is no right or wrong answer, but the fear of regret can make the trader lose out on interesting opportunities. You should take risks to make money, but only calculated risks.

Endowment bias
: This occurs when the trader believes that the owned assets are more valuable than the ones that are not owned. This may hamper the trader's opportunities and cause them to be limited to the assets they already own, even if they decay over time. The market has opportunities everywhere, and participants must always be on the lookout for the next big thing.

Trading Framework

Theoretical trading topics, paper trading, and real trading are all different. The aim of any trader is to make real money, and this is harder than it seems. Let's define these three concepts:

Theoretical trading topics
: This is where the trader thinks about trading strategies and simplifies them in their head. This is where biases are formed, especially hindsight bias.

Paper trading
: This is when the trader simulates trades using a platform. It is very similar to real trading, but the money used is not real, and therefore, it fails to simulate the rush of real-life emotions.

Real trading
: This is when the trader uses actual money through an actual broker to buy and sell assets in an attempt to increase wealth.

Real trading has the following chronological steps:

1. *Idea generation*: This is where strategy ideas are formed. For example, you think about combining the Tweezers pattern with a technical indicator in a certain market regime. You have formed an idea, and now you want to back-test it to see if it performed well in the past.
2. *Back-testing*: In this phase, you evaluate past results and decide whether to take the strategy live or not. This is perhaps the most important step because it involves many factors, such as proper back-testing practices and validation, as well as back-testing with risk management techniques.
3. *Pre-trade management*: This is where you prepare the trade setups, such as the stop-loss and take-profit orders, and decide whether to use trailing stops. Also, you must decide on position sizing using one of the techniques discussed earlier in the chapter.
4. *Trade management*: This is where you monitor your profit and loss statement and ongoing trades in order to evaluate your situation. Discipline is paramount in this step as you must not let your emotions bias your judgment and make you close out too early or too late.

This marks the end of this chapter where you learned a variety of information about how to better manage trades. Trading is a mind game, and you have the responsibility to overcome your emotional and cognitive obstacles through proper training so that you maximize your chances of survival.

This also marks the end of the adventure on candlestick patterns. I would like to thank you and congratulate you for finishing this book and for taking the time to understand my research and techniques. Patterns, with all their formidability and sophistication, must be used with caution and in appropriate combinations with other technical tools. As you saw in this chapter, a trading system must comprise idea generation, back-testing, risk management, and trade management.

If you want to dig deeper into the world of technical pattern recognition and especially candlestick patterns, make sure to start with the rationale of any configuration. The *why* is as important as the *what*. Python is an extremely powerful and versatile coding language and will facilitate your research. As such, I recommend mastering it in order to speed up your research.

Index

About the Author

Sofien Kaabar is a financial author, trading consultant, and institutional market strategist specializing in the currencies market with a focus on technical and quantitative topics. Sofien's goal is to make technical analysis objective by incorporating clear conditions that can be analyzed and created with the use of technical indicators that rival existing ones.

Having elaborated many successful trading algorithms, Sofien is now sharing the knowledge he has acquired over the years to make it accessible to everyone.

Colophon

The animal on the cover of *Mastering Financial Pattern Recognition* is a Eurasian hoopoe (*Upupa epops*).

These birds are easily recognized by their orange-brown bodies and their black-and-white striped wings and tail. An elegant crest with black-and-white tips crowns their head. When lowered, the feathers of their crest create a spike that points away from their forehead. Their bill, long and narrow, is perfect for foraging in the ground for insects, spiders, and small vertebrates and invertebrates.

The Eurasian hoopoe can be found in several parts of Africa, Europe, and Asia. They prefer to live in areas with semi-open space for foraging and trees with cavities to nest in, such as heathland, savannas, grasslands, and forest glades. They are also well adapted to living in human-made habitation such as parks, orchards, and vineyards.

There are several superstitions that involve hoopoes. According to the English, parts of its body were used to perform black magic to summon demons and poison people. In Egypt, the hoopoe is affiliated with the ability to restore youth. The song of the hoopoe has been said to predict the weather. Notably, the hoopoe coined the phrase "A little bird told me" when, according to the Quran, it brought news of the Queen of Sheba.

The current Eurasian hoopoe population is relatively stable but slowly decreasing. Many of the animals on O'Reilly covers are endangered; all of them are important to the world.

The cover illustration is by Karen Montgomery, based on a black-and-white engraving from *British Birds*. The cover fonts are Gilroy Semibold and Guardian Sans. The text font is Adobe Minion Pro; the heading font is Adobe Myriad Condensed; and the code font is Dalton Maag's Ubuntu Mono.