Online tools for demonstration of backtest overfitting

David H. Bailey * Jonathan M. Borwein† Amir Salehipour ‡
Marcos López de Prado § Qiji Zhu ¶

November 29, 2015

Abstract

In mathematical finance, *backtest overfitting* means the usage of historical market data (a *backtest*) to develop an investment strategy, where too many variations of the strategy are tried, relative to the amount of data available. Backtest overfitting is now thought to be a primary reason why quantitative investment models and strategies that look good on paper often disappoint in practice. In this document we introduce two online tools, the *Backtest Overfitting Demonstration Tool*, or BODT for short, and the *Tenure Maker Simulation Tool*, or TMST, which illustrate the impact of backtest overfitting on investment models and strategies. This report describes BODT and TSMT, and the experiments they perform, together with technical details such as the evaluation metrics and parameters used.

Key words: Backtest overfitting; Online interface, Sharpe Ratio; Deflated Sharpe Ratio; Investment strategy

AMS subject classifications: 60E, 62C, 91G10, 91G60, 91G70

1 Introduction

In mathematical finance, *backtest overfitting* means the usage of historical market data (a *backtest*) to develop an investment strategy, where too many variations of the strategy are tried, relative to the amount of data available. Backtest overfitting is now thought to be a primary reason why quantitative investment models and strategies that look good on paper (based on backtests) often disappoint in practice. Models suffering this from condition target the specific idiosyncrasies of a limited dataset, rather than any general behavior, and, as a result, perform rather poorly when presented with new data.

Backtest overfitting is an instance of the more general phenomenon of multiple testing in scientific research, where a large number of variations of a model are tested on the same data, without accounting for the increase in false positive rates. Standard overfitting techniques fail to identify this problem, because they are designed to evaluate the complexity of a model relative to the dataset, still assuming that a single test has taken place.

An example will clarify this difference: Suppose that a new compound XYZ is developed to treat headaches. We wish to test for the hypothesis that XYZ is actually effective. A false positive occurs when we falsely determine that XYZ has been effective. This can occur for a variety of reasons: The patient was misdiagnosed, the pain associated with headache oscillated closely to the threshold use to declare the condition, etc. Suppose that the probability of false positive is only 5%. We could test variations of the compound by changing an irrelevant characteristic (the color, the taste, the shape of the pill), and it is expected that at least 1 in 20 of those variations will be (falsely) declared effective. As the reader can appreciate, the problem does not lie with the complexity of the compound. The researcher has conducted multiple tests while treating each individually (full body scans and other current technology-driven medical diagnoses and methods are often compromised for the same reason).

Likewise, in finance it is common to conduct millions, if not billions, of tests on the same data. Financial academic journals do not report the number of experiments involved in a particular discovery

*Lawrence Berkeley National Laboratory (retired), 1 Cyclotron Road, Berkeley, CA 94720, USA, and University of California, Davis, Department of Computer Science. E-mail: david@davidhbailey.com.

†CARMA, University of Newcastle NSW 2308, Australia. E-mail: jonathan.borwein@newcastle.edu.au.

‡CARMA, University of Newcastle NSW 2308, Australia. E-mail: a.salehipour@newcastle.edu.au.

§Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, USA. E-mail: lopezdeprado@lbl.gov.

¶Department of Mathematics, Western Michigan University, Kalamazoo, MI 49008, USA. E-mail: qiji.zhu@wmich.edu.

(indeed authors will typically not have provided that information), and as a result it is likely that many published investment theories or models are false positives. For example, in [3] the authors show that if only five years of daily stock market data are available as a backtest, then no more than 45 variations of a strategy should be tried on this data, or the resulting strategy will be overfit, in the specific sense that the strategy's Sharpe Ratio or SR is likely to be 1.0 or greater just by "chance." SR is a popular statistic to quantify the performance of an investment strategy [9, 8]. It is the ratio between average excess returns on capital, in excess of the return rate of a risk-free asset, and the standard deviation of the same returns [3]; this implies that the higher the ratio, the greater the return relative to the risk involved. The Sharpe Ratio (SR) and similar statistics are used to allocate capital to the best performing strategy. The major drawback of the SR is that it relies on normally distributed returns [7]; in [6] the authors overcome this drawback by developing a *probabilistic Sharpe Ratio* in which both the skewness and kurtosis of returns distribution are also taken into account. Anyone who develops or even merely invests in a systematic investment strategy (or in an exchange traded fund based on such a strategy) needs to understand the degree to which strategies can be overfit, in order to avoid disappointment and financial losses.

To that end, we have developed the online *Backtest Overfitting Demonstration Tool* (BODT) and *Tenure Maker Simulation Tool* (TMST), which allow one to see first-hand how easy it is to overfit an investment strategy, and how this overfitting may impact the financial bottom-line performance. These two tools stem from two broad types of investment strategies [3]:

1. Those based on general trading rules; example is seasonal opportunities. BODT is developed for this type of investment strategies.

2. Those based on forecasting equations; example is econometric models. TMST is developed for this type of investment strategies.

BODT employs a simplified version of the process many financial analysts use to create investment strategies, namely to use a computer program to find the "best" strategy based on historical market data (often termed "in-sample" (IS) data), by adjusting variables such as the holding period, the profit taking and stop loss levels, etc. Similarly, TMST applies forecasting and econometric equations in order to find the "best" strategy. If care is not taken to avoid backtest overfitting, such strategies may look great on paper (based on tests using historical market data), but then give rather disappointing results when actually deployed on a different dataset (often termed "out-of-sample" (OOS) data). This has been illustrated in Figure 1.

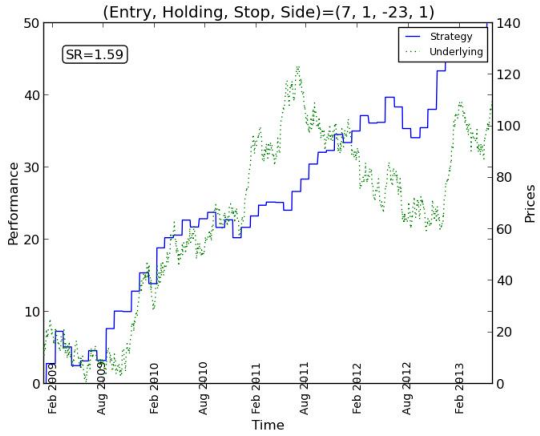
As a brief illustration of our BODT, the left plot in Figure 1 shows how an optimal strategy (associated with the blue line) can be developed based on a historical dataset (which in this case is merely a pseudorandomly generated set of daily closing prices) by varying *entry day*, *holding period*, *stop loss* and *side* parameters (later in Section 2.3 we discuss these parameters in more detail). This optimal strategy has a SR of 1.59 on that in-sample (backtest) dataset. The right plot, on the other hand, shows how the same strategy performs on a new out-of-sample dataset. In this case, the SR is -0.18, indicating rather poor performance indeed — the strategy actually *lost* money here. Figure 2 illustrates the overfitting on one real stock market dataset, namely the S&P500 index. Although, the test data SR (the right plot) is positive, it is much lower than the trial data SR (the left plot), which is 0.85.

The impact of backtest overfitting has been discussed in detail in [1]. For the single testing case, Bailey, Borwein, López de Prado and Zhu [3] proposed the *Minimum Backtest Length* (MinBTL) as a metric to avoid selecting a strategy with a high SR on IS data, but zero or less on OOS data. A *probabilistic Sharpe Ratio* (PSR) was proposed in [6] to calculate the probability of an estimated SR being greater than a benchmark SR. For the multiple testing case, Bailey and López de Prado [2] developed the *Deflated Sharpe Ratio* (DSR) to provide a more robust performance statistic, in particular, when the returns follow a non-normal distribution. In fact, DSR is a PSR wherein the threshold is set so that the impact of all tried strategies is captured as well as the non-normality of the returns' distribution. An earlier version of the online Backtest Overfitting Demonstration Tool, which will be discussed in this paper, was introduced in [4]. This paper extends the study of [4] in several ways:

1. The Deflated Sharpe Ratio (DSR) performance statistic has been computed for the IS data;

2. Outcomes have been updated, and more analyses in the form of plots and numerical values have been reported;

3. Real-world stock market data has been made available for experimentation.



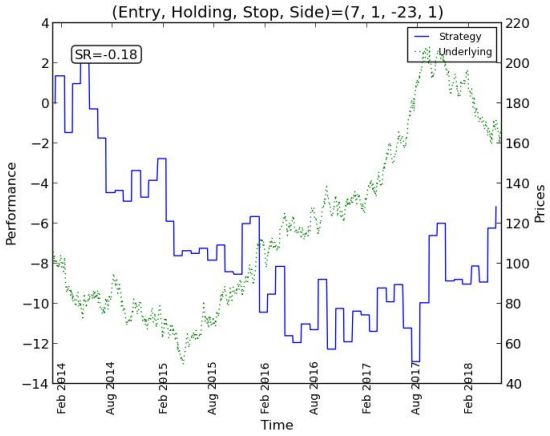
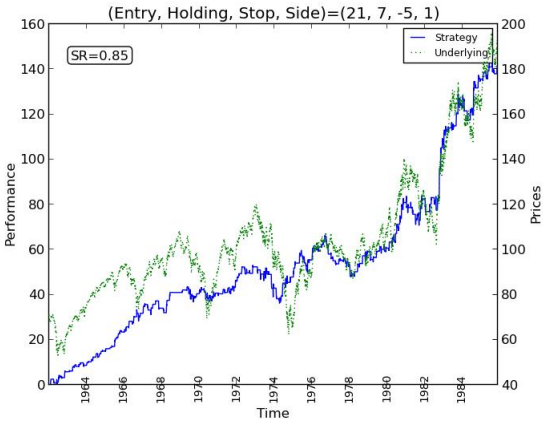


Figure 1: Sharpe Ratio (SR) optimized for the trial data (left) and observed on the test data (right). (The test data SR is not even positive.)



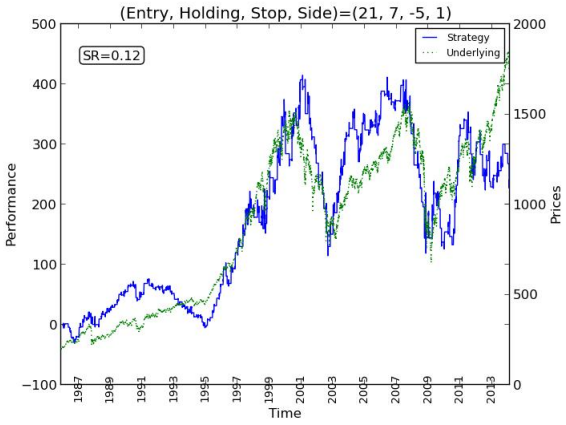


Figure 2: Sharpe Ratio (SR) optimized for the S&P500 trial data (left) and observed on the S&P500 test data (right). (The test data SR is much lower than the trial data SR.)

4. TMST is developed in order to show the impact of the overfitting on econometric models.

Furthermore, while the study of [1] introduces the probability of backtest overfitting (PBO), the online BODT and TMST focus on demonstrating the impact of the overfitting. Finally, our approach is different than that of *portfolio optimization*, in the sense that decisions are not made based on the portfolio's assets. The research on portfolio optimization is rich, and we refer the reader to the recent review article of [5].

The remainder of the note is organized as follows. In Section 2 we explain the Backtest Overfitting Demonstration Tool (BODT). This includes the structure of the tool, the datasets, the parameters, and the types of experiments available to the user. Section 2.6 provides more technical details regarding the operation of the tool, and in particular of the performance statistics Sharpe Ratio (SR) and Deflated Sharpe Ratio (DSR). In addition to this, we illustrate the graphical and numerical outcomes/reports of BODT. The outcomes include a set of plots and a set of numerical values. Similarly, in Section 3 we explain the Tenure Maker Simulation Tool (TMST). This includes the structure of the tool, the parameters, and the types of experiments available to the user. In addition to this, we illustrate the graphical outcomes/reports of TMST. The outcomes include a set of two plots and a movie. The paper ends with a few conclusions.

2 The Backtest Overfitting Demonstration Tool

2.1 The structure of the tool

The core of the *Backtest Overfitting Demonstration Tool*, or BODT for short, is an optimization program coded in programming language Python (the optimization module). The optimization program either generates pseudorandom data or loads real-world stock market data, depending on the type of the experiment chosen by the user (see Section 2.6 for more details), then generates all combinations of different parameters values (a combination is a strategy), and evaluates every strategy by calculating the Sharpe Ratio (SR). The best strategy is also evaluated by computing the more sophisticated metric Deflated Sharpe Ratio (DSR). Finally, it outputs plots and both numerical performance statistics (SR and DSR). These four steps are communicated to the user via an online interface (the communication module). BODT is available to the public for free and can be accessed through https://carma.newcastle.edu.au/backtest/.

In particular, the online interface collects and/or sets the parameters values, supplies them to the optimization program, and reports the outcomes from the optimization program. The online interface has two web pages: the first page asks for the parameters or sets them, depending on the type of the experiment chosen by the user, while the second page reports the outcomes of the optimization program. The outcomes include a movie, three plots and a summary of the numerical values. A brief explanation is available on each page of the online interface. Furthermore, a more detailed explanation is documented in a tutorial which can be downloaded from either page of the online interface.

The online interface and the Python optimization program were constructed by Stephanie Ger, Amir Salehipour, Alex Sim, John Wu and David H. Bailey, based on an earlier Python program developed by Marcos López de Prado. A schematic view of BODT has been depicted in Figure 3.

2.2 Data

BODT works with two types of data: randomly generated daily closing prices and actual daily closing prices. In each case, the sample data is divided into two sets: In-sample (IS) set, also known as the *training set*, and out-of-sample (OOS) set, also known as the *testing set*. Strategies are simulated and evaluated over the IS set. After exploring the *best* performing strategy, it is evaluated over the OOS set. Note that the OOS set is not used in the design of the strategy; thus, a backtest is *realistic* when the IS performance is consistent with the OOS performance, after controlling for the number of experiments that have taken place.

Given two parameters, the *standard deviation* and the *seed*, simulated daily closing prices are derived by drawing returns from a Gaussian distribution with mean zero. The random data is divided into two equally large sets, one for the IS set and the other for the OOS set. The real-world data values are daily closing prices of the S&P500 index between January 1962 and February 2014.

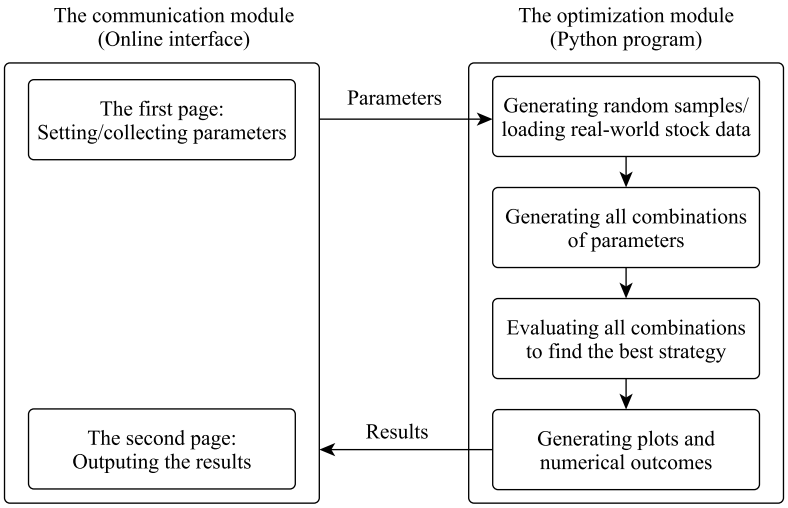


Figure 3: Interaction of the optimization and communication modules of BODT. The optimization module takes parameters from the online interface (the communication module). Upon completion, results are sent to the communication module to be reported to the user.

2.3 Parameters

BODT has seven parameters. As we shall see in Section 2.5, some of these parameters are not available to the user; thus, the user has no control over them; for these parameters, the optimization program uses pre-specified values as discussed below. The seven parameters are:

1. **Maximum holding period:** the number of days that a stock can be held before it is sold.
2. **Maximum stop loss:** the percentage of invested capital that can be lost before the position is closed.
3. **Sample length:** the number of observations used in-sample.
4. **Standard deviation:** the standard deviation of random returns used to generate daily prices (only when working with randomly generated data).
5. **Seed:** a seed for the pseudorandom numbers used to generate the random returns (only when working with randomly generated data).
6. **Entry day:** the day that one enters into the market in each trading month. The user has no control over this parameter, as it is one of the two optimized arguments. The optimization program considers 22 entry days in every trading month. All 22 options are tried by BODT.
7. **Side:** the position side, either *long*, which is to make profits when stocks are rising, or *short*, which is to make profits when stocks are falling. The optimization program considers only these two options for this parameter. Like the *Entry day*, the user has no control over this parameter, as it is the second argument of optimization.

2.4 Default values for parameters

If the user does not enter a value or enters a value that is outside the permissible ranges (see Section 2.5 for the feasible ranges of the parameters), a default value will be used. The default values are:

- Maximum holding period: 7;
- Stop loss: 10;
- Sample length: 1000;
- Standard deviation: 1.

The reason for these feasible ranges is to place an upper limit to the number of trials (or optimization iterations) conducted. Such a limit does not imply a loss of generality with regards to the analysis. On the contrary, we show that overfitting can deliver significantly high performance (in-sample) even for a relatively small number of iterations.

2.5 Four types of experiments

To study the impact of overfitting, BODT performs four different types of experiments.

- The first experiment replicates the example given on the first page of the online interface associated with two plots depicted on the first page on the online interface. Note that these are the same plots discussed in Figure 1.
- The second experiment uses randomly generated parameters as well as randomly generated sample data (daily closing prices).
- The third experiment asks the user to enter parameters and generates randomly the sample data.
- The fourth experiment asks the user to enter parameters for actual stock market sample data. Note that among these four experiments, only this one benefits from real financial data. In the other three, sample data are randomly generated (more specifically, the optimization program generates a pseudo-random time series of daily closing prices). As mentioned in Section 2.2, for each experiment the generated sample data is equally divided into two sets: the IS and the OOS.

A more detailed explanation of these four experiments is provided below. Since the user does not have any control over the parameters *entry day* and *side*, we shall not discuss them below. All 22 possible values for *entry day* and two values (-1 and +1) for *side* are tried by BODT independent of the chosen experiment.

1. **Experiment 1.** *Re-building the example which is available on the first page of the online interface.* The two plots associated with this example were shown in Figure 1 and are displayed on the first page of the online interface to illustrate the backtesting overfitting concept. Thus, the user can replicate this experiment by calling the pre-defined values for parameters. Note that, the user does not need to enter any value; the online interface sets the values upon calling the experiment 1. In this experiment, the sample data is generated randomly from the Gaussian distribution with the standard deviation and the seed values as given below. The parameters values are:
   (a) Maximum holding period: 20
   (b) Maximum stop loss: 23
   (c) Sample length: 1152
   (d) Standard deviation: 2
   (e) Seed: 308
2. **Experiment 2.** *Generating parameters randomly, from the ranges allowed for each parameter.* In this experiment, the sample data is generated randomly from the Gaussian distribution with the standard deviation and the seed values chosen randomly from the given ranges. The ranges for parameters values are:
   (a) Maximum holding period: an integer in ranges [5, 20];
   (b) Maximum stop loss: an integer in ranges [10, 40];
   (c) Sample length: an integer in ranges [1000, 2000];
   (d) Standard deviation: any positive integer;
   (e) Seed: any positive integer.
3. **Experiment 3.** *User-defined parameter values.* The user may enter any values from the specified ranges for the following five parameters. If any parameter is left blank, that is, the user does not enter any value for it, then a random value is generated from the feasible ranges by the online interface. In this experiment, the sample data is generated randomly from a Gaussian distribution with the standard deviation and the seed values given by the user or chosen randomly by the online interface if left blank.

(a) Maximum holding period: an integer in ranges [5, 20];

(b) Maximum stop loss: an integer in ranges [10, 40];

(c) Sample length: an integer in ranges [1000, 2000];

(d) Standard deviation: any positive integer;

(e) Seed: any positive integer. To set the current time as the seed, enter any non-positive integer.

4. **Experiment 4.** *Using actual stock market data.* Actual daily closing prices are taken from the S&P500 index from January 1962 to February 2014. Our preference for this index is motivated by its wide acceptance as a benchmark and financial instrument. Standard deviation is implied by the data and seed parameter is not relevant in this experiment. Moreover, the user can enter parameters for the *Maximum holding period*, the *Maximum stop loss* and the *Sample length* (any or all of them). The ranges for these parameters are as follow.

   (a) Maximum holding period: an integer in ranges [5, 20];

   (b) Maximum stop loss: an integer in ranges [10, 40];

   (c) Sample length: an integer in ranges [5000, 6000];

   Note that due to the size of the S&P500 index data, the range for the parameter *Sample length* has changed.

2.6 How BODT functions

The operation of BODT has been illustrated in Figure 3. BODT has two modules: the optimization module coded in the programming language Python, and the communication module, which is an online interface providing a bridge between the user and the optimization module. The optimization module performs the following four steps.

1. **Pseudo-random/real-world sample data:** depending on the experiment chosen by the user, this step constructs a pseudo-random time series simulating the daily closing prices of a stock. The daily returns are given by the default pseudo-random number generator *Gauss* within the Python package. This Gaussian distribution for the returns has mean zero and a user-specified standard deviation. The random numbers are controlled by the user through three parameters: the length of the time series (number of days or sample length), the standard deviation, and the seed for the random generator. If the user chooses the real-world stock market experiment, the daily closing prices of the S&P500 index are loaded by the optimization program as the sample data.

2. **Optimal strategy:** develops an "optimal" trading strategy based on the given IS sample data and the parameters, by successively adjusting the following four parameters (it performs a brute-force search by trying all combinations of the four parameters):

   (a) Stop loss: the percentage of investment notional that can be lost before the position is liquidated (exited). BODT only tries integer percentages up till the maximum given by the user.

   (b) Holding period: the number of days that a position can be held before it is liquidated. It is given in a whole number of trading days. BODT tries all integer values less or equal to the maximum given by the user.

   (c) Entry day: the day that one enters the market in each trading month, which is assumed to include 22 business days. All 22 possibilities are tried by BODT.

   (d) Side: the side of the held positions, either 'long' (profits are made when stock prices rise) or 'short' (profits are made when stock prices fall). Both options are tried by BODT.

   Note that among the above four parameters, the user specifies the maximum values for the first two, the maximum stop loss percentage and the maximum number of days to hold on to the stock.

   If a change of these parameters yields a higher Sharpe Ratio (SR) than the previous trials attempted, then a new intermediate result is displayed. The program then continues to try different permutations until it has tried every possibility. Moreover, a Deflated Sharpe Ratio (DSR) statistic

is computed for the IS set to reflect the true impact of the so called best strategy (remember that performance is assessed and optimized in terms of the SR).

An annualized SR statistic is computed by using Equation (1)

$$SR = \frac{\mu}{\sigma}\sqrt{q} \tag{1}$$

where q is the number of returns per year.

A deflated Sharpe Ratio (DSR) performance statistic is computed over IS data; see Equation (2). This is designed to have a more robust performance statistic, in particular, when the returns follow a non-Normal distribution, and multiple trials have taken place. It is obtained by setting threshold $\widehat{SR_0}$ for Probabilistic Sharpe Ratio (PSR) [6], where this threshold increases with the number of trials involved is identifying a strategy. We refer the interested reader to [6, 2] for a more detailed discussion. Precisely we use:

$$\widehat{DSR} \equiv \widehat{PSR}(\widehat{SR_0}) = Z\left[\frac{(\widehat{SR} - \widehat{SR_0})\sqrt{T-1}}{\sqrt{1 - \hat{\gamma_3}\widehat{SR} + \frac{\hat{\gamma_4}-1}{4}\widehat{SR^2}}}\right] \tag{2}$$

where,

$$\widehat{SR_0} = \sqrt{Var[\{\widehat{SR_n}\}]}\left((1-\gamma)Z^{-1}\left[1 - \frac{1}{N}\right] + \gamma Z^{-1}\left[1 - \frac{1}{N}e^{-1}\right]\right). \tag{3}$$

In Equation (3), $Var[\{\widehat{SR_n}\}]$ is the variance over all trials' estimated SR, N is the number of independent trials (the independent combinations tried out), T is the sample length, $\hat{\gamma_3}$ is the skewness of the returns distribution and $\hat{\gamma_4}$ is the kurtosis of the returns distribution, for the selected strategy and Z is the cumulative function of the Standard Normal distribution.

3. **Evaluation of the optimal strategy on the OOS data:** A second pseudo-random simulated stock market time series (out-of-sample, OOS, data) is generated (or real-world stock market data is used depending on the chosen experiment by the user), and the "optimal" strategy generated above is then applied to the OOS data. This time series simply continues from the IS time series, with the same pseudo-random number generator for the same number of simulated days (the two time series are shifted up by the same amount so that the minimum values in both cases are above 0.01). Then the SR statistic is computed.

4. **Visualization:** BODT outputs, on the result page, one movie and three graphs. The first two graphs show results on the IS set, i.e., the backtest (the graph on the left) and the OOS data (the graph on the right). A "movie" showing the progression of the generation of the optimal strategy on the IS data can be played by clicking on the graph on the left.

   In those two graphs, the green line is the underlying time series, and the blue line shows the performance of the strategy. The SR notation in those graphs denotes the standard Sharpe Ratio. In most runs, the SR of the right graph (i.e., the final strategy on the OOS data) is either negative or at the very least much lower than the SR of the final left graph (i.e., the final strategy on the IS data), evidencing that the strategy has been overfit on the IS data.

   The third graph shows the value of the Deflated Sharpe Ratio (DSR) over changes in the value of the *Number of trials*, as a blue line (see Figure 4). The same for a benchmark setting (skewness: -3 and kurtosis: 10) has been shown as a red line, only to give an idea of different behavior given a change in the values of skewness and kurtosis.

   A set of numerical values will be reported by BODT. These include the five parameters, *Maximum Holding Period*, *Maximum Stop Loss*, *Sample Length*, *Standard Deviation*, and *Seed*. In addition to these parameters values, the status of the data (either randomly generated data or real-world stock market data), the SR of the OOS data and the DSR of the IS data are reported in a table similar to Table 1 (the values of Table 1 are only for illustration purposes).

   The execution time of BODT is typically less than two minutes. The values for the maximum holding period, the stop loss and the sample length significantly affect the number of iterations performed by the program; the larger these values are, the longer the program will run.

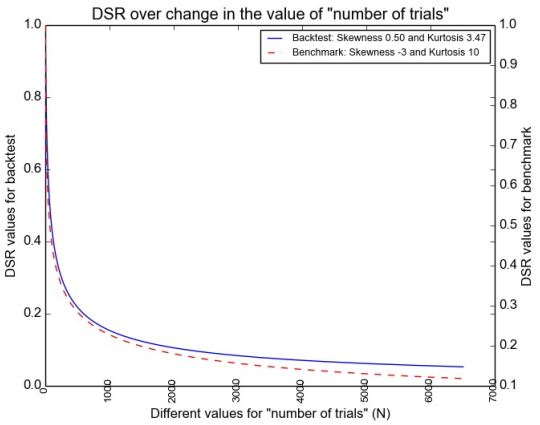


Figure 4: DSR behavior over the change in the value N, the number of trials. Note also how changes in skewness and kurtosis affect DSR.

Parameters for this run	
Maximum Holding Period	20
Maximum Stop Loss	23
Sample Length	1152
Standard Deviation	2
Seed	308
Real-World Stock Market Data Used	No
Sharpe Ratio (SR) of OOS Data	-0.2260
Deflated Sharpe Ratio (DSR) of IS Data	0.3744

Table 1: Illustration of numerical outcomes of BODT

3 The Tenure Maker Simulation Tool

The seasonal strategies are very popular among investors, and are marketed every day in TV shows, business publications and academic journals. Section 2 illustrated how easy is to overfit a backtest involving a seasonal strategy. BODT finds optimal strategies on random (unpredictable) data as well as on real-world stock market data, thus demonstrating that high Sharpe ratios (SR) are meaningless unless investors control for the number of trials.

But what about the rest of strategies? Are strategies based on econometric or statistical methods easy to overfit as well? Unfortunately the answer is that these pseudo-mathematical investments are even easier to overfit. The Tenure Maker Simulation Tool or TMST looks for econometric specifications that maximize the predictive power (in-sample) of a random (unpredictable) time series. The resulting Sharpe ratios tend to be even higher than in the 'seasonal' counterpart.

3.1 The structure of the tool

The core of the *Tenure Maker Simulation Tool* or TMST is an optimization program coded in programming language Python (the optimization module). The optimization program generates pseudorandom data, more precisely, a series of IID (independent, identically distributed) Normal returns (see Section 3.6 for more detail). It then generates a large number of time series models, where the series is forecasted as a fraction of past realizations of that same series, and calculates the Sharpe Ratio (SR) in order to evaluate the generated strategies. Finally, it outputs plots. These four steps are communicated to the user via an online interface (the communication module). TMST is available to the public for free and can be accessed through https://carma.newcastle.edu.au/tenuremaker/.

In particular, the online interface collects and/or sets the parameters values, supplies them to the optimization program, and reports the outcomes from the optimization program. The online interface has two web pages: the first page asks for the parameters or sets them, depending on the type of the experiment chosen by the user, while the second page reports the outcomes of the optimization program.

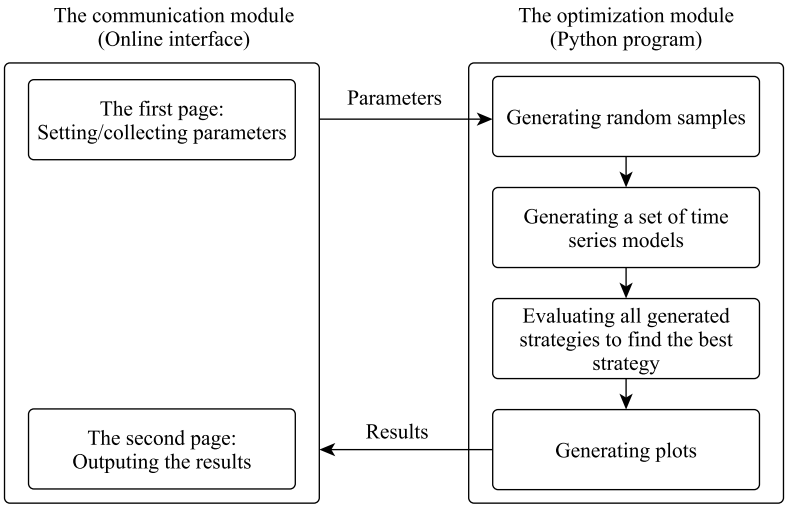


Figure 5: Interaction of the optimization and communication modules of TMST. The optimization module takes parameters from the online interface (the communication module). Upon completion, results are sent to the communication module to be reported to the user.

The outcomes include a movie and two plots. A brief explanation is available on each page of the online interface. Furthermore, a more detailed explanation is documented in a tutorial which can be downloaded from either page of the online interface.

The online interface and the Python optimization program were constructed by Amir Salehipour and Marcos López de Prado. A schematic view of TMST has been depicted in Figure 5.

3.2 Data

TMST works with a pseudorandom data. A series of IID (independent, identically distributed) Normal returns is generated. The sample data is considered the in-sample (IS) set. A set of time series models is automatically generated, where the series is forecasted as a fraction of past realizations of that same series. The forecasted series is considered the out-of-sample (OOS) set. Strategies are generated based on the IS and OOS sets, and are evaluated by using the SR statistic.

3.3 Parameters

TMST has six parameters. Five of these parameters are not available to the user; thus, the user has no control over them; for these parameters, the optimization program sets pre-specified values as discussed below. The six parameters are:

1. **Sample length:** the number of observations (IID returns) generated.
2. **Width:** sample length used as the look-back period in the rolling sum regression models (for more detail on the regression models see Section 3.6).
3. **Polynomial degree:** degrees of the polynomial fit used in the polynomial regression model.
4. **Number of lags:** number of lagged variables included in the lagged regression model.
5. **Number of cross products:** size of the cross product regressors.
6. **Maximum Computational Time:** this is the one parameter that is available to the user and is the total computational time in seconds, that the optimization module is allowed to generate the strategies. It is in the ranges [30, 900] seconds, and the default value is 90 seconds. Only integer values are allowed as the maximum computational time. Hence, if the user does not enter any value for this parameter or if the value is out of the specified ranges, the default value, which is 90, will be used.

3.4 Default values for parameters

The default values for the six parameters of Section 3.3 are:

- Sample length: 1250;
- Width: 3;
- Polynomial degree: 3;
- Number of lags: 1;
- Number of cross products: 3;
- Maximum computational time: 90.

3.5 Two types of experiments

The following two options are available.

1. **Experiment 1: Full.** The program stops when all the strategies are generated, which may take up to 10 minutes.
2. **Experiment 2: Limited.** The user sets the maximum computational time in seconds. As mentioned earlier, the acceptable entries are from 30 to 900 seconds, and the default value is 90 seconds.

3.6 How TMST functions

The operation of TMST has been illustrated in Figure 5. Similar to BODT, TMST has two modules: the optimization module coded in the programming language Python, and the communication module, which is an online interface providing a bridge between the user and the optimization module. The optimization module performs the following four steps.

1. **Return generation:** A series of IID (independent, identically distributed) Normal returns are generated.
2. **Time series model generation:** A large number of time series models are automatically generated, where the series is forecasted as a fraction of past realizations of that same series. The time series models include:
   (a) Rolling sums of the past series: the model is similar to the moving average with the difference that sum, instead of average, is used over the last past series;
   (b) Polynomials of the past series: the model is in the form of $y = a_0 + a_1x + a_2x^2 + ... + a_nx^n + \epsilon$, where, y is the dependent (response) variable, x is the explanatory variable (regressor), a is regression coefficient (a parameter) and ϵ is the error;
   (c) Lag of the past series: In this model, the dependent variable (y) is predicted based on both the current values of an explanatory variable (x) as well as the lagged (past period) values (e.g. $y = a_0 + a_1x_t + a_1x_{t-1} + a_2x_{t-2} + \epsilon$);
   (d) Cross-products of the above.
3. **Strategy evaluation:** A forward-selection algorithm is applied on these alternative specifications in order to select the improved model. The evaluation statistic is SR; see Equation (1).
4. **Visualization:** The program outputs, on the result page, two graphs. The first graph (on the left) shows the "best" strategy (which is similar to Figure 1). The second graph (on the right) shows "inflation" progress in the annualized Sharpe Ratio (aSR). This is illustrated in Figure 6. A "movie" showing the progression of the generating the strategies can be played by clicking on the graph on the left (the user can also download and save the movie).

   In the graph on the left, the green line represents the trading strategy behavior. The blue line represents the market behavior. The purpose of TMST is to show how the green line gets more and more profitable over time as the program continues to optimize the system to fit historical data. In a matter of seconds or minutes, the program creates what appears to be a very profitable equity curve (with a very high SR) based on the input dataset.

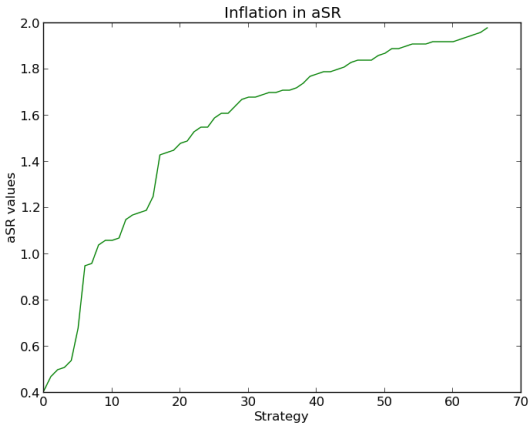


Figure 6: Inflation progress in aSR resulted by generating and selecting strategies. As the figure shows, SR is increasing, as a result of overfitting.

The outcome of TMST is very interesting; we are predicting future realizations of the series by using past realizations, which is of course impossible by construction. As Figure 6 shows, the SR is even more inflated than in the first type (those based on general trading rules). This is one justification that why econometric specifications are so flexible that it is even easier to generate a large number of independent trials.

4 Conclusion

Financial research is increasingly reliant on computational techniques to simulate a large number of alternative investment strategies on a given dataset. One problem with this approach is that the standard Neyman-Pearson hypothesis testing framework was designed for individual experiments. In other words, when multiple trials are attempted, the significance level (i.e., probability of a false positive) is higher than the value set by the researcher.

Academic articles and investment proposals almost never disclose the number of trials involved in a particular discovery. Consequently it is highly likely that many published findings are just statistical flukes. The practical implication is that investors are being lured into allocating capital to irrelevant discoveries, financial theories or investment products.

The *Backtest Overfitting Demonstration Tool* or BODT and the *Tenure Maker Simulation Tool* or TMST are, to our knowledge, the first scientific software to illustrate how overfitting impacts financial investment strategies and decisions in practice. In particular, it shows how the *optimal* strategy identified by backtesting the in-sample data almost always leads to disappointing performance when applied to the out-of-sample data. We also showed how appropriate performance statistics, such as the Deflated Sharpe Ratio, will assist the investors in identifying overfitting, and thus accepting or rejecting such an "optimal" investment strategy.

Our main goal with BODT and TMST is to raise awareness regarding the problem of backtest overfitting in the financial literature and, more generally, to highlight the problem of multiple testing in many other fields of modern science and engineering. In addition to illustrating overfitting, BODT computes the Deflated Sharpe Ratio, an advanced performance statistic to correct for the effect of multiple testing in investment strategy research. We invite the reader to explore the tools and provide feedback:

https://carma.newcastle.edu.au/backtest/ and https://carma.newcastle.edu.au/tenuremaker/.

Acknowledgment

BODT is based on an earlier version which was developed by Stephanie Ger, Alex Sim and John Wu. This is gratefully acknowledged.

References

[1] David H Bailey, Jonathan M Borwein, Marcos López de Prado, and Qiji Jim Zhu. "The probability of backtest overfitting". In: *Journal of Financial Mathematics* (Forthcoming, accepted March 2015).

[2] David H Bailey and Marcos López de Prado. "The Deflated Sharpe Ratio: Correcting for Selection Bias, Backtest Overfitting, and Non-Normality". In: *Journal of Portfolio Management* 40.5 (2014), pp. 94–107.

[3] David H Bailey, Jonathan M Borwein, Marcos López de Prado, and Qiji Jim Zhu. "Pseudomathematics and financial charlatanism: The effects of backtest over fitting on out-of-sample performance". In: *Notices of the AMS* 61.5 (2014), pp. 458–471.

[4] David H Bailey, Stephanie Ger, Marcos López de Prado, Alexander Sim, and Kesheng Wu. "Statistical Overfitting and Backtest Performance". In: *Available at SSRN 2507040* (2014).

[5] Petter N. Kolm, Reha Tütüncü, and Frank J. Fabozzi. "60 Years of portfolio optimization: Practical challenges and current trends". In: *European Journal of Operational Research* 234.2 (2014). 60 years following Harry Markowitzs contribution to portfolio theory and operations research, pp. 356 –371.

[6] David H. Bailey and Marcos López de Prado. "The Sharpe Ratio Efficient Frontier". In: *The Journal of Risk* 15.2 (2012), pp. 3–44.

[7] Andrew W. Lo. "The Statistics of Sharpe Ratios". In: *Financial Analysts Journal* 58.4 (2002), pp. 36–52.

[8] William F Sharpe. "The Sharpe ratio". In: *The Journal of Portfolio Management* 21.1 (1994), pp. 49–58.

[9] William F Sharpe. "Mutual fund performance". In: *Journal of Business* (1966), pp. 119–138.