

Financial Trading in R

Boni

9/6/2020

The mechanics of trading usually come in one of two forms:

- Trend trading (also divergence or momentum), which is a bet that a quantity, such as a price, will keep moving in its current direction.
- Reversion trading (also convergence, cycle, or oscillation), which is a bet that a quantity, such as a price, will reverse.

How to prevent overfitting

When developing a trading system, a major pitfall that can creep into system development is the desire to find a strategy that worked phenomenally in the past. This is known as overfitting. Research by leading authors in the quantitative field has shown that not only is an overfitted system unlikely to generate profits in the future, but also that its performance can lead to losses. Some steps can be done to prevent overfitting:

- Examining the robustness of system performance.
- Reducing the number of parameters in the trading system.
- Conducting tests to determine statistical significance of a strategy.

Getting financial data

2 Types important ETF: LQD and SPY.

```
# Get SPY from yahoo
getSymbols("SPY",
           from = "2000-01-01",
           to = "2016-06-30",
           src = "yahoo",
           adjust = TRUE)
```

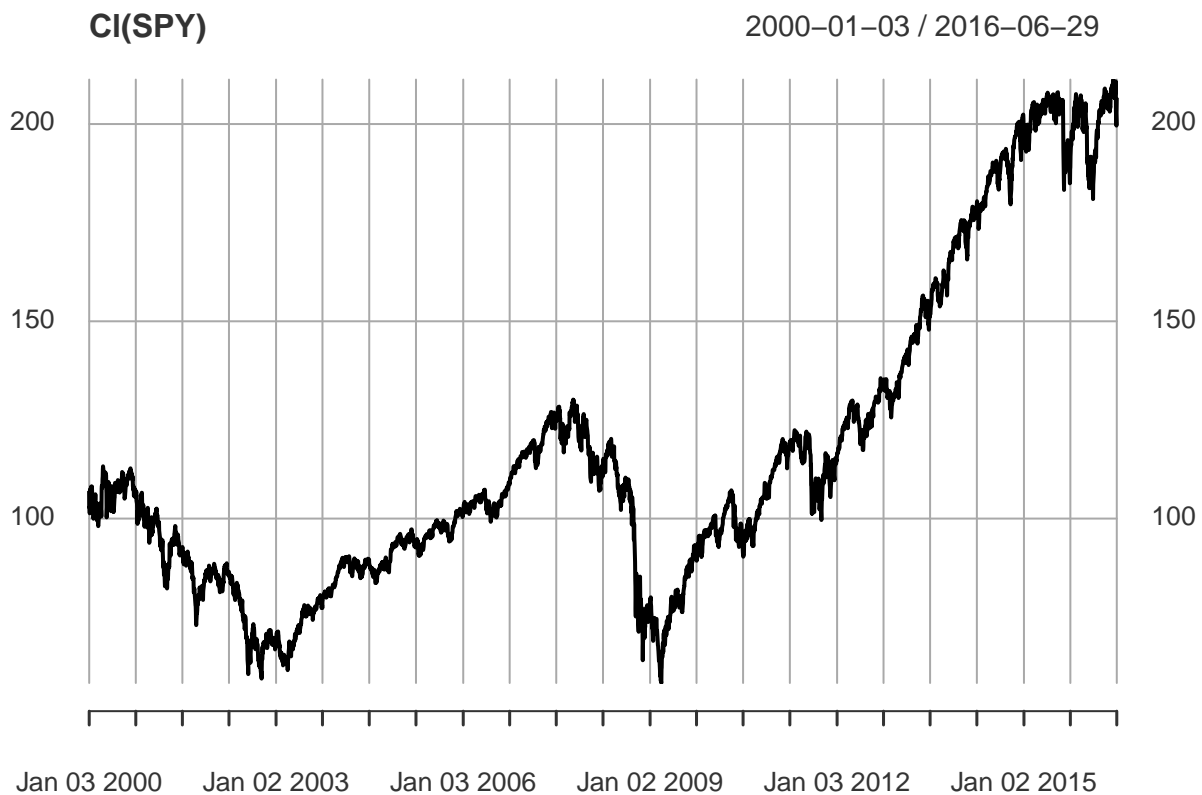
```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SPY?
## period1=-2208988800&period2=1599436800&interval=1d&events=split&crumb=eB.rUTVibEa'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SPY?
## period1=-2208988800&period2=1599436800&interval=1d&events=split&crumb=eB.rUTVibEa'

## [1] "SPY"

# Plot the closing price of SPY
plot(CI(SPY))
```



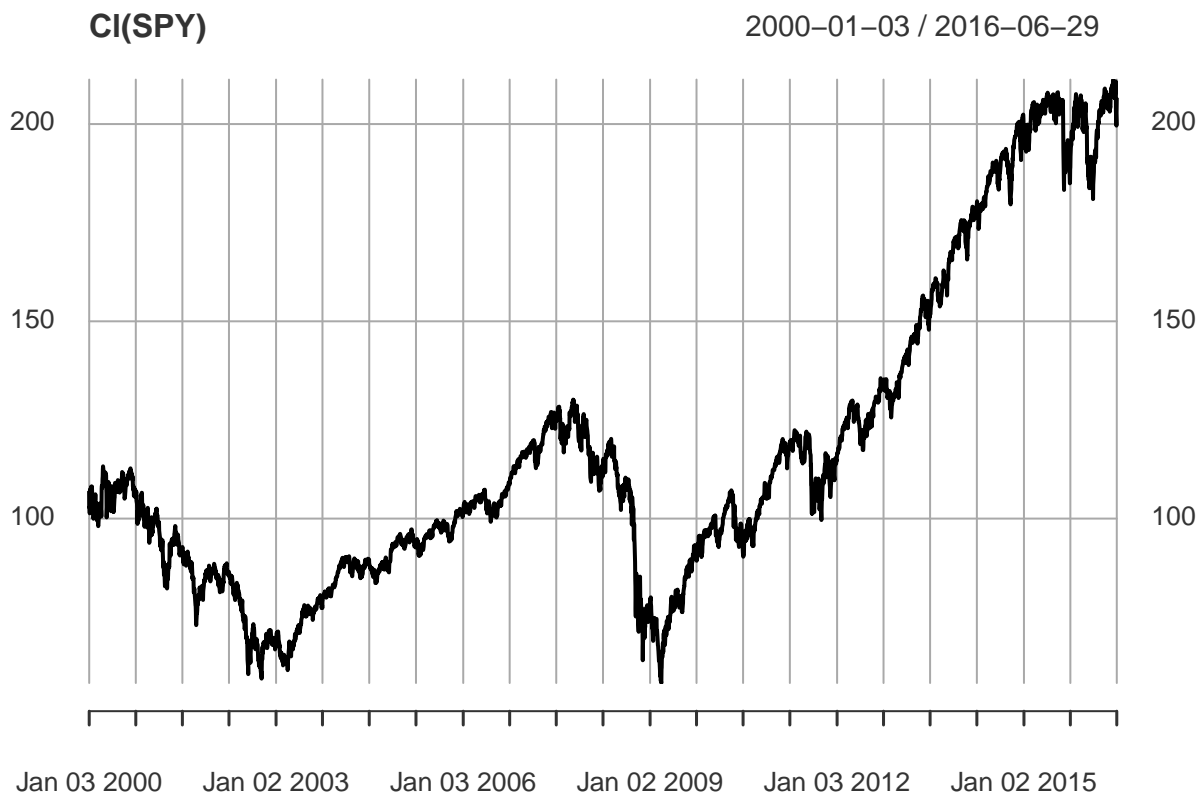
Adding a moving average to financial data

One of the most popular indicators to add to a trading strategy is the 200-day simple moving average (SMA). This is a technical indicator of the average closing price of a stock over the past 200 days. Other moving averages can be of varying length, such as 50-day, 100-day, etc.

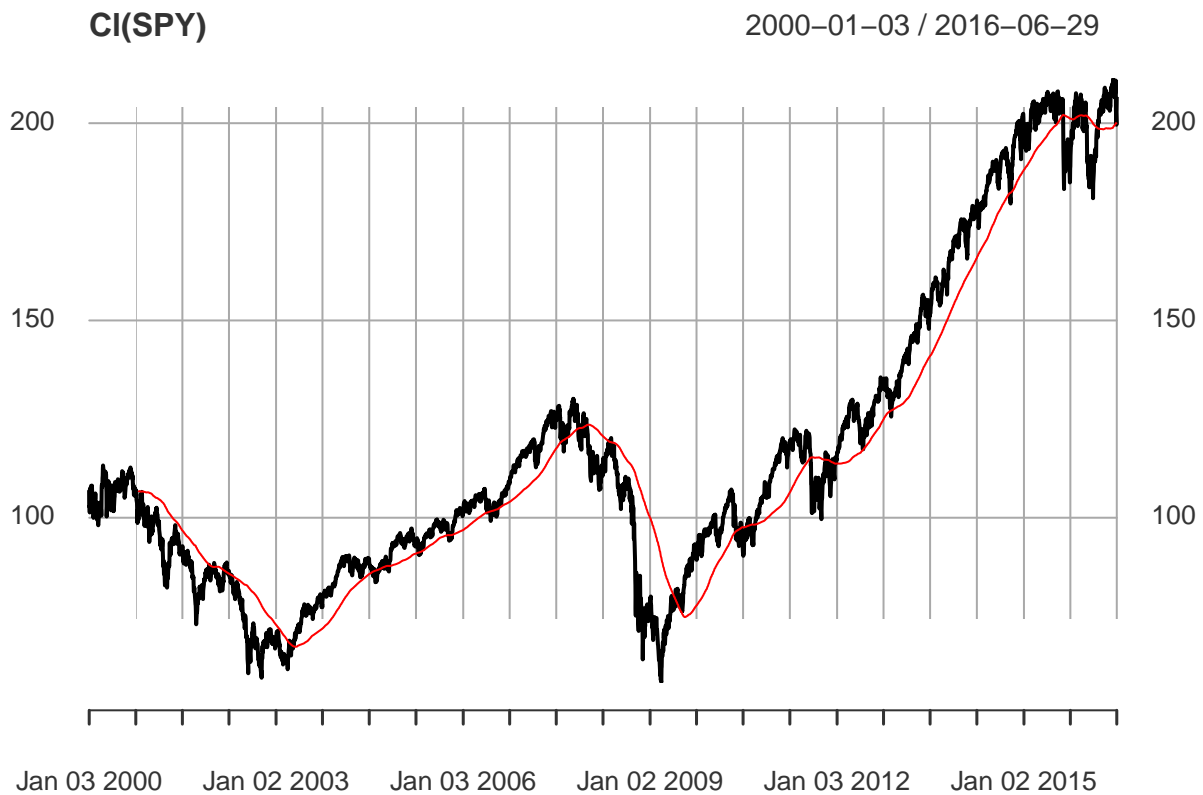
Whenever the price is above the 200-day moving average, a whole assortment of good things usually happen, such as the asset appreciating in price, low volatility, and so on. Getting a long-lasting visual might shed light on why this indicator is mentioned so often.

The TTR package has a function that calculates moving averages, `SMA()`, which takes in a price series `x` and computes the arithmetic mean over `n` days. A call of `SMA()` with a lookback window of 50 days could look like the following: `SMA(Cl(GDX), n = 50)`

```
# Plot the closing prices of SPY  
plot(Cl(SPY))
```



```
# Add a 200-day SMA using lines()  
lines(SMA(Cl(SPY), n = 200), col = "red")
```



Initializing setting quanstrat

Define boilerplate code

Let's get started with creating our first strategy in quantstrat. In this exercise, you will need to fill in three dates:

An initialization date for your backtest. The start of your data. The end of your data. The initialization date must always come before the start of the data, otherwise there will be serious errors in the output of your backtest.

You should also specify what time zone and currency you will be working with with the functions `Sys.setenv()` and `currency()`, respectively. An example is here:

```
Sys.setenv(TZ = "Europe/London") currency("EUR")
```

```
# Load the quantstrat package
library(quantstrat)
```

```
## Loading required package: blotter
```

```
## Loading required package: FinancialInstrument
```

```
## Loading required package: PerformanceAnalytics
```

```
##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend

## Loading required package: foreach

# Create initdate, from, and to strings
initdate <- "1999-01-01"
from <- "2003-01-01"
to <- "2015-12-31"

# Set the timezone to UTC
Sys.setenv(TZ = "UTC")

# Set the currency to USD
currency("USD")

## [1] "USD"

# Load the quantmod package
library(quantmod)

# Retrieve SPY from yahoo
getSymbols("SPY", src = "yahoo", from = from, to = to, adjust = TRUE)

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SPY?
## period1=-2208988800&period2=1599436800&interval=1d&events=split&crumb=eB.rUTVibEa'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SPY?
## period1=-2208988800&period2=1599436800&interval=1d&events=split&crumb=eB.rUTVibEa'

## [1] "SPY"

# Use stock() to initialize SPY and set currency to USD
stock("SPY", currency = "USD")

## [1] "SPY"
```

Understanding initialization settings - III

Let's continue the setup of your strategy. First, you will set a trade size of 100,000 USD in an object called `tradesize` which determines the amount you wager on each trade. Second, you will set your initial equity to 100,000 USD in an object called `initeq`.

Quantstrat requires three different objects to work: an account, a portfolio, and a strategy. An account is comprised of portfolios, and a portfolio is comprised of strategies. For your first strategy, there will only be one account, one portfolio, and one strategy. Let's call them "firststrat" for "first strategy".

Finally, before proceeding, you must remove any existing strategies using the strategy removal command `rm.strat()` which takes in a string of the name of a strategy.

```
# Define your trade size and initial equity
tradesize <- 100000
initeq <- 100000

# Define the names of your strategy, portfolio and account
strategy.st <- "firststrat"
portfolio.st <- "firststrat"
account.st <- "firststrat"

# Remove the existing strategy if it exists
rm.strat(strategy.st)
```

Understanding initialization settings - IV

Now that everything has been named, you must initialize the portfolio, the account, the orders, and the strategy to produce results.

The portfolio initialization `initPortf()` needs a portfolio string name, a vector for symbols used in the backtest, an initialization date `initDate`, and a currency. The account initialization call `initAcct()` is identical to the portfolio initialization call except it takes an account string name instead of a new portfolio name, an existing portfolio's name, and an initial equity `initEq`. The orders initialization `initOrders()` needs a portfolio string and an initialization date `initDate`. The strategy initialization `strategy()` needs a name of this new strategy and must have `store` set to `TRUE`.

```
# Initialize the portfolio
initPortf(portfolio.st, symbols = "SPY", initDate = initdate, currency = "USD")

## [1] "firststrat"

# Initialize the account
initAcct(account.st, portfolios = portfolio.st, initDate = initdate, currency = "USD", initEq = initeq)

## [1] "firststrat"

# Initialize the orders
initOrders(portfolio.st, initDate = initdate)

# Store the strategy
strategy(strategy.st, store = TRUE)

# Everything up and running now.
```

Introduction to indicators

The SMA and RSI functions

An indicator is a transformation of market data that is used to generate signals or filter noise. Indicators form the backbone of many trading systems, and the system you will build in this course uses several of them.

The simple moving average (SMA) and relative strength index (RSI) are two classic indicators. As you saw in Chapter 1, the SMA is an arithmetic moving average of past prices, while the RSI is a bounded oscillating indicator that ranges from 0 to 100. Their respective functions `SMA()` and `RSI()` both take in a series of prices, denoted by `x` and `price` respectively, and a lookback period `n`, for example:

```
SMA(x = Cl(GDX), n = 50) RSI(price = Cl(GDX), n = 50)
```

```
# Create a 200-day SMA
spy_sma <- SMA(x = Cl(SPY), n = 200)

# Create an RSI with a 3-day lookback period
spy_rsi <- RSI(price = Cl(SPY), n = 3)
```

Visualize an indicator and guess its purpose - I

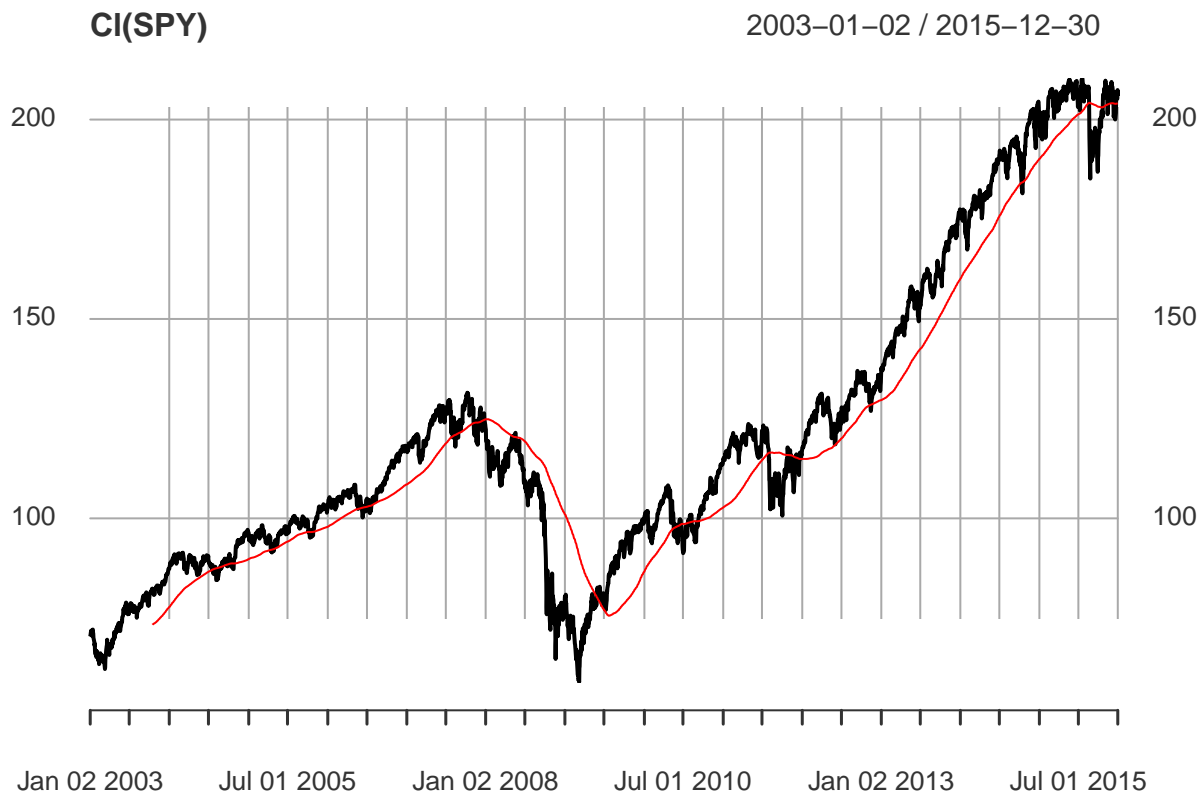
Now you will visualize these indicators to understand why you might want to use the indicator and what it may represent. Recall that a trend indicator attempts to predict whether a price will continue in its current direction, whereas a reversion indicator attempts to predict whether an increasing price will soon decrease, or the opposite.

In this exercise, you will revisit the 200-day SMA. As a refresher, this indicator attempts to smooth out prices, but comes with a tradeoff - a lag. You will plot the prices of SPY and plot a 200-day SMA on top of the price series. Using this information, you will determine if it is a trend indicator or reversion indicator.

```
# Plot the closing prices of SPY
plot(Cl(SPY))
```



```
# Overlay a 200-day SMA  
lines(SMA(CI(SPY), n = 200), col = "red")
```

```
# What kind of indicator?
"trend"
```

```
## [1] "trend"
```

```
# SMA is a trend indicator rather than reversion.
```

Visualize an indicator and guess its purpose - II

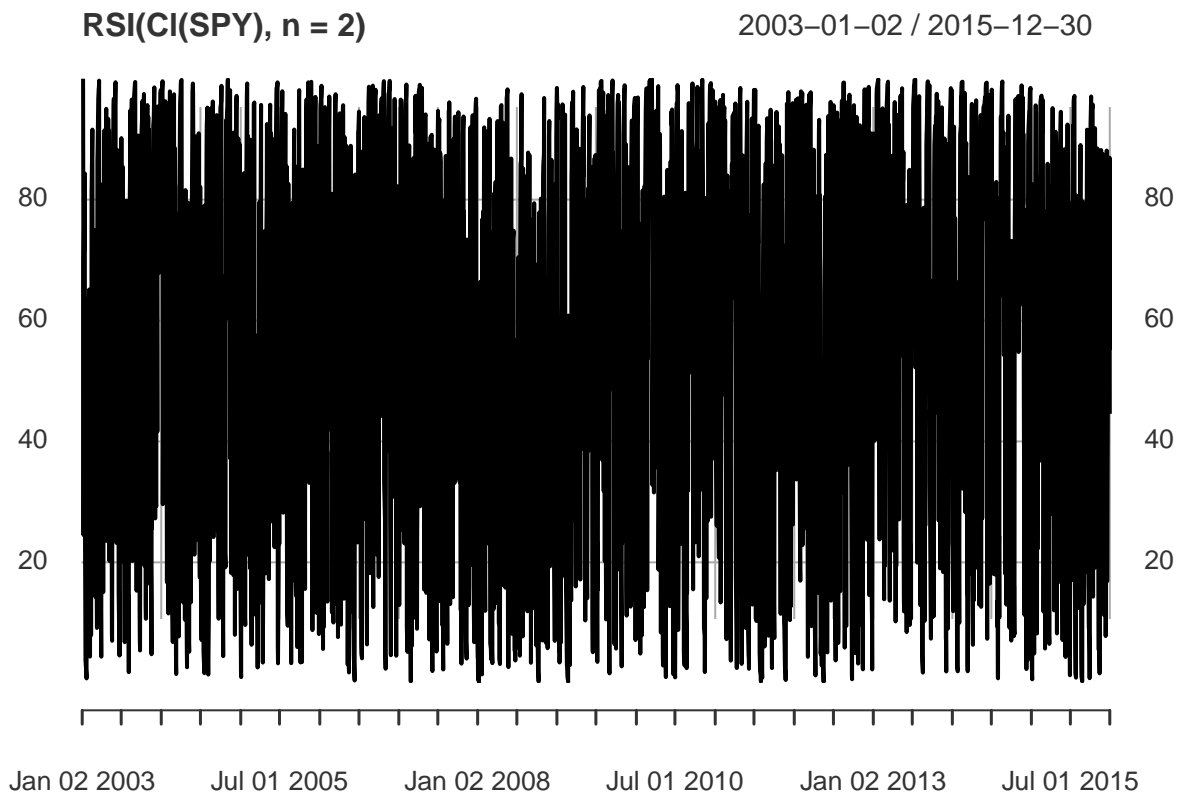
The Relative Strength Index (RSI) is another indicator that rises with positive price changes and falls with negative price changes. It is equal to $100 - 100/(1 + RS)$, where RS is the average gain over average loss over the lookback period. At various lengths, this indicator can range from a reversion indicator, to a trend filter, or anywhere in between. There are various ways of computing the RSI.

As you already know, `RSI()` takes in a price series price and a number of periods `n` which has a default value of 14. Some traders believe that the 2-period RSI, also known as the RSI 2, has even more important properties than the 14-period RSI.

```
# Plot the closing price of SPY
plot(Cl(SPY))
```



```
# Plot the RSI 2  
plot(RSI(CI(SPY), n = 2))
```



```
# What kind of indicator?
"reversion"
```

```
## [1] "reversion"
```

Indicator mechanics

Implementing an indicator - I

At this point, it's time to start getting into the mechanics of implementing an indicator within the scope of the quantstrat library. In this exercise you will learn how to add an indicator to your strategy. For this exercise you will use the strategy you created in earlier exercises, `strategy.st`. For your first indicator, you will add a 200-day simple moving average.

To add an indicator to your strategy, you will use the `add.indicator()`. Set `strategy` equal to the name of your strategy, name to the name of a function in quotes, and arguments to the arguments of the named function in the form of a list. For instance, if your function name was `SMA`, the arguments argument will contain the arguments to the `SMA` function:

```
add.indicator(strategy = strategy.st, name = "SMA", arguments = list(x = quote(CI(mktdata)), n = 500),
label = "SMA500")
```

When referencing dynamic market data in your `add.indicator()` call, include `mktdata` inside the `quote()` function because it is created inside `quantstrat` and will change depending on whichever instrument the package is using at the time. `quote()` ensures that the data can dynamically change over the course of running your strategy.

```
# Add a 200-day SMA indicator to strategy.st
add.indicator(strategy = strategy.st,

               # Add the SMA function
               name = "SMA",

               # Create a lookback period
               arguments = list(x = quote(Cl(mktdata)), n = 200),

               # Label your indicator SMA200
               label = "SMA200")
```

```
## [1] "firststrat"
```