MASTER'S THESIS

Interuniversity Master in Statistics and Operations Research UPC-UB

Title: Do Stop-Loss rules stop losses? An analytical framework based on modeling overnight gaps and the Stationary Bootstrap

Author: Albert Dorador

Advisor: Argimiro Arratia Quesada

Department: Computer Science

University: Universitat Politècnica de Catalunya

Academic year: 2016-2017

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

UNIVERSITAT DE BARCELONA

Facultat de Matematiques i Estadistica
Universitat Politecnica de Catalunya

Master in Statistics and Operations Research
Master's Degree Thesis

Do Stop-Loss rules stop losses? An analytical framework based on modeling overnight gaps and the Stationary Bootstrap

Albert Dorador

supervised by

Dr. Argimiro Arratia

Department of Computer Science

Abstract

Stop-Loss rules are a risk management tool whose basic idea is simple: the investor predefines some condition that, upon being triggered by market dynamics, imply the liquidation of her outstanding position. Such tool is widely used by practitioners in hopes of improving investment performance by cutting losses and consolidating gains. But, do Stop-Losses really add value to an investment strategy?

This Master's Degree Thesis answers this question using very diverse, rigorous methods, which complement each other and lead to a clear answer. On the one hand, we use a model-based approach in which we present two different models for the price of a New York Stock Exchange asset, which consider the very relevant and traditionally equally disregarded phenomenon of overnight gaps; on the other hand, we complement the previous approach with a data-based framework, in which we describe and implement the Truncated Geometric Stationary Bootstrap, refining nowadays' dominant algorithm in this field.

We find that, in rising markets, Stop-Loss rules improve the expected risk-adjusted return according to most metrics, while improving absolute expected return in falling markets. Furthermore, we find that the fixed percentage Stop-Loss rule may well be the most powerful Stop-Loss rule among the popular rules that we consider in this work.

Keywords: Stop-Loss, Risk Management, Financial modeling, Bootstrap

Contents

Chapter 1

Introduction

After the 2007 subprime crisis, the importance of understanding and implementing effective risk management techniques has become more apparent than ever before.

It is now, in this context, when the field of Finance is becoming increasingly more quantitative and scientific, that one may find a plethora of risk management tools. But there is one that still stands out, as it embodies the very essence of financial risk management: Stop-Loss rules.

A Stop-Loss order, henceforward SL order (or simply SL), is an order an investor may place so that her position is liquidated the moment a certain prespecified condition is met by market dynamics. This definition includes also the case in which the investor holds a short position i.e. she has entered the position by selling an asset which she must eventually buy to close the position (hoping to buy at a lower price than she sold and thereby making a profit). Whether the position the investor holds is short or long, the purpose of setting a Stop-Loss is, as the name suggests, to *stop losses*, because the investor fears that if she doesn't liquidate her position at that time, losses may end up being catastrophic.

Similarly, we will also consider Stop-Profit orders (SP), which determine a condition that if triggered the investor liquidates a winning position fearing that her current profit might vanish otherwise.

Our interest in Stop-Loss and Stop-Profit orders (sometimes referred in conjunction as just "Stop-Loss" orders in this document) lies in the fact that although they are widely used instruments in practice, there is a surprisingly scarce academic literature on this topic.

Nonetheless, our study finds its roots in the works of Acar and Toffel [1] (2001), James and Yang [15] (2010), as well as Kaminski and Lo [18] (2014). The first reference studies how stop-losses affect the returns distribution, by assuming the asset follows a Brownian Random Walk with drift, and then evaluating the financial profitability of a simple Stop-Loss strategy under the previous assumption. Kaminski and Lo follow a similar approach: they develop a rigorous analytical framework for measuring the impact of simple 0/1 stop-loss-re-entry rules on the expected return and volatility of an arbitrary portfolio strategy (again assuming that assets follow a Random Walk) and provide an empirical analysis of a stop-loss policy applied to a buy-and-hold strategy in U.S. equities. James and Yang, on the other hand, do not assume

that financial assets follow a particular model, but instead base their analysis on the use of the Stationary Bootstrap as a tool to replicate financial time series adequately.

Our work finds great inspiration in the above references and combines the model-based approach of Acar & Toffel and Kaminiski & Lo, with the data-based approach presented in James & Yang.

This work diverges with respect to the cited papers in several aspects, though. We may highlight the fact that we use higher-frequency financial data (hourly prices), as well as more complex models that take into account overnight gaps, in an attempt to mimic the behavior of real financial assets better, at the expense of having to base our conclusions in computational simulation techniques and results drawn from Probability Theory (above all, the Law of Large Numbers). Moreover, we define and implement a slightly different Stationary Bootstrap algorithm than the one James and Yang use (which is the most common version, based on the work of the fathers of the Stationary Bootstrap, Politis and Romano [27] (1994)), which deals in a more mathematically accurate manner with the fact that the random length of the bootstrapped blocks cannot be larger than the length of the time series to be bootstrapped.

Our ultimate goal is to prove the conjecture that good Stop-Loss strategies may provide higher risk-adjusted returns, with respect to a Buy-and-Hold strategy.

In the process of proving this conjecture, this study makes several contributions to the existing literature on Stop-Loss policies and beyond. As main contributions, we may highlight:

- Two new performance metrics: R-VaR and R-ES ratios (Section 3.2)
- Two financial price models that consider overnight gaps (Chapter 4)
- "Range rule of thumb" for the Generalized Normal Distribution (Section 4.1)
- Truncated Geometric Stationary Bootstrap algorithm (Section 5.3)
- Stop-Loss policy analysis using high-frequency financial data

We will focus on long-only strategies, and will not take into consideration slippage (difference between order price and execution price) or transaction costs. As a matter of fact, transaction costs are irrelevant to our study, because the transaction costs associated with a Stop-Loss strategy are exactly the same as those associated with a Buy-and-Hold strategy, for two reasons: first, because we only consider pure Stop-Loss policies, i.e. we disregard re-entry rules; second, because placing a Stop-Loss order is free, regardless of which broker the investor operates with.

This document is organized as follows. Chapter 2 provides a survey of the main Stop-Loss / Stop-Profit strategies employed by practitioners. Chapter 3 formalizes the notion of a Stop-Loss rule, and provides some of the most technical machinery we will need in the rest of the document. In Chapter 4 we describe in detail our two parametric models to simulate asset prices: one without a momentum component (positive serial correlation), and another that allows for momentum (ARMA). Chapter 5 discusses Bootstrap methods in time series, and presents our algorithm that implements the Truncated Geometric Stationary Bootstrap. Emphasis must be placed on the fact that our algorithm refines the original algorithm by Politis and Romano [27]

by truncating the support of the Geometric distribution used. This algorithm is implemented in R, thereby offering a better solution than the current algorithm (function `boot` from the R package **boot** [7]), which does not use the truncated geometric distribution. In Chapter 6 we show the results of using a Stop-Loss rule, both from a parametric approach (model-based) and a non-parametric approach (data-based). In all cases, we consider four Stop-Loss rules (outlined in Chapter 2) and we measure performance according to 5 criteria (described in Chapter 3). Chapter 7 summarizes the main conclusions in this study, and points at possible lines of research that would be a natural continuation of the present work.

Chapter 2

Survey of common Stop-Loss strategies in practice

There are many kinds of Stop-Loss / Stop-Profit rules, and especially so if one considers this concept in a broad sense.

Moreover, it is important to emphasize that having a Stop-Loss / Stop-Profit policy in place does not force one to permanently keep the barriers at the levels at which one originally placed them until the price touches one of them and the position is liquidated, but instead one may dynamically update the barriers to "follow" the current price. In fact, accompanying the price seems like a more reasonable strategy. Such a kind of stop is commonly referred to as a "trailing stop".

In this chapter, we present the strategies in increasing order of sophistication, approximately.

2.1 Fixed-percentage barrier

The most basic strategy consists on setting the barriers based on a prespecified monetary loss or profit. A variant of this policy consists on setting the barriers according to a prespecified percentage loss or profit relative to the current price or to another reference price e.g. the maximum price reached so far.

In the first case, the barriers are established simply as follows:

$$SL_t \coloneqq P_{t-1} - m$$

where $m > 0$ is the maximum monetary value per share that the investor is willing to lose in that operation. And

$$SP_t \coloneqq P_{t-1} + m$$

where $m > 0$ is the monetary value per share that the investor would be realistically satisfied to win in that operation.

On the other hand, in the case of a fixed percentage, the barriers are established as:

$$SL_t \coloneqq P_{t-1}(1-a)$$

where $a > 0$ is the maximum percentage of the current price the investor is willing to lose in that operation. And

$$SP_t \coloneqq P_{t-1}(1+b)$$

where $b > 0$ is the percentage of the current price the investor would be realistically satisfied to win in that operation.

In this work we implement a very simple variation of this strategy that is well-known both in practice and in academia (see for example Zambelli [42]), which takes the maximum price up until time $t-1$ as a reference, instead of the current price. More formally, in our code we implement the following Stop-Loss rule:

$$SL_t \coloneqq P_{t-1}^{max}(1-a)$$

where $a > 0$ is the maximum percentage of the highest price achieved until time $t-1$ that the investor is willing to lose in that operation. And

$$SP_t \coloneqq P_{t-1}^{max}(1+b)$$

where $b > 0$ is the percentage of the highest price achieved until time $t-1$ that the investor would be realistically satisfied to win in that operation.

2.2 The support/resistance method

This method can be stated very simply: set your Stop-Loss barrier just below the last support level for the asset, and set your Stop-Profit barrier just above the last resistance level for the asset. Of course, sometimes determining the support and resistance levels may not be easy, and there is almost an inevitable amount of subjectivity involved in that judgment. The two most common ways of deciding the support and resistance levels are either by seeking recent "floors" and "ceilings" in the price chart of the asset, or by considering a Moving Average, for instance, the popular 50-period one. We may define our barriers as

$$SL_t \coloneqq Supp_{t-1} - \epsilon \quad \text{for } \epsilon \in \mathbb{R}$$

and

$$SP_t \coloneqq Res_{t-1} + \epsilon \quad \text{for } \epsilon \in \mathbb{R}$$

where $Supp_{t-1}$ and Res_{t-1} are predefined Support/Resistance levels according to some criterion, and ϵ is a small predefined tolerance, in the form of a price or percentage differential. Setting a

small tolerance reduces the probability of triggering the Stop-Loss or Stop-Profit too soon, while staying close to the hypothetically correct barrier. A common range for ϵ is between 0.7% and 1% below or above the support or resistance level, respectively.

2.3 Average True Range (ATR)

A concept related to the ATR, the True Range (TR), is a measure of volatility that is defined as

$$TR_t \coloneqq max\{High_t - Low_t\,,\;\; High_t - Close_{t-1}\,,\;\; Close_{t-1} - Low_t\}$$

In plain words, TR is the difference between today's high and today's low (intra-day difference), *unless* one of those bounds makes the difference smaller than if you considered yesterday's closing price.

Then, we define

$$ATR_t \triangleq \frac{1}{N}\sum_{i=0}^{N} TR_{t-i}$$

A typical value for N is 14. Note that the ATR is in the currency of the asset (not in relative terms).

The Stop-Loss is then set at level:

$$SL_t \coloneqq P_{t-1} - \alpha ATR_t$$

where α is a real number usually picked between 1.5 and 3. The practitioner's rationale behind this range of values is (probably) that she sees ATR as a metric akin to the standard deviation of the price in the recent past, and so α is the number of ATR's below the current price.

A symmetrical argument would justify setting the Stop-Profit barrier as

$$SP_t \coloneqq P_{t-1} + \beta ATR_t$$

The general idea behind this method is that if volatility, as measured by the ATR, is high, one should set wider profit-taking barriers around the current price, so as not to liquidate the position unnecessarily in one of the multiple and pronounced ups-and-downs that a high volatility level anticipates.

This method is thus similar to the so-called Bollinger Bands, which are two "bands" placed at 2 standard deviations below and above the 20-period moving average. Although is not uncommon to base one's Stop-Loss / Stop-Profit barriers on the Bollinger Bands, the ATR seems to be much more popular for such purpose, and for this reason we focus on the ATR.

2.4 Indicator Stop

In this case the investor looks for a quantifiable sign of weakness to liquidate her position (assuming she holds a long position). It is not a Stop-Loss order in the canonical sense, but it is definitely a stopping mechanism, and serves exactly the same purpose. The two most popular metrics used are a combination of Moving Averages (MAs), and the Relative Strength Index (RSI).

2.4.1 Moving Averages (MAs)

A popular exit signal is the triple crossover of short, medium and long-term MAs. Practitioners use many different MA periods, and one that seems reasonable would be the triplet 5-20-70. Assuming the starting scenario is that the shorter-period MAs are above the longer-period ones, the exit signal is then given by the cumulative event of the 5-period MA crossing the 20 and the 70-period MAs, together with the 20-period MA crossing the 70-period MA. If those three events happen, there is a high probability of a continued fall in price, according to Technical Analysis. This method's rationale is that longer-term MAs show more of an asset's historical price trend, whereas shorter-term MAs show more the asset's recent price trend, and so if the recent trend crosses the historical trend, we may be witnessing a change of regime (for the worse if the crossover happens in the above-mentioned manner). A MA of p periods is defined as

$$MA(p) \triangleq \frac{1}{p}\sum_{i=0}^{p-1} C_{t-i}$$

where C_t is the asset's closing price at session t.

2.4.2 Relative Strength Index (RSI)

The RSI is a momentum indicator that compares the magnitude of an asset's recent gains and losses over a specified time period. It is primarily used to attempt to identify overbought or oversold conditions in the trading of an asset. It is defined as

$$RSI_t \triangleq 100 - \frac{100}{1 + RS(w)_t} \in [0, 100]$$

where $RS(w)_t$ is the average gain of up periods during the specified time frame, divided by the average loss of down periods during the specified time frame, that is, for a window of w periods (usually, 14), containing u up and $w - u = d$ down periods,

$$RS(w)_t \coloneqq \frac{\frac{1}{u}\sum_{i\in U}(P_i - P_{i-1})}{\frac{1}{w-u}\sum_{i\in D}(P_i - P_{i-1})}$$

RSI values of 70 or above indicate that a security is becoming overbought or overvalued, and therefore may be primed for a trend reversal or corrective pullback in price. The rationale for this strategy may lie in the belief that too much euphoria can in fact anticipate a change of regime. This behavior is in fact what one observes in major market corrections [34].

Chapter 3

Stop-Loss rule, Performance metrics and Distributions

In this chapter we establish the general framework for Stop-Loss analysis. Some of the assumptions and definitions are similar to the ones laid out in Kaminski and Lo [18]. We also describe two non-trivial probability distributions that will play a central role in our analysis: the Weibull distribution, and the Generalized Gaussian Distribution (also known as Generalized Normal Distribution, or Generalized Error Distribution).

3.1 Stop-Loss rule: formalization

Consider an arbitrary portfolio strategy P with returns $\{r_t\}$, which satisfies the following assumption:

Assumption The expected return of P is greater than the risk-free rate r_f, and let $\pi \coloneqq \mu - r_f$ denote the risk premium of P.

This assumption simply excludes the perverse case where the Stop-Loss policy adds value just because the risk-free asset that the investor transfers the capital to has a higher expected return than P. This is certainly a necessary assumption, yet a very weak one, because both by theoretical reasons (e.g. the Efficient Market Hypothesis or basic Utility Theory) and by empirical evidence, a risky asset with a lower expected return than a risk-free asset is unlikely to exist (and in any case it would certainly be an exception, not a rule, which is what we intend to uncover in this study).

In our code, we guarantee this assumption is satisfied by setting the risk-free rate and the expected average asset return to appropriate values.

We set the risk-free rate to the historical mean return of a 1-year US Treasury bill, which, during the period 1990-2017 has been 3.171%.

On the other hand, the expected return of our hypothetical asset is set to the average return of the NYSE during the same period (1990-2017), which is 6.14%. Assuming compounding, this implies an hourly return of $3.942924 * 10^{-5}$, just considering trading hours.

Definition 1 A simple **Stop-Loss (or Stop-Profit) rule** $\mathbf{s}(\vec{x_t}, \vec{\gamma})$ for a portfolio strategy P with returns $\{r_t\}$ is a dynamic binary asset-allocation rule $\{s_t\}$ between P and a risk-free asset F (e.g. cash, a government bond, or a bank deposit) with return r_f, where s_t is the proportion of assets allocated to P, and:

$$s_t = f(\vec{x_t}, \vec{\gamma}) = \begin{cases} 1, & \text{stay in} \\ 0, & \text{exit} \end{cases} \tag{3.1}$$

where $\vec{x_t}$ is a vector of known information up until time t, and $\vec{\gamma}$ is a vector of parameters that guide the Stop-Loss policy. In the case of the Stop-Profit, the definition is identical and hence omitted, as Stop-Loss and Stop-Profit are fundamentally the same type of rule.

Definition 1 describes a 0/1 asset-allocation rule between P and the risk-free asset F, where 100% of the assets are withdrawn from P and invested in F as soon as the stopping rule, as a function of $\vec{x}$ and $\vec{\gamma}$, is triggered. Some examples of $\vec{x}$ are the current price, the average price over a certain time window, the maximum price so far, or the volatility over a predefined time interval. On the other hand, some examples of $\vec{\gamma}$ are the critical percentage above and/or below the maximum price so far, or the time window of interest regarding volatility or the asset price average(s).

Naturally, we make the simplifying assumption that the interest rate is constant throughout the year, and so the investor can switch between the risky asset and the safe asset at any time and earn the proportional risk-free return based on how much of the year still remains when she leaves the risky asset. In reality, the return that the risk-free asset provides depends on the time the investor contracts the risk-free instrument, but here we just assume this rate is constant and equal to the historical (1990-2017) mean return of a 1-year US T-bill.

In order to assess the impact of the Stop-Loss policy **s** on performance, in next section we present the four metrics that we will use.

3.2 Performance metrics

Definition 2 The **Sharpe Ratio** SR of a Stop-Loss policy, as defined by Sharpe in 1994 [33], is the expected return in excess of the risk-free rate divided by the standard deviation of that risky return, for a given time horizon (usually, 1 year):

$$SR \triangleq \frac{\mathbb{E}[r_{St}] - r_f}{sd(r_{St})} \tag{3.2}$$

In our code, we use the Method of Moments approximation to the expected return and the standard deviation of returns. Because the number of simulations we perform is large (in the order of millions), the Law of Large Numbers suggests that these empirical approximations are good.

If the numerator of the Sharpe Ratio happens to be negative, a smaller risk would produce a worse performance metric (a more negative one), which might be counterintuitive. However, this is not really a problem, for two reasons: first, because both upside and downside risk is considered, a smaller risk when the expected return is below the risk-free return might as well be regarded as something undesirable because it means that more of the density is accumulated around that expected value, i.e. the probability of achieving a return near the expected return is higher, which is not good as it is a very poor return; second, a more general reason is that any risky strategy with an expected return below the risk-free rate should in general be automatically disregarded (unless the investor is risk-loving and it then might make sense for her from a utility-theoretical point of view). As the vast majority of investors are not risk-lovers (in the formal sense that her investment Risk Premium is negative), we can simply handle the case of a negative numerator in the aforementioned way.

Although the Sharpe Ratio is a popular metric in finance to compare performance, it has some downsides. For example, due the fact that the Sharpe Ratio uses the standard deviation as a measure of risk, this metric treats downside risk the same as upside risk, which in general is not desirable.

In order to address this issue and penalize only for downside risk, we present another relatively well-known alternative, and we also propose two new metrics.

Definition 3 The **Sortino Ratio** SOR of a Stop-Loss policy, as defined by Sortino in 1994 [35], is the expected return in excess of the minimum return rate acceptable (e.g. the risk-free rate) T divided by the so-called Downside Risk (DR), for a given time horizon (usually 1 year).

$$SOR \triangleq \frac{\mathbb{E}[r_{St}] - T}{DR} \tag{3.3}$$

The Downside Risk, despite the fancy name, is a measure that is conceptually similar to the standard deviation of the returns (as a random variable), restricting the support to values below target, but it's not exactly a standard deviation, because the reference value is not an expected return, but a target return. Mathematically,

$$DR \coloneqq \sqrt{\int_{-\infty}^{T} (r_{St} - T)^2 f(r_{St}) dr_{St}}$$

where $f(r_{St})$ is the density function of the returns in the Stop-Loss / Stop-Profit strategy.

As already pointed out, although the DR is often mistakenly thought of as the standard deviation of the below-target returns, it is not so (for the reason above), and in fact it is something potentially better: by considering the deviations with respect to T (and not with respect to the

expected below-target return) we guard ourselves from pathological cases like having most of the below-target returns clustered around the expected below-target return, inducing a very small standard deviation if that was the reference value, which would fool the investor into thinking that there is small downside risk when clearly that's not the case (at least according to generally accepted measures of downside risk like the Value at Risk or the Expected Shortfall), while by considering T as the reference value, you capture downside risk better because only in cases where truly there's little downside risk you would obtain a small DR value.

In our code, we use function `SortinoRatio` from package **PerformanceAnalytics** [26], which provides an efficient computation using the `apply` base R function.

Next, we propose two new performance metrics.

Definition 4 The **Return-VaR ratio** $RVaR$ of a Stop-Loss policy is the expected return (in excess of the risk-free rate) divided by the median return minus the 1-year 5% VaR:

$$RVaR \triangleq \frac{median(r_{St}) - r_f}{median(r_{St}) - VaR_{5\%}} \tag{3.4}$$

where $VaR_{5\%} = CDF^{-1}_{r_{St}}(5\%)$, CDF being the cumulative distribution function of the returns generated by the Stop-Loss / Stop-Profit strategy. Perhaps, since we look at the returns at risk, a more appropriate name would be "Return-RaR" or "RRaR", but we decide to keep the more common term VaR even if we define it slightly differently here.

We consider the empirical 1-year 5% VaR, that is, the return that is at the 5th percentile of the ranked observed returns (i.e. based on the Empirical CDF), after millions of simulations (for reasons analogous to the ones above regarding the Law of Large Numbers).

We choose the median return instead of the mean return (or, rather, the expected return), for two reasons: first, because the median is a metric that is more robust to outliers than the mean; second, because the ratio as a whole becomes easier to interpret if both the numerator and denominator have the same metric. The ratio can then be interpreted as follows: the numerator informs about how larger is the median return of the risky investment or strategy compared to the return of a risk-free asset, while the denominator takes some of those merits off by penalizing for downside risk.

One may argue that the numerator now is harder to interpret, as the expected return is a concept that most investors are perfectly acquainted with and may be easier to interpret. However, by using the median return we provide a different perspective, which has a very powerful interpretation: the median return is the smallest return the investment or strategy "promises" to deliver 50% of the time. Hence, by using the median return, the numerator of our ratio is telling one the minimum amount by which the return of the risky investment surpasses that of the risk-free asset 50% of the time.

However, there's arguably one shortcoming of using the median instead of the mean return: from a utility theoretic perspective, the difference in expected values $\mathbb{E}(r_S t) - rf$ is more meaningful as it can be related to the risk premium the investor expects from the risky investment.

Hence, by using the median, we lose one interpretation (albeit a purely theoretical one) but we gain another, more practical one: indeed, it matters little that a risky investment has a higher expected return than the risk-free rate if higher returns than the risk-free rate happen just 1% of the time; in an extreme case, the investor might go broke before that attractive yet unlikely high returns happen.

The function applied to the denominator, the identity function, has been carefully chosen. Power functions with powers below and above 1 have been considered.

On the one hand, a power below 1 (but above 0) e.g. a square or cube root, may be interesting because it would diminish the attractiveness of Stop-Loss / Stop-Profit strategies yielding "overly safe" annual returns having in the 5th percentile a return smaller than 1 in magnitude, which almost surely implies a small expected return. More specifically, recall that a number between 0 and 1 raised to a power between 0 and 1 is mapped to a larger number, but still smaller than 1 (ensuring the function is strictly increasing for values of the domain between 0 and 1). Moreover, considering the fact that by definition a $VaR_{5\%}$ is a measure that focuses on an event that statistically happens only 5% of the time, perhaps a performance measure should not give equal weights to this measure of risk and the median excess return.

On the other hand, a power above 1 may capture the fact that investors tend to penalize losses super-linearly e.g. quadratically, so it may make sense that a strategy yielding a VaR that is twice as large in magnitude as another one may be regarded more than twice as bad.

Both arguments are sound, and this may advise choosing a power exactly equal to 1. But we have one more argument that supports the choice of a unitary exponent: a $VaR_{c\%}$ is sometimes computed as an affine function of the standard deviation of the portfolio's value, in particular, of the form $VaR_{c\%} = \mathbb{E}(P_h) - k\sigma$, where k is chosen so that $CDF_{P_h}(k\sigma) = c\%$, P_h being the value of the portfolio with horizon h. Therefore, a change in the standard deviation affects the VaR linearly.

By subtracting the $VaR_{5\%}$ from the median return, by the definition of the terms involved, we guarantee that the denominator is positive, which makes the RVaR metric a meaningful one under all possible scenarios, even pathological ones like the extremely unlikely case that $\mathbb{E}(r) < VaR_{5\%} < median(r)$, which still can be handled, following the second argument presented above in the case of a negative Sharpe Ratio: any risky strategy with an expected return below the risk-free rate should in general be automatically disregarded. Furthermore, the denominator can never be 0, by definition, or at least, such an scenario has probability 0, in a measure-theoretical sense (which in purity is not synonymous with "impossible").

Finally, note that the numerator and the denominator are in the same units: percentage returns. This fact, which also happens in the Sharpe Ratio, adds to the soundness of the RVaR ratio. We obtain, therefore, a unitless ratio.

Definition 5 The **Return-ES ratio** RES of a Stop-Loss policy is the median return, in excess of the risk-free rate, divided by the median return minus the 1-year 5% Expected Shortfall:

$$RES \triangleq \frac{median(r_{St}) - r_f}{median(r_{St}) - ES_{5\%}} \tag{3.5}$$

where $ES_{5\%} := \frac{1}{5\%} \int_0^{5\%} VaR_\gamma \, d\gamma$, which effectively computes the average VaR_γ with γ ranging from 0 to 5%. An intuitive justification of this formula and our interpretation as the average VaR in that range is that the expectation of any random variable, in this case, the VaR, is computed as the product of all the values it may take and its probability density function (PDF). In this case, we impose that the VaR's PDF is that of a uniform distribution, which enables us to give equal weight to all the (infinitely-many) VaRs in the range 0-5%.

We highlight the fact that the Expected Shortfall is a "coherent risk measure" in the sense of Artzner [3], even though it is very closely related to the VaR, which is not a coherent risk measure, as it does not respect the sub-additivity property. An immediate consequence is that Value at Risk might discourage diversification. Furthermore, after the 2007 subprime crisis, the Expected Shortfall has been chosen to replace the Value-at-Risk in the Basel III regulatory framework, as it is arguably a more cautious metric.

Nonetheless, it is true that since the portfolio we consider is composed of just 1 stock, there is no practical difference between a coherent risk measure and a risk measure that violates the sub-additivity property, in our particular case.

Just as in the Return-VaR ratio, here we consider the same time-percentile parameters, i.e. the empirical 1-year 5% Expected Shortfall, that is, the mean return within the bottom 5% observed returns, after millions of simulations.

For reasons analogous to those in the previous performance metric, we consider the denominator above. Note that the argument that the VaR can be seen as an affine function of the standard deviation of the portfolio's value is still valid for the Expected Shortfall, as the $ES_{c\%}$ can be understood as an average of all the possible $VaR_{\gamma\%}$, for $0 < \gamma < c$.

All the other considerations explained in the case of the previous metric apply analogously to this one, so we will refrain from stating them again.

Comparing our two new metrics with the Sharpe and Sortino ratios

The two performance metrics we propose are in principle better than the Sharpe Ratio, as we specifically penalize just for the downside risk of the strategy. Moreover, those two metrics provide a different perspective compared to the Sharpe and Sortino ratios (which essentially is an improved version of the Sharpe ratio).

Besides providing a completely different point of view compared to the Sortino ratio, our two metrics have two more advantages over the Sortino ratio.

First, the choice of the parameter involved in the metrics (the percentile used as reference) is much less arbitrary than the choice of the Minimum Acceptable Return in the Sortino Ratio. The reason is that the choice of 5% for the Value-at-Risk and the Expected Shortfall is very standard, and there is little debate on this point (besides considering the other two common values of 2.5% and 1%, although far less popular); however, it is largely arbitrary what value to give to the minimum acceptable return: many (myself included) may advocate to use the risk-free rate, for simplicity, but it may make sense to require a larger return, because after all, the risky investment should have a higher return than a risk-free asset. How much higher? This is arbitrary.

Second, on the technical side: when evaluating a risky investment, the denominator of the two metrics we propose is non-zero (in fact, positive) with probability 1; this is not true about the Sortino ratio, which may yield a zero denominator if the strategy does not yield any return below the minimum acceptable return, in the period considered (and of course depending on the minimum acceptable return specified).

3.3 Probability distributions

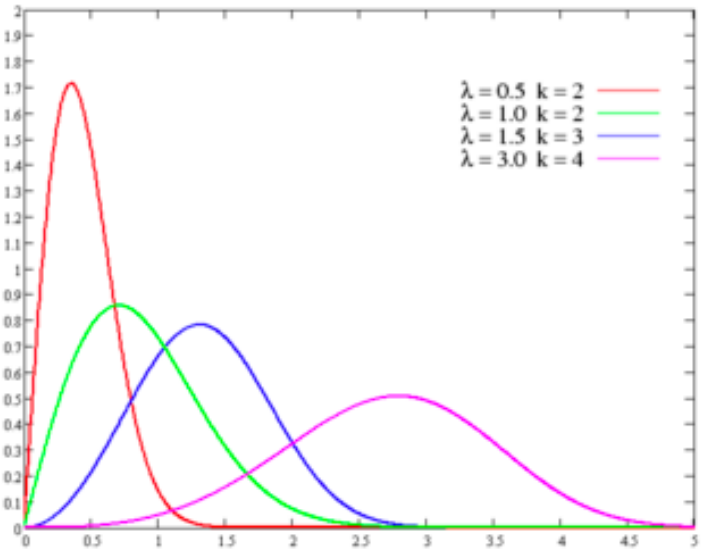
Definition 6 The **Weibull distribution** is a continuous probability distribution named after the Swedish mathematician Waloddi Weibull, who described it in detail in 1951, although it was first identified by Frechet (1927) and first applied by Rosin & Rammler (1933) to describe a particle size distribution. Indeed, it constitutes another example of Stigler's law of eponymy, according to which no scientific discovery is named after its original discoverer.

Although there are several possible parametrizations (at least three that are relatively widely used) we consider the probably most common 2-parameter specification. Hence, the probability density function (PDF) of a Weibull random variable is

$$f(x;\lambda,\kappa) = \begin{cases} \frac{\kappa}{\lambda}\left(\frac{x}{\lambda}\right)^{\kappa-1} e^{-(x/\lambda)^{\kappa}}, & x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

where $\lambda > 0$ is the scale parameter and $\kappa > 0$ is the shape parameter of the distribution. The Weibull distribution is related to a number of other probability distributions; in particular, it interpolates between the exponential distribution ($\kappa = 1$) and the Rayleigh distribution ($\kappa = 2$ and $\lambda = \sqrt{2}\sigma$, σ being the scale parameter of the Rayleigh).

The above PDF is graphed below, considering different values for the scale and shape parameters.



We can see that the main features and moments of the distribution - e.g. mean, median, variance, skewness, kurtosis - change completely when the parameters are altered. More formally, the PDF of the Weibull distribution changes drastically with the value of κ. For $0 < \kappa < 1$, the density function tends to ∞ as x approaches zero from above and is strictly decreasing. For $\kappa = 1$, the density function tends to $\frac{1}{\lambda}$ as x approaches zero from above and is strictly decreasing. For $\kappa > 1$, the density function tends to zero as x approaches zero from above, increases until its mode and decreases after it. This PDF has infinite negative slope at x = 0 if $0 < \kappa < 1$, infinite positive slope at x = 0 if $1 < \kappa < 2$, a finite positive slope if $\kappa = 2$, and null slope at x = 0 for $\kappa > 2$. As a matter of fact, as $\kappa \to \infty$, the Weibull converges in Distribution to the Dirac Delta function, thereby placing all the density on a single point of its support, the point $x = \lambda$.

Also in relation to the shape parameter κ, we present a remarkable fact that will be used in Section 4.1.

Proposition 1. *A sufficient condition to increase the downside risk of a Weibull distribution is to reduce its shape parameter κ.*

Proof

In the case that the downside risk is measured by the Expected Shortfall, Chen and Gerlach [8] showed that, for a *two-sided Weibull**,

$$ES_\alpha := \frac{-\lambda_1^2}{\alpha b_p \kappa_1} \Gamma\left(1 + \frac{1}{\kappa_1}, \left(\frac{-b_p VaR_\alpha}{\lambda_1}\right)^{\kappa_1}\right);$$

$$0 \leq \alpha \leq \frac{\lambda}{\kappa_1}$$

where $\Gamma(s, x) := \int_x^\infty t^{s-1} e^{-t}\, dt$, i.e. the upper incomplete Gamma function

Therefore, it is straightforward to see that if we decrease κ,

i the fraction premultiplying the upper incomplete Gamma function increases

ii in $\Gamma(s, x)$, s increases, and so the power function has a larger area under its curve, i.e. the definite integral increases, ceteris paribus

iii in $\Gamma(s, x)$, x decreases, and so the lower limit of integration decreases, and hence the definite integral increases, ceteris paribus

Therefore, the combined effect is that the ES_α increases in magnitude if we reduce κ.

* The PDF of a Standardized Two-sided Weibull (STW) is given by

$$f(x; \lambda_1, \kappa_1, \kappa_2) = \begin{cases} b_p \left(\frac{-b_p x}{\lambda_1}\right)^{\kappa_1 - 1} e^{-(-b_p x/\lambda_1)^{\kappa_1}}, & x < 0 \\ b_p \left(\frac{b_p x}{\lambda_2}\right)^{\kappa_2 - 1} e^{-(b_p x/\lambda_2)^{\kappa_2}}, & x \geq 0 \end{cases} \tag{3.7}$$

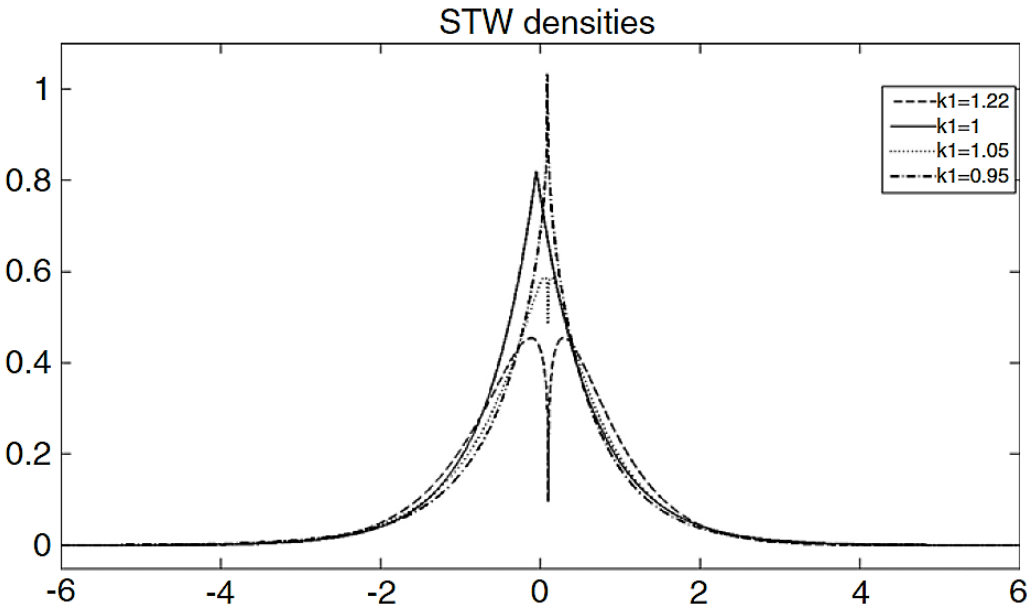


Figure 3.1: Density of a Standardized Two-sided Weibull (STW)

And it's graphed in Figure 3.1.

We conclude the proof by justifying the use of a result concerning a two-sided Weibull distribution rather than a plain (one-sided) Weibull distribution, which is the distribution we use: the Expected Shortfall only concerns the "left" Weibull in the context of a two-sided Weibull distribution, i.e. the Weibull with negative support, because the Expected Shortfall, by definition, deals with losses. In our case, as it will be discussed further later, we use the Weibull distribution to model the overnight gaps, and for us a "loss" happens when the gap is in the open interval defined by 0 and 1. Therefore, in our case we just consider the "right" i.e. one-sided Weibull, whose PDF is clearly just that of the "left" Weibull mirrored across the vertical axis by taking the negative of x. By doing this, the negative signs in the definition of ES_α vanish and the Expected Shortfall clearly increases (not just in magnitude) when κ is decreased, ceteris paribus. Hence, reducing κ increases the downside risk of a Weibull distribution, as measured by the Expected Shortfall. □

Numerical Evidence

For a fixed scale of 1 and a fixed percentile of 50, we observe numerically that the first derivative of the Weibull CDF with respect to κ ranges from 0 to 1.476541, i.e. is non-negative for all the equally spaced 10^8 values of κ in $[10^{-6}, 10^6] \cup 10^{12}$. This is highly suggestive of the fact that reducing κ reduces the value of the overnight gap needed to accumulate 50% of the density. In other words, the left tail of the distribution is composed of smaller gaps below 1 (i.e. closer to the minimum gap possible, 0), which clearly implies an increased downside risk, by any account (not just Expected Shortfall).

The aforementioned numerical result is shown graphically on Figure 3.2.

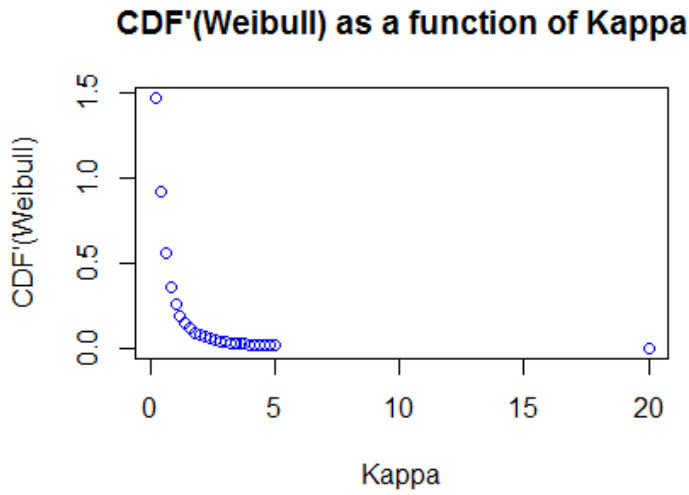


Figure 3.2: 1st derivative of the CDF of a Weibull, as a function of κ

Definition 7 The **Generalized Gaussian distribution** is a continuous probability distribution that adds a shape parameter β to the Gaussian distribution. The fact that there are two "versions" of this distribution may cause confusion, and for that reason it is crucial that we clarify which one we mean. Here, when we talk about the Generalized Gaussian distribution, or Generalized Error distribution (GED) we refer to a parametric family of symmetric distributions which include the Laplace distribution ($\beta = 1$), Normal distribution ($\beta = 2$), and continuous uniform distribution ($\beta \to \infty$). As such, this family allows for tails that are either heavier than Normal (when $\beta < 2$) or lighter than Normal (when $\beta > 2$).

The probability density function (PDF) of the Generalized Error Distribution is given by

$$f(\mu, \sigma, \beta) = \frac{\beta}{2\sigma\Gamma(1/\beta)} e^{-(|x-\mu|/\sigma)^{\beta}}$$

where μ, σ, β are the location, scale, and shape parameters respectively, and Γ denotes the Gamma function. As Nadarajah [22] showed, the PDF of a Generalized Normal distribution is just the same as the PDF of a Normal distribution but with a general β coefficient that is not necessarily equal to 2 (and hence the absolute value function is needed), as well as a different regularization term premultiplying the exponential function, so that the PDF integrates to unity.

The PDF defined above is graphed in Figure 3.3, considering different values for the shape parameter.

Nadarajah proved that the expected value and the variance of a random variable X that follows this distribution are

$$\mathbb{E}(X) = \mu$$

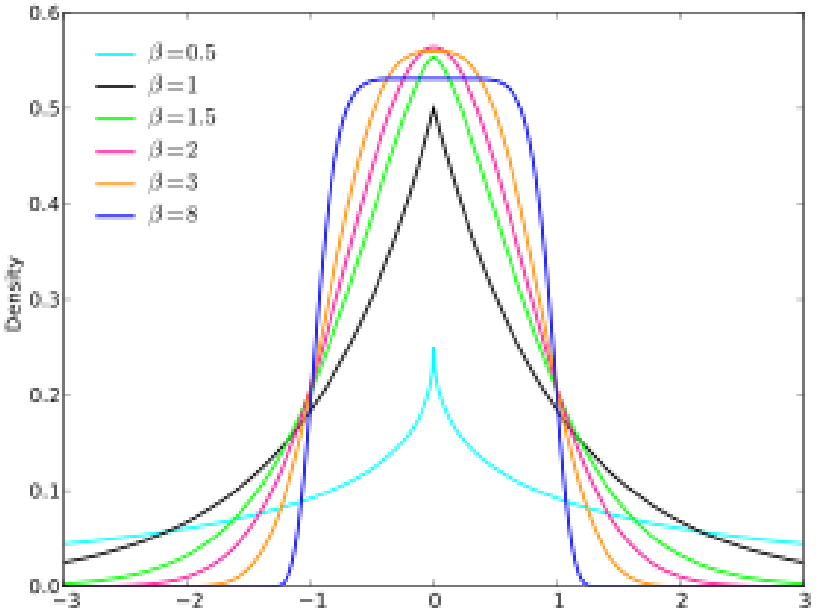


Figure 3.3: Probability Density Function for a GED, with different shape parameters

$$Var(X) = \frac{\sigma^2 \Gamma(3/s)}{\Gamma(1/s)}$$

Note that in the Generalized Normal σ is not the standard deviation, but just a scale parameter that is related to the standard deviation. We will use the expression for the variance in a proof in section 4.1, in the context of our Range heuristic.

But before we continue, we must shed some light on the meaning of a key concept that we are about to use and it is often misunderstood: kurtosis. The standard measure of kurtosis, as conceived by Karl Pearson, is based on a scaled version of the fourth moment of the data. As mathematician Peter Westfall [39] proves, this quantity is related to the tails of the distribution, not its peak. Westfall builds on the findings of Johnson, Tietjen, and Beckman (1980) [16], who provided examples of distributions with identical kurtosis as the Normal, with widely different peaks. Hence, the frequent characterization of kurtosis as "peakedness" is mistaken: higher kurtosis is the result of infrequent extreme deviations (or outliers), and this has almost nothing to do with the shape of the distribution's peak.

As Moors [21] showed, the kurtosis coefficient of a random variable X can be expressed simply as

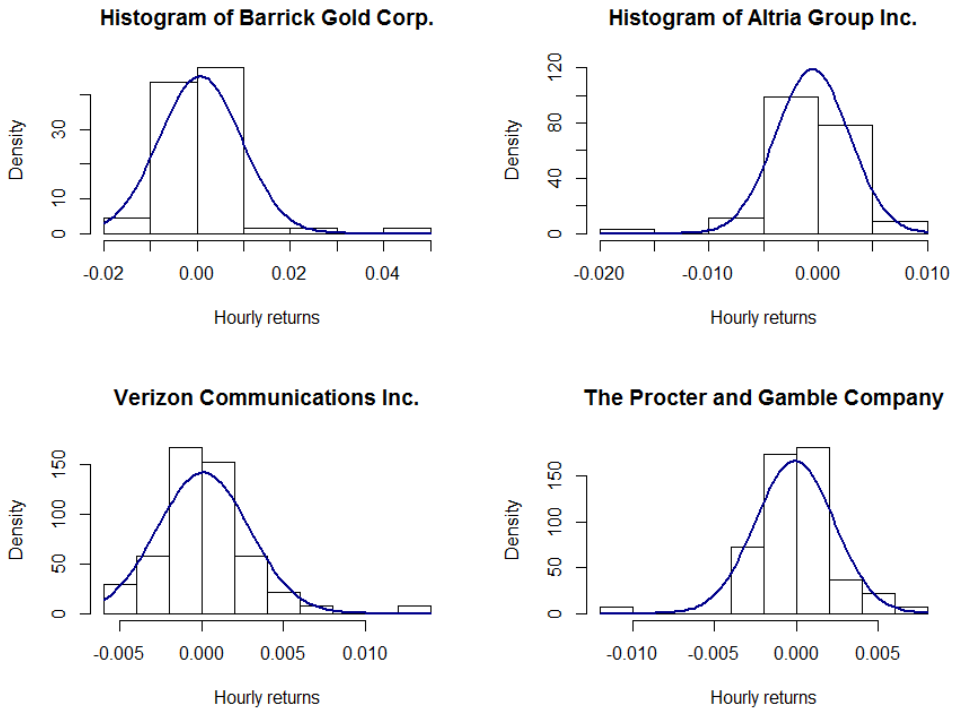
$$Kurt(X) = \mathbb{E}\left(\left(\frac{x-\mu}{\sigma}\right)^4\right)$$

Where μ is the expected value of X and σ its standard deviation.

Looking at the mathematical definition, it is clear why this concept has to do with outliers and not "peakedness": kurtosis is the expected value of the standardized data raised to the fourth power. Any standardized values that are less than 1 (i.e. data within one standard deviation of the mean, where the "peak" would be), contribute virtually nothing to kurtosis, and so most of the contribution to kurtosis comes from the data values outside the region of the peak. What's more, this contribution is quartic, so kurtosis is very sensitive to outliers, and for that reason it is a measure almost exclusively about the frequency and magnitude of outliers.

Continuing with our discussion about the Generalized Error Distribution, this distribution provides a useful way to parametrize a continuum of symmetric, platykurtic densities ($kurtosis < 3$) spanning from the Normal to the continuous uniform, and a continuum of symmetric, leptokurtic densities ($kurtosis > 3$) spanning from the Laplace to the Normal.

As we can see by looking at the densities graphed in Figure 3.3, by adjusting β we can easily change the kurtosis of the distribution, which will be useful when modeling the fat tails that stock returns typically show. Indeed, we have observed such behavior in several of the stocks considered, and below we show the histograms of the four stocks with highest kurtosis in the period considered (February and March 2017), which are, in order, Barrick Gold Corp., Altria Group Inc., Verizon Communications Inc., and The Procter and Gamble Company.



We see that the empirical distribution of the hourly returns above do have a high kurtosis, as characterized by the presence of outliers.

Lastly, it is worth pointing out that a Student-t distribution with a small number of degrees of

freedom (e.g. 4) would also allow fat tails, but in a slightly different way: unlike the Generalized Error Distribution, the Student-t obtains heavier-than-normal tails without acquiring a cusp at the origin. Nonetheless, we find that the Generalized Error Distribution provides a better fit for our empirical hourly returns than a Student-t, and for that reason that's the distribution we will use in our first parametric model (see next chapter).

Chapter 4

Parametric models: Modeling asset prices

We present two models: the first one without a momentum component, and the second one, of type ARMA, which by definition allows the possibility of momentum.

The modeling process consists of three parts:

1. Modeling the overnight gap
2. Modeling the standard deviation of returns, including white noise
3. Modeling the actual return

Since the first two parts are common in the two models we explore, we present them here first, and then we will focus on the third part separately for each of the two models.

Our analysis is based on the 30 NYSE stocks that were most traded (i.e. the ones with highest transaction volume) in 2013, which is the most recent year for which officially released NYSE data is publicly available at this moment. For these simulations, we consider hourly quotes from March 13 to March 24 (both included), for each of the 30 stocks, which equates to 2070 hourly returns.

Hourly price quotes allow for more realistic simulations, since the asset price as well as the Stop-Loss / Stop-Profit barriers are generated and checked each hour (respectively), instead of each day or even month (the usual time-frames employed in many similar simulations). Hence, in order to obtain a tractable yet more realistic simulation, we consider hourly price quotes (instead of the more common, yet less realistic, daily/monthly quotes). These high frequency financial data have been downloaded using VBA excel in conjunction with a free API provided by Google Finance, as it's not directly accessible for free on the web.

4.1 Modeling overnight gaps

The hypothesis that stock market prices follow a Random Walk is very well-known in finance, dating back to Louis Bachelier's PhD Thesis in 1900 [4].

However, although Bachelier is arguably the father of quantitative finance, the Random Walk hypothesis has been severely criticized on the grounds of abundant empirical evidence. For this reason, we depart from a pure Random Walk stochastic model, and present our own model.

One of the most important novelties we present in this study is the introduction of overnight gaps in the determination and assessment of Stop-Loss policies, which is something that has a tremendous impact and has not been explored before, to the best of our knowledge.

An overnight gap is defined as the difference between the Open price at period t, and the Close price at period $t-1$. It is possible (and indeed quite frequent) that the stock is traded for the first time at 9:30 a.m. EST (Open price) at a *different* price than the last traded price the day before at 4:00 p.m. EST (Close price), typically because some event (e.g. earnings announcement, court decision, or, simply, rumors) might have changed the investors' perception about the fair value of a given asset.

Sell and buy orders are first ranked based on price and then based on chronological placement order, and they are executed whenever a match exists: the buyer offers a Bid price that is larger than or equal to the Ask price the seller intends to sell the stock for. Therefore, an event that is deemed relevant for the asset's fair value might trigger a certain reaction from investors that translates into buy and sell orders that might be placed immediately when the event happens, e.g. during the Extended Trading Hours (After Hours: 4:00 p.m. to 8:00 p.m. EST and Pre-Market: 8:00 a.m. to 9:30 a.m. EST), or even later in the early morning, but all of them may not be executed until liquidity is higher the next morning after the bell ring signals the opening of the Regular Trading Hours at 9:30 a.m. EST. Then, more investors are operating and the buy-sell pressures can be fully absorbed by the market and translated into a first transaction whose price differs, perhaps substantially, from yesterday's Close price.

But, should we consider yesterday's "raw" Close price, or the so-called "Adjusted Close price"? We follow the common practice of using the Adjusted Close price (which adjusts for stock splits, dividends and right offerings), which is automatically displayed by most financial data providers like Yahoo Finance, but it naturally requires considering the Adjusted Open price as well, which is not given, but can be easily computed as

$$adj.open_t = \frac{adj.close_t}{close_t} * open_t.$$

In any case, it should be clear why incorporating overnight gaps into the model is a major improvement over simpler models: whatever the price of the stock does outside the Regular Trading Hours does not affect one's Stop-Loss (or Stop-Profit) order, and one's order will only be taken into consideration after the bell rings again next morning at 9:30 a.m. EST... by which time the stock price may open below one's Stop-Loss barrier and one's Stop-Loss order will automatically become a market order, which means that it will be executed at the first available price the market offers, i.e. around the price the stock opened at 9:30 a.m. In short,

overnight gaps can severely affect the profitability of an investing strategy that uses Stop-Loss orders, because they render the Stop-Loss barrier completely unable to protect the investor from a loss that is substantially larger than expected. In particular, this means that disregarding the phenomenon of overnight gaps may significantly inflate the usefulness of a Stop-Loss rule, and this is a mistake we avoid.

Using our empirical data on 204,120 gaps (27 years' worth of data on 30 NYSE stocks), we find that the average gap is about 1.000463 (i.e. +0.0463%). In effect, we measure overnight gaps as a non-negative real number, usually around 1, which multiplies the last Close price of a stock. Furthermore, on average, gaps of any size above a minimum threshold of 0.1% happen around 77.93% of the days, and this is the frequency we will use in our simulations.

Furthermore, it is reasonable to believe that the stock's volatility in a given day is a function of the magnitude of the gap (among other things, of course), and so we attempt to model this relationship in order to obtain an even more realistic model. If we didn't do this, we would calibrate volatility based on the data we have, but it would be a kind of "average volatility", without drawing any distinction between days with no gaps and days with a 5% gap, for example. That would be clearly unsatisfactory.

In order to accomplish this, we randomly chose some of the stocks and fit a polynomial in the gap magnitude to the *relative range* (a proxy for volatility to be made precise below). We observed that high order terms of the variable "gap magnitude" were significantly different from 0 in a statistical sense, but we decided to keep the polynomial order low (order 3) for three reasons.

First, because we would like to keep the risk of overfitting low. In that regard, on top of keeping the polynomial order low, we take care of outliers: after fitting the model, we employed function `offliers` from our own R package **analytics** [12] (which we have developed and published on CRAN motivated by this Thesis), specifying that at most 0.1% of the observations are to be identified as outliers according to two different criteria (Cook's Distance and DFBetas), and removed prior to re-fitting the model once again.

Second, because statistical significance does not necessarily mean practical significance. Indeed, in several cases the magnitude of the estimated coefficient of the quartic term is of the order of 10^{-6}, and so we find it wise to keep the order of the model at 3. For the record, the magnitude of the estimated coefficient of the cubic term is in general much larger, of the order of 10^{-3} on average.

Third, because we are going to use the same polynomial model for all 30 stocks, parsimony is important in order to reduce the probability of including non-statistically significant terms for some of the stocks.

Therefore, as already advanced, in order to model the impact of the gap magnitude on the relative range we put forward a cubic polynomial of the form

$$RelRange = \beta_0 + \beta_1 gm + \beta_2 gm^2 + \beta_3 gm^3 + u \tag{4.1}$$

where gm is the gap magnitude, and $RelRange$ is defined as $\frac{Range}{Low} = \frac{High - Low}{Low}$ i.e. the

relative range of the stock in a given day.

In our case, measuring volatility through the relative range is more convenient than through the standard deviation of returns, which would be the most natural choice. The reason is that with no higher frequency data than hourly quotes, we have no idea how wildly the stock moved, especially in the first minutes of the day, and this is why standard deviation would not allow us to gain an appropriate insight.

So, we have seen that the relative range might be a convenient measure of volatility, but is it a good one? Sure. Considering that it is generally accepted that standard deviation is a good measure of volatility (by no means the only measure, of course), we claim that the relative range is a good proxy for standard deviation. In fact, in some introductory statistics textbooks [36] one can find the so-called "Range rule of thumb", which, for a sample of size n drawn from a random variable X:

$$\sigma = \frac{Range(X_n)}{4}$$

This heuristic is arguably quite naive as it is unlikely that such a simple functional form is able to work under a wide array of cases (e.g. different distributions of X, and/or different sample sizes).

Ramirez and Cox [32] provide an improved heuristic for several distributions and sample sizes. They claim that standard deviation is approximately proportional to the range, but the proportionality constant may not necessarily always be 1/4. And so they claim that

$$\sigma \approx \frac{Range(X_n)}{\zeta}$$

where ζ is the constant of proportionality, which depends on sample size and distribution. In case the data come from a Normal distribution, Ramirez and Cox show that the following functional form for ζ provides a good approximation for large sample sizes

$$\zeta = 3\sqrt{\ln n} - 1.5$$

n being the sample size.

However, as already pointed out, our empirical data seems to follow a Generalized Normal distribution, not a "plain" Normal distribution. For that reason, we wanted to obtain an improved functional form for ζ, by following the steps:

Step 1 : Define a function that performs a Monte Carlo simulation for each sample size. In our case, our function, `zeta.GED` , includes the option of running the computation in parallel, which is automatically activated when the sample size is large enough.

Step 2 : Plot in a graph the natural logarithm of the sample size (due to the huge difference in scale) against the average ζ found in Step 1.

Step 3 : Deduce a general functional form that would make the graph in Step 2 a straight line, and use numerical methods to calibrate the involved parameter(s). Here we use function `explus` from our own package **powerplus** [11], as it's a safer choice when the exponentiation operation is somewhat sophisticated.

Step 4 : Fit the affine function that minimizes the sum of the squared errors (Least Squares method).

Following the above steps, we find a closed-form expression for ζ that seems to fit our data better than the one Ramirez and Cox found for the Normal distribution. The way we proceed is by generating 10^6 random samples from the Generalized Normal Distribution. When sample size is above $3{*}10^3$, our function automatically switches the parallelization option we provide to `TRUE` for higher computational performance (for smaller sample size, the overhead makes parallelization *slower* than sequential computing).

As it will be discussed later in Section 4.3, we find that the shape parameter of the Generalized Normal distribution that our empirical hourly returns follows is approximately 1.3, and this is the only parameter we need to fix.

Proposition 2. *Let* $X \sim GED(\mu_x, \sigma_x, \beta)$*, and* $Y \sim GED(\mu_y, \sigma_y, \beta)$ $\forall \mu_x, \mu_y, \sigma_x, \sigma_y, \beta \in \mathbb{R}$ *such that* $Y = aX + b$ $\forall a, b \in \mathbb{R}$*. Then,*

$$\zeta_Y = \zeta_X$$

Proof

First of all, we must state and prove the following lemma.

Lemma 1 (Linear Transformation Theorem)**.** *Let* $Y = aX + b$*. Then*

$$X \sim GED(\mu_x, \sigma_x^2, \beta) \Rightarrow Y \sim GED(a\mu_x + b, a^2\sigma_x^2, \beta)$$

Proof

As shown in Nadarajah and Pogany [23], if $X \sim GED(\mu, \sigma^2, \beta)$ then its characteristic function is given by

$$\varphi_x(t) = \frac{\sqrt{\pi} e^{it\mu}}{\Gamma(1/\beta)} * {}_1\Psi_1 \left[\begin{matrix} (1/\beta, 2/\beta) \\ (1/2, 1) \end{matrix} ; -\frac{\sigma^2 t^2}{4} \right] \tag{4.2}$$

where i is the imaginary unit, and ${}_1\Psi_1$ is the Fox-Wright generalized confluent hypergeometric function ${}_p\Psi_q$, which requires $\beta > 1$ in order to converge. As in our case $\beta \approx 1.3$, we can be sure the rest of the proof will remain applicable.

Let $Y = aX + b$. Then, by the definition of characteristic function,

$$\varphi_y(t) \coloneqq \mathbb{E}\left(e^{itY}\right) = \mathbb{E}\left(e^{it(aX+b)}\right) = \mathbb{E}\left(e^{itaX+itb}\right) = e^{itb}\mathbb{E}\left(e^{itaX}\right) = e^{itb}\varphi_x(at)$$

which equals

$$e^{itb} * \frac{\sqrt{\pi} e^{iat\mu}}{\Gamma(1/\beta)} * {}_1\Psi_1 \left[\begin{matrix} (1/\beta, 2/\beta) \\ (1/2, 1) \end{matrix} ; -\frac{\sigma^2 a^2 t^2}{4} \right] = \frac{\sqrt{\pi} e^{it(a\mu+b)}}{\Gamma(1/\beta)} * {}_1\Psi_1 \left[\begin{matrix} (1/\beta, 2/\beta) \\ (1/2, 1) \end{matrix} ; -\frac{\sigma^2 a^2 t^2}{4} \right]$$

$\therefore$ looking at Equation 4.2, by the Uniqueness Theorem (which establishes that a probability measure on $\mathbb{R}$ is uniquely determined by its characteristic function), we conclude that

$$Y \sim GED(a\mu_x + b, a^2\sigma_x^2, \beta)$$

□

Having proved that Y, like X, follows a Generalized Normal distribution, we proceed with the rest of the main proof.

On the one hand,

$$Var(Y) = Var(aX + b) = a^2 Var(X) = \frac{a^2\sigma_x^2\Gamma(3/\beta)}{\Gamma(1/\beta)} \Rightarrow \sigma_y^2 = a^2\sigma_x^2$$

as β is the same for both random variables by assumption, and hence

$$sd(Y) = |a|sd(X) \Rightarrow \sigma_y = |a|\sigma_x$$

On the other hand,

Let X_n be a sequence of length n generated from the distribution of X. Then, by definition, $Range(X_n) = X_{(n)} - X_{(1)}$, where $X_{(i)}$ is the standard notation for the i-th element of the sequence X_n when sorted in increasing order. Then

$$Range(Y_n) = Range(aX+b) = (aX+b)_{(n)} - (aX+b)_{(1)} = |a|X_{(n)} + b - (|a|X_{(1)} + b) = |a|Range(X_n)$$

Therefore,

$$\zeta_Y = \frac{Range(Y_n)}{sd(Y)} = \frac{|a|Range(X_n)}{|a|sd(X)} = \frac{Range(X_n)}{sd(X)} = \zeta_X$$

$\forall \mu_x, \mu_y, \sigma_x, \sigma_y, \beta \in \mathbb{R}$ such that $Y = aX + b \; \forall a, b \in \mathbb{R}$. □

Therefore, we have proved that the ζ we find is valid for *any* $GED(\mu, \sigma, 1.3)$.

In particular, we have used $\mu = 0$, and $\sigma = 1$, and then checked that it provides the exact same results as using the values for μ and σ that we later find our hourly returns adhere to.

We can now present the following result

Empirical Range heuristic: For samples of size n from a random variable X obeying a $GED(\mu, \sigma, 1.3)$,

$$\sigma \approx \frac{Range(X_n)}{1.714(\ln n)^{0.816} - 0.217}$$

The above expression provides an adjusted R^2 of 0.9999675. Such accuracy is apparent in Figure 4.1, where we just show the fit for values of n up to 500, since higher values would make the shape of the curve for smaller values difficult to visualize properly.

Table 4.1 summarizes our empirical findings, and we compare our model with that of Ramirez-Cox, as well as the naive rule of thumb. In all cases, we employ 1 million trials in our model.

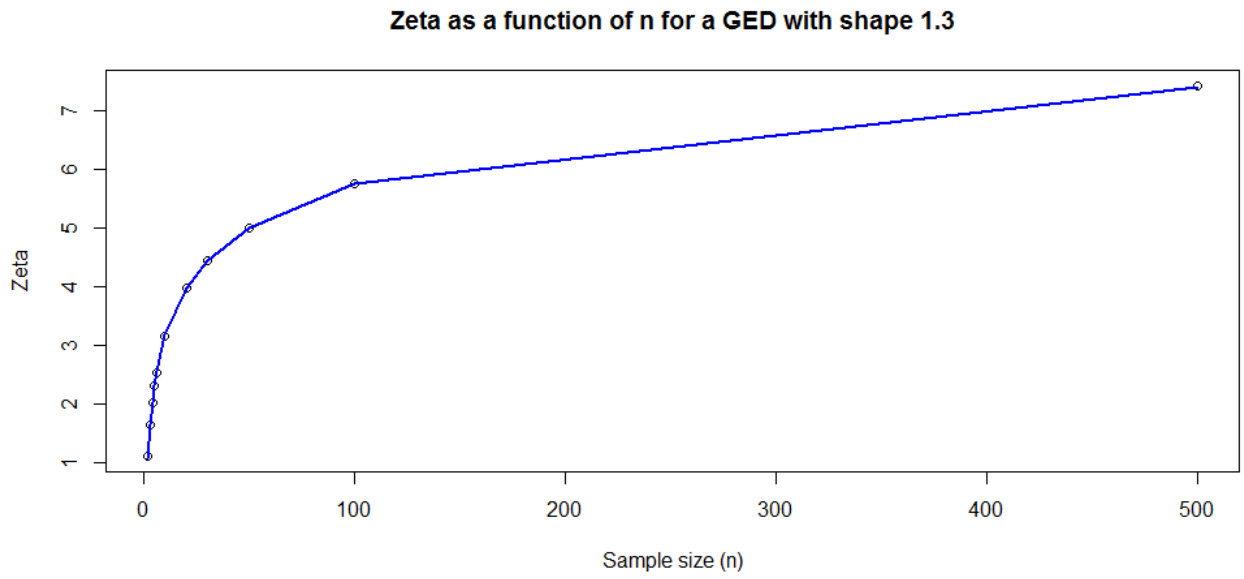


Figure 4.1: Modeling Zeta for a GED with shape parameter of 1.3

Table 4.1: Finding a better ζ for a $GED(\mu, \sigma, 1.3)$: empirical results

n	Q1	**mean**	Q3	**Our model**	Ramirez-Cox	Naive
2	0.408	**1.095**	1.557	**1.054**	0.9977	4
3	0.919	**1.645**	2.193	**1.634**	1.644	4
4	1.279	**2.013**	2.588	**2.021**	2.032	4
5	1.555	**2.293**	2.88	**2.310**	2.306	4
6	1.783	**2.522**	3.117	**2.542**	2.516	4
10	2.411	**3.142**	3.746	**3.168**	3.052	4
20	3.247	**3.962**	4.560	**3.979**	3.693	4
30	3.724	**4.428**	5.017	**4.437**	4.033	4
50	4.315	**5.005**	5.583	**5.000**	4.434	4
⋮	⋮	⋮	⋮	⋮	⋮	⋮
30000	10.720	**11.290**	11.750	**11.286**	8.132	4
50000	11.180	**11.740**	12.200	**11.749**	8.368	4
100000	11.790	**12.350**	12.800	**12.370**	8.679	4

Looking at Table 4.1, we see that the naive rule of thumb does relatively well for sample sizes around 20, and increasingly poorly as sample size deviates from that value. Furthermore, notice that the heuristic by Ramirez and Cox, although devised for the "plain" Normal distribution (i.e. $\beta = 2$), does a good job for a Generalized Normal with $\beta = 1.3$ for sample sizes between 2 and 50, providing an even slightly better average fit for some particular values in that interval than our own heuristic. As sample size increases beyond 50, Ramirez-Cox's heuristic is no longer valid for the case of a Generalized Error Distribution with shape parameter 1.3.

As a matter of fact, in Appendix A we provide an extension of our heuristic for other values of the shape parameter β. This way, our rule can be applied reasonably well for a random variable following virtually any Generalized Normal distribution.

By having shown the functional relationship that, on average, exists between the range and the standard deviation for the case of $X \sim GED(\mu, \sigma, \beta)$, we have, in effect, shown that there exists indeed a tight relationship between both concepts (on average).

Therefore, we deduce that, approximately

$$\sigma = \frac{Range(X_n)}{\zeta} = \frac{Range(X_n)}{min(X_n) * \omega} = \frac{RelRange(X_n)}{\omega}$$

where $RelRange(X_n) \coloneqq \frac{Range(X_n)}{min(X_n)}$

and more importantly, we approximately have that,

$$RelRange(X_n) * \delta \Rightarrow \sigma * \delta$$

Therefore, if we can know by how much the relative range is changed when the gap magnitude changes, then we can approximately know how the standard deviation of that day would change as a function of the magnitude of the gap in that day.

For that reason, we fit the model previously shown in Equation 4.1 and obtain the estimates of the β parameters. Once we have these estimates for each stock, we regularize them by dividing them by the mean relative range of that stock in the period we have observed. The reason for this regularization is that, if we want to estimate the average impact of the gap magnitude for the 30 stocks in our universe (so that we can construct a general model), it is meaningless to say that for one stock an increase of x percentage points in the magnitude of the gap multiplies the relative range by y, since without knowing the average size of the relative gap we have no idea if y is small or big for that stock, and more importantly, we cannot simply average those effects, as they are in different scales.

This way, after regularizing the β coefficients obtained for each stock, we can compute the average coefficients for the 30 stocks. We will use these coefficients to modulate the standard deviation we generate in our simulations, according to the magnitude of the gap observed in that period.

At this point, we need to make a conjecture regarding the distribution that the gap follows. We find that the Weibull distribution, a very flexible distribution with non-negative support as we have shown in Section 3.3, may fit very well the gaps observed if we calibrate the parameters of scale (λ) and shape (κ) appropriately.

Next we will present our way of estimating the Weibull parameters. Because modeling the overnight gap plays a central role in this study, we lay out a very careful procedure, based on 3 steps:

Step 1: Explore 7 estimation methods to obtain a first approximation. In every case, we make a heavy use of the Bootstrap methodology[1], for two reasons: first and foremost, because this way we can have an idea about the main moments of the parameter distribution, and even compute confidence intervals; second, in order to obtain a slightly more robust estimation. Due to the highly computationally intensive nature of this task, we have parallelized our code, allowing us to use all the cores of the machine but one (for safety reasons).

Step 2: Select the best estimation method and perform a grid search within an interval around the estimated parameters, employing a very fine discretization.

Step 3: Perform a final correction for Survivor Bias: the 30 stocks we have picked are the ones that have "survived" from 1990 to 2017, and so it is advisable to be more conservative and consider slightly more downside risk, which is achieved through choosing a smaller shape parameter, as we proved in Section 3.3.

In order to assess how well each estimation method calibrates the Weibull parameters, we consider the Kolmogorov-Smirnov D statistic, as well as the Mean Squared Error (MSE), which we define next:

$$D_n = \sup_x \{|ECDF_n(x) - CDF(x)|\}$$

where $ECDF_n(x)$ is the empirical cumulative distribution function of the n data observed (in our case, gaps) and the $CDF(x)$ is the theoretical cumulative distribution function we have estimated (in our case, a Weibull distribution with parameters κ and λ). In plain words, the D statistic records the largest difference, in absolute value, between the observed cumulative distribution function, and the hypothesized one. Thus, it might be a fairly sensitive statistic.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\widehat{ECDF}(x_i) - ECDF(x_i))^2$$

where $\widehat{ECDF}(x)$ is the predicted empirical cumulative distribution function based on the estimated parameters. Note: effectively, it is the theoretical CDF, just as in the D statistic, but historically the MSE has been conceived to check how well a "prediction" fits the observations.

Because both metrics are in the same 0 to 1 scale, and are similar in nature but the MSE is more robust to outliers (and so we give it more weight), we propose a metric that is a convex combination of the two and that we call Stress (inspired in a somewhat similar concept drawn from the non-metric Multi-Dimensional Scaling literature), and we define as

$$Stress = 0.3D + 0.7MSE$$

Therefore, Stress will be the criterion we will use to compare the performance of the 7 estimation methods.

[1] A thorough treatment of the Bootstrap method will be provided in Chapter 5

Step 1

Method of Moments (MOM)

We employ one million Bootstrap resamples, and use the expressions presented in Nwobi and Ugomma [24] that allow one to easily estimate the shape parameter κ and scale parameter λ. With just a very minor modification for greater computational efficiency, the first expression is as follows:

$$1 + \left(\frac{s}{m}\right)^2 = \frac{\Gamma\left(1 + \frac{2}{\kappa}\right)}{\Gamma\left(1 + \frac{1}{\kappa}\right)^2}$$

Where s and m are the empirical standard deviation and mean, respectively.

Therefore, we can employ numerical methods to find the parameter κ that satisfies the equation. In other words, we find the κ that makes the difference between the two sides equal to 0.

Once we have found κ, we use a closed-form expression to find the scale parameter λ:

$$\lambda = \frac{m}{\Gamma\left(1 + \frac{1}{\kappa}\right)}$$

Maximum Likelihood Estimation (MLE)

Nwobi and Ugomma [24] worked out the log-likelihood expression and the corresponding First Order Conditions (one for each parameter to be optimized), being able to provide a closed-form solution for the scale parameter, and a relatively simple expression for the shape parameter, which can be solved using a numerical algorithm. The expressions are

$$\lambda = \frac{1}{n}\sum_{i=1}^{n} x_i^{\kappa}$$

$$\frac{1}{\kappa} + \frac{1}{n}\sum_{i=1}^{n} \ln x_i - \frac{\sum_{i=1}^{n} x_i^{\kappa} \ln x_i}{\sum_{i=1}^{n} x_i^{\kappa}} = 0$$

We also employ one million Bootstrap resamples.

Median and Quartiles (MQ)

This method is presented in Justus et al. [17], which provides two closed-form expressions to find the shape and scale parameters:

$$\kappa = \frac{\ln\left(\frac{\ln 0.25}{\ln 0.75}\right)}{\ln\left(\frac{ECDF_x^{-1}(0.75)}{ECDF_x^{-1}(0.25)}\right)}$$

$$\lambda = \frac{ECDF_x^{-1}(0.5)}{(\ln 2)^{\frac{1}{\kappa}}}$$

where $ECDF_x^{-1}(a)$ is the value that leaves behind the $(100 * a)$-th percentile of the empirical cumulative distribution function of the data.

We employ one million Bootstrap resamples.

Mean and Standard Deviation (MSD)

Justus et al. [17] present also this method, and provide two closed-form expressions to find the shape and scale parameters:

$$\kappa = \left(\frac{s}{m}\right)^{-1.086}$$

$$\lambda = \frac{m}{\Gamma\left(1 + \frac{1}{\kappa}\right)}$$

where s and m are the sample standard deviation and mean, respectively.

We employ one million Bootstrap resamples too.

The following three methods are drawn from Nwobi and Ugomma [24], and the authors refer to them as "graphical procedures". All three are very similar and only differ in the way their cumulative distribution function (CDF) is estimated. Hence, we present here first the derivation that is common to the three methods, and then highlight each particular way of estimating the CDF of the data.

We generate one million Bootstrap resamples to increase robustness. For each resample, we rank the observations from smallest to largest, estimate the CDF following the appropriate rule in each of the three cases, and then use this estimated CDF to compute the following quantity:

$$Y = \ln\left(\ln\left(\frac{1}{1 - \widehat{CDF}}\right)\right)$$

Then we compute the quantity

$$x = \ln x_{Boot}$$

Finally, we fit the following linear model in order to estimate the Weibull parameters

$$Y = c + \kappa x + u$$

where $\lambda = exp\left(-\frac{c}{\kappa}\right)$

As anticipated earlier, there are three ways of estimating the CDF, each one giving a name to the whole algorithm:

Mean Rank (MR)

$$\widehat{CDF} = \frac{\{i\}}{n+1}$$

Median Rank (MDR)

$$\widehat{CDF} = \frac{\{i - 0.3\}}{n + 0.4}$$

Symmetric CDF (SCDF)

$$\widehat{CDF} = \frac{\{i - 0.5\}}{n}$$

where $\{i\}$ is the sequence of indexes of the observed gaps.

The results and performance of each of the 7 methods are presented in Table 4.2.

Table 4.2: Results and performance after Step 1

Method	**Shape**	**Scale**	D	MSE	**Stress**
MOM	**98.565**	**1.006**	0.194	0.013	**0.068**
MLE	**20.224**	**1.174**	0.918	0.291	**0.479**
MQ	**203.615**	**1.002**	0.105	0.003	**0.034**
MSD	**112.532**	**1.006**	0.171	0.010	**0.059**
MR	**81.416**	**1.008**	0.227	0.019	**0.081**
MDR	**81.371**	**1.008**	0.227	0.019	**0.081**
SCDF	**81.456**	**1.007**	0.227	0.019	**0.081**

Step 2

We have seen that the method of Median and Quartiles (MQ) provides the best parameter estimates based on Stress: $shape = 203.615$ and $scale = 1.002$. Now that we know the parameter estimations of the best method, we perform a grid search in the Cartesian product defined by two intervals around the so-far-optimal parameters. For the shape parameter, we consider an interval of length 100 units asymmetrically centered around the so-far-optimal value, in steps of 0.1 (i.e. around 1,000 values), and for the scale parameter, we consider an interval of length 0.03, asymmetrically centered too, in steps of 0.0005 (i.e. around 60 values), so we explore around 60,000 combinations in total. Due to the highly computationally intensive nature of this grid search, we have parallelized the code here as well.

Based on minimum Stress, our refined estimated parameters are $shape = 159.320$ and $scale = 1.003$, yielding a Stress level of 0.024, which is better than the optimal stress level found in Step 1 (0.034).

Now we are ready to move on to the last step.

Step 3

In this final step we would like to correct for Survivor Bias, by adding slightly more downside risk. As we proved in Section 3.3 (Proposition 1), we can increase the downside risk of a Weibull

distribution by increasing its shape parameter κ.

With this in mind, we perform our last adjustment following the next (pseudo) algorithm:

for each $shape_i/scale_j$ combination **do**
  if $Stress < Stress_{optim} * threshold$ **then**
    $m \leftarrow mean(Weibull_{simulation})$
    if $1 \leq m < mean.gap$ **then**
      $new.shape \leftarrow shape[i]$
      $new.scale \leftarrow scale[j]$
    end if
  end if
end for

where *threshold* is the smallest constant (in steps of 0.5) for which there exists at least an m that lies in the desired half-closed interval $[1, mean.gap)$.

Table 4.3 shows the optimal Weibull parameters found.

Table 4.3: Results and performance after Step 3

Shape	Scale	Stress
170.7193	1.0033	0.0363

Note that we have deliberately increased the Stress, but we have minimized this increment and the value of κ, subject to the constraint that the average gap is between 1 and the empirical average gap (around 1.0000453). This constraint is important for two reasons: first, because we would like to produce simulated gaps that are similar to the ones observed (but with slightly more downside risk, as already explained); second, because this way introducing a gap has no effect in the average annualized return of the simulated prices, and so it should be close to the average observed one for those stocks in the NYSE in the period 1990-2017.

4.2 Modeling standard deviation and white noise

In this section we are going to model the standard deviation of the hourly returns. Since we wish to fit a GARCH model, this will require that we model the white noise component as well.

One of the main aspects that make a GARCH model an attractive choice to model the standard deviation of returns is that such models are designed to produce clusters of volatility, which is a distinctive feature of financial time series in general, and of our 30 stocks in particular: applying the Ljung-Box test to each of our stocks sequentially for different lag orders, we verify that, for 16 out of the 30 stocks (about 53.3%) the null hypothesis of Homoskedasticity is rejected at the 10% significance level for at least one of the six lag orders tried. Rejecting Homoskedasticity in 53.3% of the stocks may not seem completely revealing, but it is indeed remarkable considering the fact that we only have 2 weeks' worth of hourly stock prices, and in such a short period of time it is not uncommon that the evidence of volatility clustering may not be as strong as in

longer periods.

Because we will consider the average estimated coefficients of a GARCH(p,q) for all 30 stocks, we must agree on p and q beforehand. Noting that the probability that there is at least one stock for which at least one of the coefficients is not statistically significant increases as we increase p and/or q, we decide to set $p = q = 1$. Convenience aside, the GARCH(1,1) is indeed one of the most widely used alternatives to model return volatility, probably due to its in general superb ratio performance/parsimony.

The GARCH(1, 1) is defined by the following equation

$$\sigma_t^2 = \omega + \alpha\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2$$

where σ_t is the standard deviation at time t, and ϵ_t is the random shock (also known as white noise, or innovation) that occurs at time t, and which itself follows a certain conditional distribution that we must specify and calibrate with the observed data.

In order to fit the best GARCH(1,1) possible, we try each of the implemented conditional distributions in the package **fGarch** [41] in R, fit the model, record its corrected Akaike Information Criterion (AICc), and eventually choose the conditional distribution (for the white noise) that provides the best (i.e. lowest) AICc on average over our 30 selected stocks.

The AICc is similar to the well-known AIC, but includes a correction for finite sample sizes. It is defined as

$$AICc = AIC + \frac{2k(k+1)}{n-k-1}$$

where k and n are the number of parameters to be estimated and the sample size, respectively.

In general, the AICc tends to be preferred over the AIC (see for example Brockwell & Davis [5], McQuarrie & Tsai [20], and Burnham & Anderson [6]). The general observation is that, since AICc converges to AIC as n gets large, AICc - rather than AIC - should generally be employed.

We find that the best conditional distribution is the Skewed Generalized Error Distribution (SGED), and the final GARCH(1,1) model fitted, which we will use in our simulations, is (rounding to three significant digits)

$$\sigma_t^2 = 8.12 * 10^{-6} + 0.17\epsilon_{t-1}^2\sigma_{t-1}^2 + 0.48\sigma_{t-1}^2$$

where $\epsilon_t \sim SGED(mean = 0\,,\ sd = 1\,,\ shape = 1.40\,,\ skewness = 1.00)$

Note that the shape coefficient of 1.4 is almost exactly half way between that of a Laplace distribution (shape=1) and a Gaussian distribution (shape=2), the former having a (much) larger kurtosis than the latter.

Even more importantly, note that the sum of the α and β coefficients adds to less than 1, otherwise the variance would eventually grow to infinity, as t tends to infinity (note that both $\epsilon_t\sigma_t$ and σ_t are squared, so they are always non-negative, and more importantly, even though $\mathbb{E}(\epsilon_t\sigma_t) = \mathbb{E}(\epsilon_t)\mathbb{E}(\sigma_t) = 0$, $\mathbb{E}(\epsilon_t^2) = Var(\epsilon_t^2) > 0$).

In fact, once checking that for each stock the sum of the GARCH coefficients is below 1, checking that the sum of the coefficients averaged across stocks is below 1 is not necessary strictly speaking, because in that case the latter is guaranteed to remain below 1.

Proof

Consider you have $n \in \mathbb{N}$ stocks, each having their corresponding two GARCH(1,1) coefficients: $(c_1,\ c_2)$, $(c_3,\ c_4)$, $\ldots$, $(c_{2n-1},\ c_{2n})$. Note that the GARCH coefficients $c_i,\ i \in \mathbb{N} \cap [1, n]$, are non-negative by assumption, and this is likely an enforced constraint in the optimization algorithm that the R function `garchFit` from package **fGarch** employs ("Quasi-Maximum Likelihood Estimation").

Let $O = \{1, 3, ..., 2n-1\}$, and $E = \{2, 4, ..., 2n\}$. Therefore,

$$\frac{1}{n}\sum_{i\in O} c_i + \frac{1}{n}\sum_{j\in E} c_j = \frac{1}{n}\sum_{k=1}^{2n} c_k = \frac{1}{n}[(c_1 + c_2) + (c_3 + c_4) + \ldots + (c_{2n-1} + c_{2n})]$$

Now, if $(c_1 + c_2) < 1, \ldots, (c_{2n-1} + c_{2n}) < 1$, then

$$\frac{1}{n}[(c_1 + c_2) + (c_3 + c_4) + \ldots + (c_{2n-1} + c_{2n})] < \frac{1}{n}n = 1$$

□

4.3 Return model 1: Generalized Normally distributed returns

We present the following model for the price of a hypothetical stock that trades in the NYSE:

$$P_t = \begin{cases} P_{t-1} * Gap_t, & \text{for } \ t \ mod \ 7 = 1 \\ P_{t-1} * (1 + r_t), & \text{otherwise} \end{cases} \tag{4.3}$$

where Gap_t is the value of the gap at period t, which occurs with a certain probability that must be determined empirically (in our case, 79.93%)

$$Gap_t = \begin{cases} 1, & \text{if no gap occurs} \\ Weibull(\lambda, \kappa), & \text{if a gap occurs} \end{cases} \tag{4.4}$$

and

$$r_t \sim GED(\mu, \sigma_t, \nu),\ \sigma_t^2 \sim GARCH(1,1) \text{ with } \epsilon_t \sim SGED(\mu = 0, \sigma = 1, \nu, \xi)$$

As already mentioned in Section 3.1, the expected return of this hypothetical asset is set to the 27-year (1990-2017) average hourly return of the NYSE, which assuming compounding is roughly $3.942924 * 10^{-5}$. This return is indeed approximately obtained in our simulations as the average gap is approximately equal to 1.

In order to find the optimal parameters for the model above, we follow a 2-step procedure:

Step 1: Determine the best distribution, in general terms

Step 2: Once the distribution has been fixed, refine the parameter estimation using MLE and grid search to minimize Stress

Step 1 We explore the Student-T with and without skewness parameter, as well as the Generalized Error Distribution (also known as Generalized Normal/Gaussian Distribution), with and without skewness parameter as well.

We select the best distribution based on the lowest AICc, averaged across the 30 stocks in our universe. We find that, on average, the distribution that fits the observed returns best is the Generalized Error Distribution without skewness.

Step 2 We begin refining our parameter estimates by calibrating them through Maximum Likelihood Estimation (MLE) using all the optimization methods available in function `mle` from package **stats4** [31] in R: Nelder-Mead, BFGS, CG, SANN.

We find that for all 30 stocks the optimization method that provides the lowest AICc is the Nelder-Mead, which is a derivative-free optimization heuristic based on the concept of a simplex, especially suited for highly non-linear optimization problems, but which can converge to local - rather than global - optima (like many other optimization methods).

Precisely because we are aware that the optimization might converge to a local optimum only, we finish our refining process by conducting a grid search over the Cartesian product given by the intervals we define around the so-far-optimal Generalized Error Distribution parameters (standard deviation and shape).

Again, we run our grid search in parallel (which, using 3 cores in our machine, cuts CPU time in half[2]) and find the parameter combination that minimizes Stress.

The optimal parameters are $sd = 0.00417$, and $shape = 1.326$.

Therefore, we now have the full model to simulate the hourly returns of a hypothetical NYSE stock.

This model is

$$P_t = \begin{cases} P_{t-1} * Gap_t, & \text{for } \ t \ mod \ 7 = 1 \\ P_{t-1} * (1 + r_t), & \text{otherwise} \end{cases} \tag{4.5}$$

where the Gap_t occurs with a certain probability (again, in our case, 77.93%)

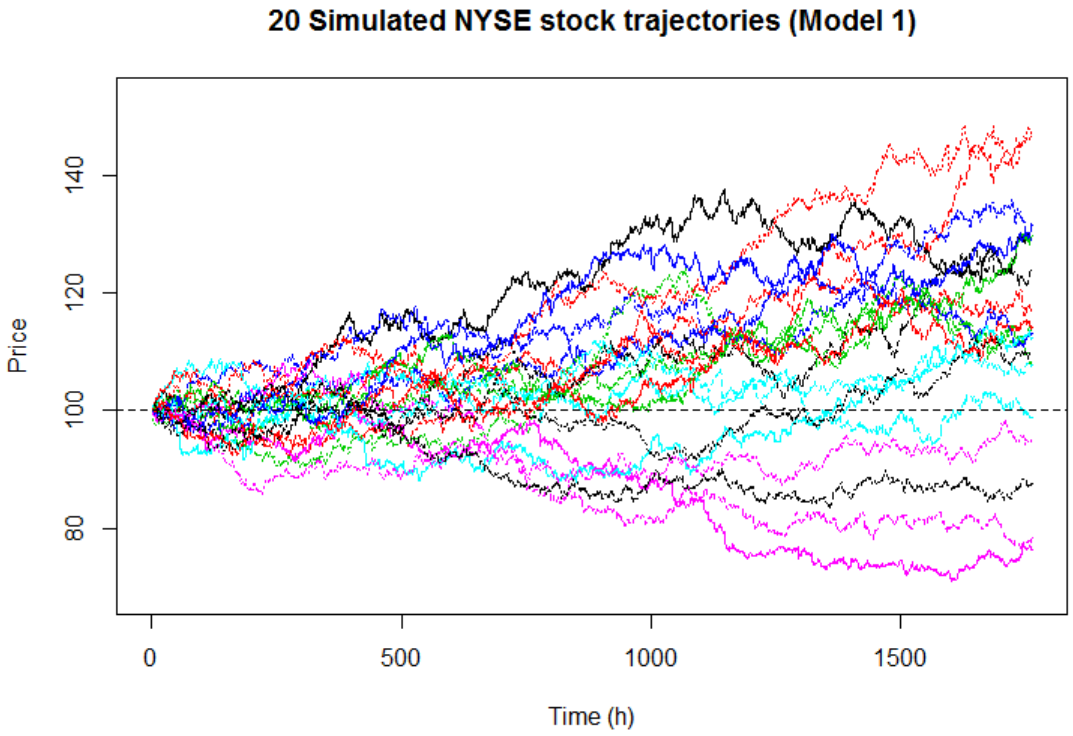
$$Gap_t = \begin{cases} 1, & \text{if no gap occurs} \\ Weibull(1.0033,\ 170.7193), & \text{if a gap occurs} \end{cases} \tag{4.6}$$

[2]According to Amdahl's Law (see [2] and [19]) computing time rarely gets divided by the number of cores when running a process in parallel, as rarely 100% of the process can be parallelized.

and

$r_t \sim GED(3.9429{*}10^{-5},\ 0.00417,\ 1.326),\ \sigma_t^2 \sim GARCH(1,1)$ with $\epsilon_t \sim SGED(\mu = 0, \sigma = 1, 1.4, 0.9275)$

In the graph below, we show 20 randomly simulated 1-year price trajectories obeying Model 1.



The price trajectories seem plausible, and show a behavior that is reminiscent of that of a Random Walk: variance is an increasing function of time. Nonetheless, the mathematical model for the price that we employed here is very different from a Random Walk (which would be a plain AR(1) model with a unity r_{t-1} coefficient).

4.4 Return model 2: ARMA returns

We present our second model for the price of a hypothetical stock that trades in the NYSE:

$$P_t = \begin{cases} P_{t-1} * Gap_t, & \text{for } \ t \ mod \ 7 = 1 \\ P_{t-1} * (1 + r_t), & \text{otherwise} \end{cases} \tag{4.7}$$

where Gap_t is defined as in Equation 4.6, and

$$r_t \sim ARMA(3,0),\ \sigma_t^2 \sim GARCH(1,1) \text{ with } \epsilon_t \sim GED(\mu = 0, \sigma = 1, \nu)$$

As showed in Model 1, it is possible to model volatility as a GARCH and then use that model for variance (or its square root) in the generation of random returns, which need not obey an ARMA model.

However, the GARCH model for volatility works even more naturally when considered in conjunction with an ARMA model for the return, which is the approach we consider here, in an attempt to obtain a model for the return that allows momentum. Following the findings of Kaminski & Lo [18], we would like to test the conjecture that Stop-Loss / Stop-Profit strategies become useful once positive serial correlation is present. Nevertheless, as we will see in a few lines below, our empirical returns do not exhibit positive serial correlation, as not all coefficients associated with past returns are positive. Because, above all, we want to stay as close to our empirical data as possible when modeling asset prices, we keep the coefficients as they are.

Using the R function `garchFit` from package **fGarch** [41], we model simultaneously the return as well as the volatility. After analyzing several different model specifications, and discarding those that produce the slightest numerical problem, we find that an ARMA(3,0) - i.e. an AR(3) - together with a GARCH(1,1) provide a very good compromise between performance and parsimony (avoiding overfitting).

The reason why the ARMA and GARCH models, in general, tend to work so well together may become clear by considering our model, based on Tsay's conventions laid out in "Analysis of Financial Time Series" [37]. In an ARMA(3, 0) + GARCH(1, 1) model,

$$r_t = \phi_0 + \phi_1 r_{t-1} + \phi_2 r_{t-2} + \phi_3 r_{t-3} + a_t$$

where $a_t = \sigma_t \epsilon_t$ and

$$\sigma_t^2 = \omega + \alpha a_{t-1}^2 + \beta \sigma_{t-1}^2$$

$$\epsilon_t \sim GED(0, 1, \nu)$$

Being $\phi_0 = \mathbb{E}(r_t) * (1 - \phi_1 - \phi_2 - \phi_3)$

Therefore, we can see that modeling volatility as a GARCH is directly useful in modeling the returns, if one chooses an ARMA model. This combined ARMA+GARCH model is extremely flexible, and it allows one to capture other so-called "stylized facts" of financial time series, besides volatility clustering: high kurtosis (through the high-kurtosis distribution for the innovations, as well as non-zero autocorrelation among returns (through the AR model itself). Note that skewness of returns might also be included by choosing a skewed distribution for the innovations, but our data, according to the corrected Akaike Information Criterion (AICc) seems to fit best an un-skewed distribution.

For completeness, we present our specific model coefficients (up to three significant digits):

$$r_t = 4.21 * 10^{-4} - 0.05 r_{t-1} - 0.1 r_{t-2} + 0.08 r_{t-3} + a_t$$

where $a_t = \sigma_t \epsilon_t$ and

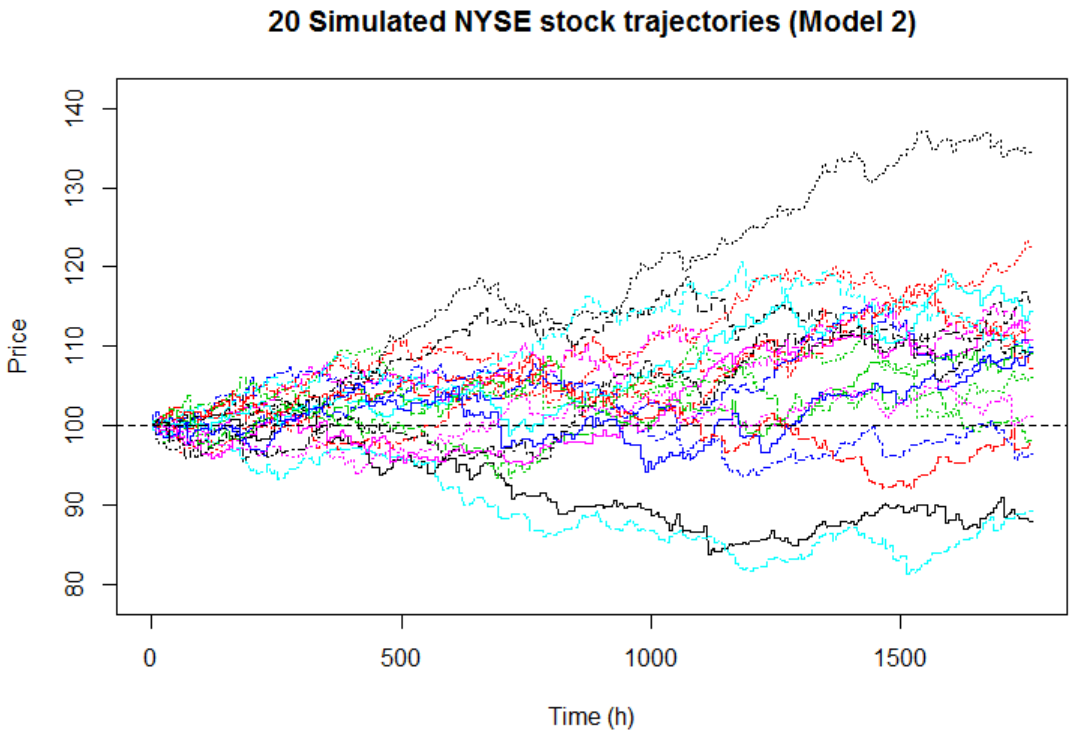
$$\sigma_t^2 = 8.02 * 10^{-6} + 0.16 a_{t-1}^2 + 0.55 \sigma_{t-1}^2$$

$$\epsilon_t \sim GED(0,\ 1,\ 1.47)$$

and we verify that the sum of the α and β GARCH coefficients is indeed below 1. Note also that on this occasion, similar to our volatility specification in Model 1, the shape parameter of the Generalized Error Distribution corresponding to the white noise is half-way between 1 and 2, which means that the kurtosis of this distribution is between that of a Laplace and that of a Gaussian.

Finally, on a procedural level, note that we cannot begin a simulation using such a model right from the start, because we would need 3 previous returns to generate the 4th one in such a fashion. What we do is generate the first return according to Model 1 and Model 2's GARCH, and from that point forward we start using as many return terms as possible from return Model 2, until we have generated 3 returns and can start using Model 2 in full for our simulation.

In the next graph we display 20 randomly simulated 1-year price trajectories that follow Model 2.



Again, the price trajectories seem plausible, and show a similar behavior compared to Model 1, even though the formulations are quite different.

Chapter 5

Data-based approach: Bootstrapping time series

5.1 Introduction to the bootstrap and the block bootstrap

"No current model exists for the DGP [Data Generating Process] of asset returns that is universally accepted by all researchers" - James & Yang [15] (2010). A very similar statement appears in chapter 5.1 of Chernick & LaBudde [9] (2011).

Therefore, a possible, "academically safe" solution would be to just use past observed prices and back-test with them ones' financial strategies (e.g. Stop-Loss policies, and beyond). However, this approach is rather limited, as the observed prices are just one possible realization of an underlying stochastic process. It is, in some sense, similar to basing ones' model on a single observation.

For those reasons, many financial researchers, looking for a more powerful way of using past prices and thereby avoid questionable price models, have resorted to the use of bootstrap methods.

The bootstrap "is one of a number of techniques that is now part of the broad umbrella of nonparametric statistics that are commonly called resampling methods" [9]. As a non-parametric method, the bootstrap does not rely on the specification of a price model. In contrast, the bootstrap uses the actual observed data to generate hypothetical price scenarios, usually making use of Monte Carlo methods, because, for an observed data sequence of length n, there exist n^n possible distinct ordered bootstrap samples (although some would be permutations of each other).

However, the sometimes called "standard bootstrap", which samples (with replacement) individual data observations from the original data series at random, cannot be used because it assumes that the underlying data comes from an independently and identically distributed process. This procedure would generate "bootstrapped series" that would obviously fail to capture the dependent structures exhibited by financial time series. Clearly, one cannot simply construct a simulated financial time series by randomly selecting prices from a pool of past observed prices (and here we are hinting at two problems: dependencies and stationarity).

A first step towards a valid bootstrap method for financial time series is to consider the so-called "block bootstrap". This algorithm (or rather, family of algorithms) generates their bootstrap sequences by concatenating blocks of data randomly sampled from the original data series. Because such sequences are generated from the actual data, they are able to retain some of the original dependencies, such as volatility clustering and chart patterns, but lose them between the blocks. The central idea of the block bootstrap is that for a *stationary* time series, if observations are sufficiently separated in time, they are nearly uncorrelated, so blocks separated far enough can be treated as exchangeable.

There exist a variety of block bootstrap algorithms, generally differing from one another by the way the blocks sizes are determined, as well as the manner in which the concatenations are performed. A popular choice among financial researchers, and the particular choice of James and Yang [15], is the "stationary bootstrap", first introduced by Politis and Romano [27] (1994).

5.2 Strict stationarity and weak dependence

Before we can even discuss the stationary bootstrap as a possible nonparametric model, we must verify our time series (or more precisely, the data generating process of our time series) satisfies two basic requirements imposed by that technique: strict stationarity and weak dependence.

From Wooldridge [40] "the stochastic process $\{x_t : t = 1, 2, ...\}$ is **[strictly] stationary** if for every collection of time indices $1 \leq t_1 < t_2 < \cdots < t_m$, the joint distribution of $(x_{t_1}, x_{t_2}, ..., x_{t_m})$ is the same as the joint distribution of $(x_{t_{1+h}}, x_{t_{2+h}}, ..., x_{t_{m+h}})$ for all integers $h \geq 1$."

The above definition implies, for example, that the joint distribution of (x_1, x_2) must be the same as the joint distribution of (x_t, x_{t+1}), for any $t \geq 1$. In short, strict stationarity requires that the data generating process is exactly the same, in all respects, for all periods.

In theory, it can be very difficult to determine whether our data, which is just one realization of an underlying stochastic process, were generated by a stationary process.

In fact, testing strict stationarity is outside of the scope of the present work. If the reader is interested, a thorough (yet considerably advanced) treatment of this matter can be found in Franco & Zakoian [13] (2012).

Before we continue any further, we must stress a crucial aspect: testing either strict or weak stationarity (also known as covariance stationarity) is, in general, not the same as testing weak dependence.

Stationarity and weak dependence are two concepts that are surprisingly frequently confused. The first concept, as already explained, deals with the joint distribution of a stochastic process as it evolves through time, while the second concept is *exclusively* concerned with how strongly related two observations can be as the time distance between them increases.

In particular, **weak dependence** simply requires that for any two observations x_t and x_{t+h},

$$\lim_{h \to \infty} Corr(x_t, x_{t+h}) = 0$$

i.e. that observations are asymptotically uncorrelated.

If the underlying data generating process is assumed to be an Autoregressive process of order p (AR(p)), then a sufficient condition for this process to be weakly dependent is that the AR coefficient is smaller than 1 in absolute value. This result is commonly referred to as testing for (the absence of) a Unit Root. A straightforward proof of this result (at the price of assuming covariance stationarity in the development) is shown in pp. 380-381 of Wooldridge [40] (2008).

What's more, as stated in p. 391 of Wooldridge (2008), a process may not be stationary but be weakly dependent, and conversely, a process may be stationary (in mean, at least), but not weakly dependent.

Therefore, in purity, we must test for stationarity and for weak dependence separately.

As pointed out before, testing for strict stationarity is out of the scope of the present work, so we will content ourselves with testing for weak stationarity, which only requires that the mean and the second moments remain constant with time changes.

As a matter of fact, this relaxation may not be as severe as it may seem: if the data generating process is Gaussian, then it can be proved that strict and weak stationarity are equivalent.

Moreover, and this may explain why stationarity and weak dependence are often confused, *if* the underlying data generating process is assumed to be an Autoregressive process of order p (AR(p)), then there is a characteristic polynomial corresponding to the model, and the series is covariance stationary if and only if [38] all the roots of the characteristic polynomial are outside the unit circle in the complex plane. So testing for unit roots is a test for weak dependence that serves as a test for weak stationarity for a specific type of time series models (the AR(p)).

Therefore, *if* we assume the data comes from an AR(p) process, then testing for a unit root is sufficient to test both for weak stationarity and for weak dependence.

As shown in Section 4.4, it is reasonable to consider that our returns come from an AR(3) with GARCH(1,1) volatility, so we may just apply the traditional tests for unit root (Augmented Dickey Fuller, KPSS, Phillips-Perron) in order to test both for weak stationarity and for weak dependence.

However, we also explore a somewhat more advanced test called the Priestley-Subba Rao (PSR) test for nonstationarity, introduced by those authors in 1969 [30], and which is based upon examining how homogeneous a set of spectral density function (SDF) estimates are across time, across frequency, or both. The original test was formulated in the terms of localized lag window SDF estimators, but such estimators can suffer from bias due to leakage. To circumvent this potential problem, the SDF estimators are averages of multitaper SDF estimates using orthogonal sinusoidal tapers. This test is implemented in function `stationarity` of the R package **fractal** [10].

We then combine the well-known tests for unit root above (which, on top of testing weak dependence, also test weak stationarity *if* an underlying Autoregressive model is assumed) with the more sophisticated PSR test, specifically conceived to detect non-stationarity. The result is a convenient function called `weakly.stationary` which tests whether a time series is weakly

stationary and weakly dependent (assuming an underlying Autoregressive model).

The way `weakly.stationary` works, in short, is by first performing a Philips-Perron test including a time trend. If the trend is significantly different from 0 (based on a t-test), then automatically the time series must be non-stationary. Otherwise, the Augmented Dickey Fuller test is applied, including just as many lags as necessary so that the Durbin-Watson test applied on the residuals produces a statistic greater than 1.85 [38], thereby rejecting the Null Hypothesis that there's no serial correlation. Including more lags past the point when there's no residual serial correlation is not recommended because the power of the test (in the technical sense) will be reduced; if not enough lags are included, then there will be serial correlation in the residuals and hence the critical values of the ADF test will not be valid (as they assume no serial correlation in the residuals). Then, we apply the KPSS test with the same optimal number of lags as in the ADF test. Finally, we apply the PSR test, and restrict our attention to the T score that this test outputs, which is the test result for spectral variation over time; if the associated p-value is close to zero, then it indicates that there is very strong evidence to reject stationarity.

5.3 The stationary bootstrap

5.3.1 Introduction

Only after having shown our time series is weakly stationary (and possibly strictly too), as well as weakly dependent does it make sense to consider the stationary bootstrap.

The stationary bootstrap is one of the existing block bootstrap methods. Unlike its predecessor, the moving-block bootstrap which uses a fixed block length, the stationary bootstrap uses random block lengths that are distributed according to the geometric distribution. Provided that the original data is stationary, the use of geometrically distributed block lengths ensures that the resampled series are stationary, again unlike the moving-block bootstrap which produces non-stationary samples.

Here we must be cautious: "geometric distribution" is ambiguous because there are two versions of this discrete distribution, each differing in the lower bound of their support. Traditionally, each version is used to model (slightly) different random variables.

On the one hand, one version of the geometric distribution, having support $k \in \{1, 2, 3, ...\}$, may be used to model the probability that the first "success" requires k number of independent trials, each with success probability p. Of course, this implies $k - 1$ *consecutive* failures, and one last successful trial. This is an evident interpretation stemming from its probability mass function (PMF):

$$\mathbb{P}(X = k) = (1 - p)^{k-1}p$$

On the other hand, the other version of the geometric distribution, having support $k \in \{0, 1, 2, 3, ...\}$, may be used to model the number of failures needed until the first success. Clearly, it is certainly possible (if $p > 0$) that the number of failures experienced until the first success is 0, i.e. it is possible to succeed on the first try. From this, it should be clear that its PMF must be given by

$$\mathbb{P}(X = k) = (1 - p)^{k}p$$

This distinction is indeed critical: using the second version, one would allow the possibility of a 0-length block, which is meaningless. Naturally, Politis and Romano [27] chose the correct version of the geometric distribution, in contrast to what it is implied on page 123 of Chernick and LaBudde [9]. However, the latter two authors are correct in pointing out that some normalization must be carried out as the support of the geometric distribution, in either version, is unbounded above, which is obviously meaningless for values of k (the length of the block) greater than n (the length of the sequence). Truncating the support above at $k = n$ is needed and this implies normalizing the PMF so that $\sum_{i=1}^{n} \mathbb{P}(k) = 1$

As shown in Olatayo [25] (2014), the regularization process is straightforward: we look for a constant c in $\mathbb{P}(X = k) = c(1-p)^{k-1}p$ such that $\sum_{i=1}^{n} \mathbb{P}(k) = 1$. Olatayo shows that

$$c = \frac{1}{1-(1-p)^n}$$

which is not hard to understand if one recalls that the cumulative probability function of this version of the geometric distribution is precisely given by $1-(1-p)^k$. Therefore, this implies that at $k = n$ we have accumulated exactly $1-(1-p)^n$ of the probability, and since we are making n the support's upper bound, this accumulated probability should be identically equal to 1, which is accomplished by dividing by the same quantity, which in turn implies a new PMF given by

$$\mathbb{P}(X = k) = \frac{(1-p)^{k-1}}{1-(1-p)^n}$$

for $k \in \{1, 2, 3, ...\}$, and 0 otherwise.

This regularization is, in purity, necessary. However, for large n, it may not make a big difference in practice, and this might be one of the reasons why James & Yang [15] omit it completely, and even Politis and Romano, the original authors, seem to omit it too (and it is only hinted at in Chernick & LaBudde [9]). This relatively small practical importance can be appreciated visually by inspecting the graphs of the PMF and cumulative probability function of an "ordinary" geometric distribution with support starting at 1.

As it can be seen by looking at Figure 5.1 and Figure 5.2, large values of x have relatively small mass.

However, it cannot be overlooked that small values of p make large values of x relatively more probable, as p is the probability of success at any given trial, and x is the number of trials. Therefore, the need for an accurate PMF (i.e. including the regularization) increases (exponentially) as p decreases.

Although we have seen that for quite small values of p, like 0.2, values of x larger than 10 have small mass, we prefer to use the regularized geometric distribution (i.e. the truncated geometric distribution), as it is technically the correct one, even though it seems not many academics use it in the context of the stationary bootstrap: neither James & Yang [15], nor Politis & Romano [27], nor Canty & Ripley [7] (which is an R package whose stationary bootstrap function is based on Politis & Romano), to name a few.

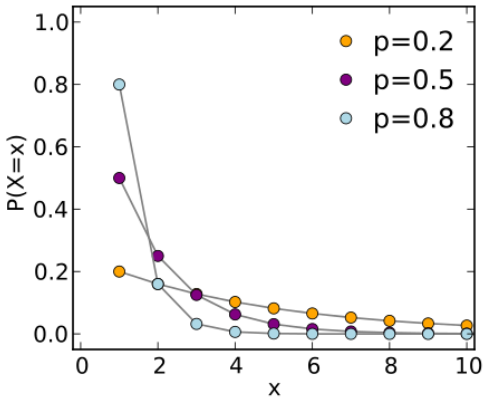


Figure 5.1: Probability mass function of a geometric with support starting at 1

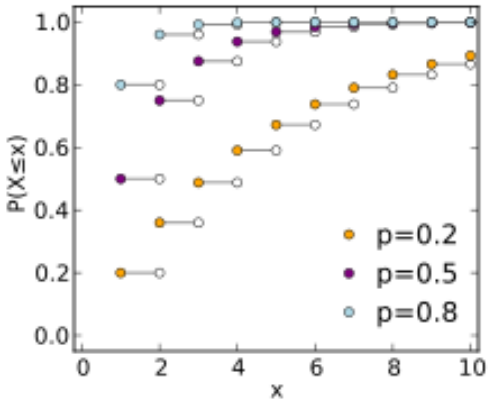


Figure 5.2: Cumulative probability function of a geometric with support starting at 1

5.3.2 Algorithm

Since we have reasons to believe that function `boot` from the R package **boot** [7] does not consider the *truncated* geometric distribution - and therefore the probabilities attached to each block length are not correct - we will implement our own algorithm for the stationary bootstrap: the Truncated Geometric Stationary Bootstrap. Our algorithm is inspired by the algorithm shown in James & Yang [15], which is, in our opinion, an equivalent yet easier-to-follow algorithm compared to the original (Politis & Romano [27]).

1. **Determine the p parameter of the truncated geometric distribution with support starting at 1**, according to which the random block lengths are distributed. In order to find p, we find b via the R implementation [14] of the algorithm described in Politis & White [29], which in turn is based on the notion of spectral estimation via the flat-top lag-windows of Politis & Romano [28]. The relationship between b and p is that $b := \mathbb{E}(L)$, where L is the length of the block, and in turn $L \sim trunc.Geom(p)$, with N terms.

   Now, as Olatayo [25] showed, the expectation of a truncated geometric distribution with support starting at 1 is given by

   $$\mathbb{E}(L) = \frac{1 - q^N + q - Nq^N(1-q)}{(1-q^N)(1-q)}$$

   where $q = 1 - p$. Therefore,

   $$\frac{1 - q^N + q - Nq^N(1-q)}{(1-q^N)(1-q)} = b$$

   which, manipulating algebraically, implies

   $$(N-b)q^{N+1} + (b-1-N)q^N + (1+b)q + 1 - b = 0$$

   which is a polynomial of degree $(N+1)$ in variable q that we can solve numerically, and then obtain the sought after p parameter.

Note that b is a constant in the above equation. In particular, we find b using function `b.star` from package **np** [14], as anticipated previously.

2. Let X_i, $i = \{1, 2, ..., N\}$ be the **original data series**, and let X_i^*, $i = \{1, 2, ..., N\}$ be the **bootstrapped series**

3. The **first bootstrapped observation** X_1^* is selected at random from the original data $\{X_N\}$, which corresponds to observation X_n in the original data. We interpret that by "at random" James & Yang [15] mean that all observations are given a probability of $\frac{1}{N}$, i.e. the observations are selected according to a discrete uniform distribution.

4. For X_i^*, $i \in \{1, \cdots, N-1\}$,

   a) if $X_i^* = X_N$, then

   $$X_{i+1}^* = \begin{cases} X_m, & \text{with probability} \quad p \\ X_1, & \text{with probability} \quad 1-p \end{cases} \tag{5.1}$$

   b) if $X_i^* \neq X_N$, then

   $$X_{i+1}^* = \begin{cases} X_m, & \text{with probability} \quad p \\ X_{n+1}, & \text{with probability} \quad 1-p \end{cases} \tag{5.2}$$

   where X_1 and X_{n+1} are, respectively, the first and the next observation after X_n in the original data series. The above equations just mean that with probability p, X_{i+1}^* is randomly selected from $\{X_N\}$ (as in step 3), and with probability $1-p$, X_{i+1}^* is deterministically assigned the value of the first or the next observation in the original data.

The above algorithm is an alternative, more compact implemention of the Truncated Geometric Stationary Bootstrap than the one proposed by Olatayo [25] as a correction to the original (untruncated) Geometric Stationary Bootstrap. Furthermore, our version is also different in that it makes use of the automatic block-length selection heuristic laid out in Politis and White [29] (2004).

5.3.3 Applying the algorithm

After using our R function `weakly.stationary` as per Section 5.2, we determine that at the 1% significance level, eleven stocks (BAC, PFE, F, C, JPM, MRK, XOM, AIG, HAL, KEY, and BMY) seem to have stationary hourly returns (at least in the period considered: March 13 to May 26, 2017). Of course, we didn't even try with the prices, as they are most definitely not stationary, and clearly so because after differencing (computing the returns) 19 stocks still are not stationary at the 1% significance level (although we must stress that most of the non-stationarity has been detected only according to the PSR test). More intuitively, it just doesn't make sense to resample blocks of prices, because it would be problematic to adjust the blocks' ends (because we are implicitly assuming that prices are not stationary: neither in mean nor in variance, at least).

Therefore, as the Truncated Geometric Stationary Bootstrap algorithm requires stationarity and weak dependence, we apply the algorithm to only those eleven stocks.

In order to implement the Truncated Geometric Stationary Bootstrap, and since, to the best of our knowledge, there is currently no R function that implements it using the correct probability mass function, we create our own function `SBoot_trunc.geom(tseries)`, which takes one argument `tseries` (hourly returns in our case) and outputs the bootstrapped series (again, hourly returns). We also created a special-purpose wrapper function `SB_trunc.geom.Prices` that outputs a vector with the bootstrapped *prices* (not returns) of the desired stock.

Therefore, under this nonparametric approach, our price model is

$$P_t = P_{t-1}(1 + r_t^*)$$

for all $t \geq 1$, where r_t^* is the bootstrapped return at time t.

In effect, this price model, compared to the other two - parametric - models, is quite simple to state, but the complexity lies in correctly computing r_t^*, as the previous algorithm showed, as well as proving the algorithm preserves the desired properties of the observed returns (the so-called "stylized facts").

Graphs showing both the empirical and the bootstrapped price evolution (March 13 - May 26, 2017) for each weakly stationary stock can be found in Appendix B.

Chapter 6

Testing Stop-Loss rules

6.1 Stop-Loss rules implemented

6.1.1 Fixed percentage strategy

As mentioned in Section 2.1, we implement the following Stop-Loss / Stop-Profit rule:

$$SL_t = P_{t-1}^{max}(1-a)$$

where $a > 0$ is the maximum percentage of the highest price achieved until time $t-1$ that the investor is willing to lose in that operation. And

$$SP_t = P_{t-1}^{max}(1+b)$$

where $b > 0$ is the percentage of the highest price achieved until time $t-1$ that the investor would be realistically satisfied to win in that operation.

For $a \in [0.03, 0.1]$ in steps of size 0.01 and for $b \in [0.04, 0.15]$ U $\{10^6\}$ in steps of size 0.01 too, we perform a grid search over the Cartesian product that those two intervals define, in order to find the best pair of parameters in terms of a) expected return, b) Sharpe Ratio, c) Sortino Ratio, d) Return-VaR ratio, and e) Return-ES ratio. That is, we analyze 5 different optimization criteria for the parameters a and b above.

Note that by allowing $b = 10^6$ we are effectively including the possibility of not having a Stop-Profit barrier.

Further note that the ranges for a and b are not identical (besides the obvious difference in their most extreme attainable value). This asymmetry makes sense both from a mathematical perspective, and from a utility-theoretical one. In the first case, because there is no mathematical reason why the best performance cannot be delivered by an asymmetric configuration; and in the second case, because this asymmetry, in practice, may help correct for an important behavioral bias: an individual becomes less risk-averse when facing mounting losses, so we need to force her out sooner than she might otherwise be willing to leave; on the other hand, individuals

tend to be excessively risk-averse in the face of mounting gains. For these reasons, it is in general preferable that $a < b$, unless there is strong evidence that otherwise is in fact more profitable in a particular case. But of course, we trust our parameter optimization function `pctg.barrier.param.optimizer` and we will use the values for a and b that this routine finds as optimal based on 10^4 repetitions (which on our main machine[1] takes more than 10 hours in the fastest case (model 3), even with parallelized code).

Once we find the best parameter pair for each of those 5 metrics, we record their performance and present the results in each metric, together with the results of a Buy-and-Hold strategy and those of Model 2. These results, which include the optimal (a,b) parameter pair used in each case, can be found in Table 6.1, for models 1 and 2, and in Table 6.5 for model 3.

For this Stop-Loss rule, the algorithm is virtually identical in both the model-based and the data-based frameworks. The only slight difference worth mentioning between both frameworks is that for the data-based framework we optimized the barrier parameters to maximize expected return only (and not the other 4 performance metrics), as it will be discussed in Section 6.3.

6.1.2 ATR

We simulate a Stop-Loss / Stop-Profit policy based on the Average True Range, as defined in Section 2.3,

$$SL_t = P_{t-1} - \alpha ATR_t$$

$$SP_t = P_{t-1} + \beta ATR_t$$

where α and β are real numbers usually picked between 1.5 and 3.

We choose the usual time frame of 14 days and values for α and β of 2.5 and 3, respectively. With smaller parameter values we observe that the stop is triggered too early. The slight asymmetry is intended to correct for the behavioral biases discussed previously.

For the ATR rule, the algorithm is identical in both the model-based and the data-based frameworks: the same period of 14 days is considered, even though in the first framework we simulate prices for a whole (trading) year, while in the second framework we simulate prices for only two and a half months (as our data in this case are real hourly prices from March 13 to May 26, 2017).

The results obtained are shown in Table 6.2 for models 1 and 2, and in Table 6.6 for model 3.

6.1.3 RSI

We simulate a Stop-Loss / Stop-Profit policy based on the Relative Strength Index, as defined in Section 2.4.2.

[1] Windows 10 64-bit, Intel Core i5-5200U CPU @ 2.20GHz, 8.0 GB RAM

We choose a time frame of 7 days and set our threshold at the usual level of 70 for Model 1, and at 85 for Model 2, as it appears that the RSI is more volatile in Model 2, and hence a higher threshold avoids an excessively early exit.

As stated in Section 2.4.2, the investor liquidates her position the moment the RSI reaches or surpasses the threshold level set, which is thought to mean the stock is "overbought" by that point.

In the data-based simulation framework, we use the exact same configuration as in Model 1: a time horizon of 7 days and exit level set at a score of 70.

The results obtained are shown in Table 6.3 for models 1 and 2, and in Table 6.7 for model 3.

6.1.4 Triple MA crossover

We simulate a Stop-Loss / Stop-Profit policy based on the triple crossover of a short, medium and long - term Moving Average, as defined in Section 2.4.1.

We choose the Moving Averages of 5, 20 and 70 days for the short, medium and long terms, respectively.

Provided the shorter-term MAs are above the longer-term ones, the exit signal is produced, as stated in Section 2.4.1, when we have the case that the MA-5 is below both the MA-20 and the MA-70, and the MA-20 is below the MA-70. This pattern suggests the reversal of an upward trend, and thus it would be advisable to liquidate the position.

However, we must stress that in the data-based framework, since the simulation period is much shorter than in the model-based one (two and a half months compared to twelve), it would not make much sense to still use a Moving Average of 70 days (as, in fact, we just have 54 trading days!), which naturally suggests that we adjust the time-frame for the shorter-period Moving Averages as well. As such, for the data-based framework we use the MA-3, MA-12, and MA-30.

The results obtained are shown in Table 6.4 for models 1 and 2, and in Table 6.8 for model 3.

6.2 Model-based simulation results

Table 6.1: Fixed percentage SL: Simulation results

Strategy-Model	$\mathbb{E}$(r)	Sharpe	Sortino	R-VaR	R-ES
SL - Model 1	(0.1, 0.07) 4.09%	(0.03, 0.10) 0.81	(0.09, 0.04) 0.15	(0.03, 0.14) 0.83	(0.03, 0.09) 0.69
B&H - Model 1	4.96%	0.30	0.18	0.15	0.13
SL - Model 2	(0.1, 0.08) 5.90%	(0.03, ∞) 0.75	(0.10, 0.05) 0.53	(0.03, 0.04) 0.72	(0.03, 0.09) 0.60
B&H - Model 2	6.44%	0.57	0.56	0.35	0.28

Table 6.2: ATR-based SL: Simulation results

Strategy-Model	$\mathbb{E}$(r)	Sharpe	Sortino	R-VaR	R-ES
SL - Model 1	3.96%	0.36	0.12	0.20	0.15
B&H - Model 1	4.97%	0.30	0.18	0.16	0.13
SL - Model 2	3.59%	0.88	0.17	0.57	0.42
B&H - Model 2	6.45%	0.58	0.56	0.35	0.28

Table 6.3: RSI-based SL: Simulation results

Strategy-Model	$\mathbb{E}$(r)	Sharpe	Sortino	R-VaR	R-ES
SL - Model 1	4.50%	0.32	0.14	0.25	0.20
B&H - Model 1	5.00%	0.30	0.18	0.15	0.13
SL - Model 2	3.47%	1.13	0.12	0.60	0.41
B&H - Model 2	6.45%	0.57	0.56	0.35	0.28

Table 6.4: MA crossover - based SL: Simulation results

Strategy-Model	$\mathbb{E}$(r)	Sharpe	Sortino	R-VaR	R-ES
SL - Model 1	4.33%	0.32	0.14	0.17	0.13
B&H - Model 1	4.96%	0.30	0.18	0.16	0.13
SL - Model 2	5.28%	0.57	0.44	0.35	0.26
B&H - Model 2	6.45%	0.58	0.56	0.35	0.28

Conclusions

- Under both Model 1 and Model 2, there exist Stop-Loss / Stop-Profit strategies that provide a higher risk-adjusted return than Buy-and-Hold, for all risk-adjusted metrics except the Sortino ratio. In some cases, the improvement is very significant; for example the improvement in R-VaR when using the fixed percentage Stop-Loss / Stop-Profit rule, *irrespective* of whether the underlying stochastic process that drives stock returns is a Generalized Normal Distribution or an ARMA(3, 0).
- The expected return when a Stop-Loss policy is in place is always smaller than when it isn't. This is actually a predictable consequence of the fact that, in our model, because we wanted to model a "typical" stock in the NYSE, we imposed a small drift term that ensured the expected annual return would be around 5%-7% with a very high probability, which is a lot higher than the risk-free rate that we have considered (3.17%) and that the portfolio starts earning as soon as the Stop is triggered. Under such scenario, it is extremely difficult to devise a strategy that is systematically able to beat Buy-and-Hold in terms of expected return by switching to a risk-free asset at some point during the year and never coming back.
- The usefulness of each Stop-Loss rule is similar in both models, i.e. when the rule is useful for one model, it is for the other model as well. However, by carefully examining the tables, we may conclude that, at least in our simulations: the fixed percentage rule, as well as the triple Moving Average crossover are relatively more useful if the underlying stochastic process is in reality (close to) Model 1; in contrast, the ATR and RSI rules are relatively more powerful if the underlying stochastic process is in reality closer to Model 2.

6.3 Data-based simulation results

Tables 6.5, 6.6, 6.7, and 6.8 show the results obtained for the Stop-Loss / Stop-Profit framework under the four Stop-Loss rules considered, and Table 6.9 show the results for the Buy-and-Hold strategy.

In the first table, we have optimized the barrier parameters individually for each stock in order to maximize performance according to the expected return criterion, as in most stocks here the performance metrics are below zero, which leave them essentially meaningless except for the expected return, which is such a simple metric that under all circumstances can be unambiguously interpreted.

Moreover, it must be stressed that the values in Table 6.9 have been obtained by averaging the values in all the Buy-and-Hold tables that accompany each Stop-Loss / Stop-Profit table in their original form (not shown here). The way we created this table of (very similar) average values is by row-binding the 4 Buy-and-Hold tables derived as a by-product when calculating the performance of each Stop-Loss rule, and then compute the mean of each column grouped by stock (row names). There is no function that accomplishes this, and so we created a new function, `rowmean` which is essentially a wrapper of function `rowsum` (a very efficient function in base R to compute column sums by group).

Table 6.5: Truncated Geometric Stationary Bootstrap: results under fixed % SL-SP

Stock	SL.expret	SL.Sharpe	SL.Sortino	SL.RVAR	SL.RES
BAC	−0.1776	−0.0584	−0.3774	−0.5771	−0.4750
PFE	−1.6250	−1.2779	−0.8879	−1.9837	−1.5457
F	−1.1911	−0.5818	−0.7304	−1.0902	−0.7889
C	1.3577	0.1510	0.1362	−0.0126	−0.0121
JPM	−0.5532	−0.2252	−0.5555	−0.8903	−0.7374
MRK	0.2513	0.0780	0.2139	−0.3957	−0.3483
XOM	0.4234	0.1281	0.1284	−0.3124	−0.2891
AIG	3.1993	0.4151	0.6792	0.2428	0.2292
HAL	−0.2553	−0.0851	−0.4088	−0.6507	−0.5380
KEY	0.3231	0.0839	0.1585	−0.4259	−0.3529
BMY	−0.9611	−0.4972	−0.7137	−1.3190	−1.1418

Table 6.6: Truncated Geometric Stationary Bootstrap: results under ATR SL-SP

Stocks	SL.expret	SL.Sharpe	SL.Sortino	SL.RVAR	SL.RES
BAC	−5.9667	−0.6003	−0.5919	−0.3894	−0.3148
PFE	−4.5217	−1.2204	−0.8162	−0.6335	−0.5030
F	−9.9151	−1.3213	−0.8187	−0.7171	−0.5762
C	1.4588	0.1582	0.1374	0.0756	0.0614
JPM	−5.3045	−0.7670	−0.6747	−0.4691	−0.3780
MRK	−0.2619	−0.0545	−0.2337	−0.0579	−0.0460
XOM	0.1452	0.0273	0.1348	0.0007	0.0006
AIG	3.0206	0.4207	0.6458	0.2484	0.2004
HAL	−8.8081	−0.7911	−0.6737	−0.5355	−0.4391
KEY	−2.2297	−0.1993	−0.3069	−0.1569	−0.1265
BMY	−6.4733	−1.0806	−0.7740	−0.6605	−0.5342

Table 6.7: Truncated Geometric Stationary Bootstrap: results under RSI SL-SP

Stock	SL.expret	SL.Sharpe	SL.Sortino	SL.RVAR	SL.RES
BAC	−6.9405	−0.6392	−0.6080	−0.4730	−0.3867
PFE	−5.6835	−1.4322	−0.8500	−0.8825	−0.7060
F	−12.6406	−1.5410	−0.8547	−1.0653	−0.8613
C	1.5160	0.1650	0.1436	0.0947	0.0770
JPM	−5.9711	−0.8022	−0.6867	−0.5623	−0.4567
MRK	−0.3376	−0.0660	−0.2347	−0.0548	−0.0443
XOM	0.1320	0.0239	0.1321	0.0069	0.0056
AIG	3.0394	0.4322	0.6293	0.2751	0.2244
HAL	−9.2419	−0.7961	−0.6757	−0.6040	−0.4969
KEY	−2.6581	−0.2237	−0.3237	−0.1856	−0.1524
BMY	−7.1498	−1.1651	−0.7943	−0.7808	−0.6305

Table 6.8: Truncated Geometric Stationary Bootstrap: results under MA SL-SP

Boot.Perf.df	SL.expret	SL.Sharpe	SL.Sortino	SL.RVAR	SL.RES
BAC	−6.7485	−0.6443	−0.6116	−0.4184	−0.3424
PFE	−5.4526	−1.3503	−0.8364	−0.8115	−0.6514
F	−12.4293	−1.5484	−0.8545	−1.0095	−0.8205
C	1.4548	0.1550	0.1352	0.0666	0.0536
JPM	−5.8825	−0.8154	−0.6913	−0.5034	−0.4105
MRK	−0.3131	−0.0621	−0.2346	−0.0484	−0.0388
XOM	0.1397	0.0257	0.1343	−0.0007	−0.0005
AIG	3.1469	0.4170	0.6664	0.2360	0.1886
HAL	−9.0873	−0.8085	−0.6797	−0.5406	−0.4447
KEY	−2.6610	−0.2266	−0.3315	−0.1725	−0.1414
BMY	−6.8331	−1.1178	−0.7822	−0.6982	−0.5673

Table 6.9: Truncated Geometric Stationary Bootstrap: results under Buy-and-Hold

Stock	BH.expret	BH.Sharpe	BH.Sortino	BH.RVAR	BH.RES
BAC	−7.1794	−0.6747	−0.6273	−0.4822	−0.3943
PFE	−5.6818	−1.4347	−0.8502	−0.8827	−0.7053
F	−12.8653	−1.6387	−0.8669	−1.0756	−0.8700
C	1.5171	0.1578	0.1423	0.0741	0.0603
JPM	−6.1557	−0.8472	−0.7039	−0.5708	−0.4628
MRK	−0.3745	−0.0722	−0.2421	−0.0623	−0.0502
XOM	0.1034	0.0182	0.1392	−0.0061	−0.0049
AIG	3.3076	0.4296	0.6963	0.2527	0.2053
HAL	−9.5564	−0.8444	−0.6948	−0.6144	−0.5040
KEY	−2.8110	−0.2334	−0.3376	−0.1981	−0.1626
BMY	−7.1723	−1.1828	−0.7980	−0.7790	−0.6288

Conclusions

- First of all, it must be stressed that in the period March 13 to May 26, all the stocks under consideration, except Citigroup (C) and American International Group (AIG), experienced a negative return, which explains why most of the expected returns are negative, especially under Buy-and-Hold. This observation reinforces our belief that the Truncated Geometric Stationary Bootstrap has been able to capture well the real behavior of stocks. This impression is in fact proved by James and Yang [15], based on the Brock, Lakonishok and LeBaron (BLL) trading rule test.

- In such a situation in which 9 out the 11 stocks have experienced a negative return in the period considered, the Stop-Loss rules considered have provided a higher expected return, in some cases extremely higher than if no Stop-Loss had been used, as in the case of the fixed percentage strategy, which has achieved a positive expected return in 5 out of 11 stocks, and near positive in two more stocks. In fact, on average, the fixed percentage rule has increased the expected return in over 4 percentage points compared to Buy-and-Hold, and in some cases (Ford), as much as 11.7 percentage points higher than Buy-and-Hold, on average (in 200,000 repetitions), in just two and a half months! In a full year, the improvement might be even more spectacular.

- Of course, this is the opposite case as in our model-based simulation: in this case, using real (bootstrapped) prices, because mid March to late May has been a bad period for most of the stocks under consideration, it makes sense that a strategy that at some point switches to a risk-free asset will perform better than merely Buy-and-Hold. However, in the case of the fixed percentage rule, the expected return premium over Buy-and-Hold in a Bear (i.e. falling) market is much higher than the expected return premium of Buy-and-Hold over this Stop-Loss rule in a Bull (i.e. rising) market, in general.

- As already pointed out, it makes little sense to discuss risk-adjusted measures when the expected return in excess of the risk-free rate is negative, because in that case less volatility would in fact worsen the metric, ceteris paribus. However, we cannot simply consider that

a more negative risk-adjusted metric is better (having in mind that this might mean the strategy induces less volatile returns), because a more negative ratio might as well come from a very negative excess expected return (which is obviously undesirable). For these reasons, under such circumstances, we just focus on expected return, and according to this metric, all Stop-Loss rules beat Buy-and-Hold... and the fixed percentage rule by a very wide margin in some cases.

- In contrast, if we focus our attention on the only two stocks for which the vast majority of performance metrics are above 0, both under SL-SP rules and under Buy-and-Hold (which coincide with the ones whose empirical return for the period of interest is positive: Citigroup and AIG), we conclude that also in this case we can find a Stop-Loss rule that improves performance over Buy-and-Hold: for Citigroup, the RSI-based Stop-Loss beats Buy-and-Hold in all risk-adjusted metrics (including the Sortino ratio); for AIG, the same Stop-Loss rule beats Buy-and-Hold for all metrics (even including expected return), except the Sortino ratio.

6.4 Overall simulation conclusions

We have seen that, on the one hand, in the case of a Bull market, some Stop-Loss rules provide a better risk-adjusted return than Buy-and-Hold under all risk-adjusted metrics except the Sortino ratio, and this has been verified under the three models considered in this work.

On the other hand, in the case of a Bear market, all Stop-Loss rules provide a better expected return than Buy-and-Hold. In the case of the fixed percentage rule, the improvement in expected return is outstanding in many cases.

Therefore, we have shown that no matter the situation (Bull or Bear market), Stop-Loss rules provide a better expected risk-adjusted return, at least. Because the vast majority of human beings (investors included) are risk-averse, it is reasonable to claim that risk-adjusted returns are more important than "raw" returns, and so in this case we see that Stop-Loss strategies do add value.

One might then wonder, which Stop-Loss rule is the best one. As the reader might have already realized, it turns out that the fixed percentage rule may be considered the best one, especially in a Bear market. There are at least two reasons why that rule has turned out to be the best:

1. The very nature of the rule ensures relatively good results, because we always take as a reference the maximum price achieved until the preceding period. Of course, this is so unless an overnight gap bypasses the barriers and thereby renders the rule useless (which is why it is crucial to consider overnight gaps when evaluating the usefulness of Stop-Loss strategies, as otherwise results might be inflated and/or misleading).

2. It is the only rule whose parameters have been optimized. The other three rules can also be optimized, although we have used the parameters that are, by far, most popular among practitioners. In any case, optimizing those parameters (and thus possibly departing from mainstream practices) may be something worth considering for further research.

Table 6.10 shows in a rational manner why it is reasonable to claim that the overall best Stop-Loss strategy is the fixed percentage rule. The winner rule in each cell has been found by comparing, for a given price model (GED, ARMA or Bootstrap), which rule improved the most (or worsened the least) each performance metric, compared with Buy-and-Hold.

Table 6.10: Best SL-SP rule for each Model-Criterion pair

Model	$\mathbb{E}$(r)	Sharpe	Sortino	R-VaR	R-ES
1	RSI	%	%	%	%
2	%	RSI	%	%	%
3	%	(%)	(%)	(ATR)	(ATR)

Chapter 7

Conclusion and further research

In this Thesis we provide an answer to the question: "do Stop-Loss rules stop losses?", and, by considering two alternative price models as well as bootstrapping empirical asset prices, we are confident to claim that the answer is, with some nuances, affirmative.

In particular, we have shown that, on the one hand, in rising markets, some Stop-Loss rules provide a better risk-adjusted return than Buy-and-Hold under all risk-adjusted metrics except the Sortino ratio.

On the other hand, in the case of a falling market, all Stop-Loss rules provide a better expected return than Buy-and-Hold. In the case of the fixed percentage rule, the improvement in expected return is outstanding in many cases, providing a boost in expected return of up to 11.7 percentage points over Buy-and-Hold, in a time horizon of just two and a half months.

Therefore, we have shown that irrespective of whether the market is rising or falling, Stop-Loss rules improve investment performance in terms of expected risk-adjusted returns, at least. Because humans are, in general, risk-averse it is reasonable to focus on risk-adjusted returns rather than absolute returns, and so in this case we see that Stop-Loss strategies do add value.

In the process of answering the question that motivated this work, we have made contributions to the existing literature on Stop-Loss rules (through a rigorous treatment of overnight gaps), as well as in other, more general areas of Financial modeling and performance evaluation (e.g. the definition and motivation of two new risk-adjusted performance metrics). In particular, considering overnight gaps is crucial because their occurrence in real financial markets hurts Stop-Loss performance, as it bypasses any barriers set, and therefore this way we are able to obtain a more accurate (less inflated) picture of the usefulness of Stop-Loss policies.

Finally, we may highlight our contribution to the Stationary Bootstrap, by refining the currently dominant algorithm, and implementing our version in R code, which is somewhat more compact than the version of Olatayo [25] and makes use of the automatic block-length selection heuristic laid out in Politis and White [29] (2004).

A natural continuation of our study would be to go beyond pure Stop-Loss rules and explore stop-loss-re-entry rules, similarly to what Kaminiski and Lo [18] (2014) did. Moreover, other Stop-Loss

(or stop-loss-re-entry) rules may be considered. A particular rule that we find interesting is one based on the theory of Conic Finance, which revolves around the concept of Bid and Ask prices, which may serve as Stop-Loss / Stop-Profit barriers, respectively.

Appendix A

Range heuristic for the Generalized Normal Distribution

$\underline{\beta = 0.5}$

$$\sigma \approx \frac{Range(X_n)}{0.519(\ln n)^{1.733} + 0.746}$$

$\underline{\beta = 1 \text{ (Laplace distribution)}}$

$$\sigma \approx \frac{Range(X_n)}{1.26(\ln n)^{1.039} + 0.191}$$

$\underline{\beta = 1.3 \text{ (Empirical hourly returns)}}$

$$\sigma \approx \frac{Range(X_n)}{1.714(\ln n)^{0.816} - 0.217}$$

$\underline{\beta = 1.6}$

$$\sigma \approx \frac{Range(X_n)}{2.146(\ln n)^{0.671} - 0.618}$$

$\underline{\beta = 2 \text{ (Normal distribution)}}$

$$\sigma \approx \frac{Range(X_n)}{2.87(\ln n)^{0.515} - 1.312}$$

$\underline{\beta = 2.5}$

$$\sigma \approx \frac{Range(X_n)}{4.012(\ln n)^{0.373} - 2.42}$$

$\underline{\beta = 3}$

$$\sigma \approx \frac{Range(X_n)}{4.801(\ln n)^{0.258} - 3.168}$$

$\underline{\beta = 3.5}$

$$\sigma \approx \frac{Range(X_n)}{4.303(\ln n)^{0.169} - 2.599}$$

$\underline{\beta = 5}$

$$\sigma \approx \frac{Range(X_n)}{15240(\ln n)^{0.0000002} - 15238} \approx \frac{Range(X_n)}{2}$$

In fact, looking at Figure A.1 it seems one can infer a pattern relating the shape parameter (β) with the multiplicative constant (m), the additive constant (a) and the power (p).

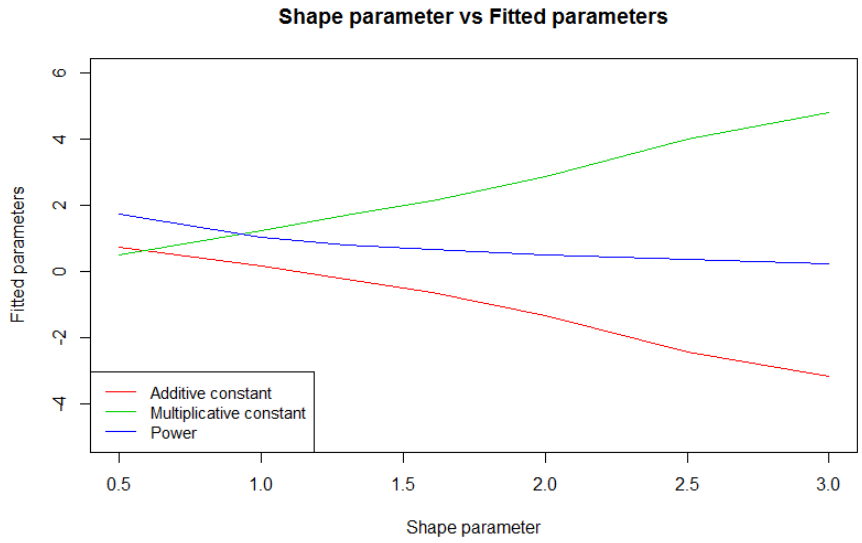


Figure A.1: Shape parameter against fitted parameters

As a very simple approximation, and only for values of β between 0.5 and 3, here are some decent functional forms in terms of Least Squares:

$$m = 1.749\beta - 0.499 \qquad [R^2 = 0.9876]$$

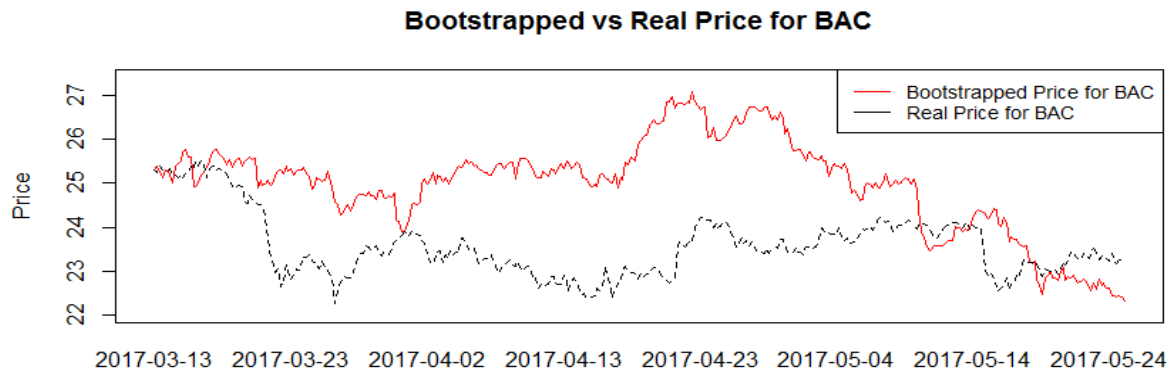
$$a = -1.621\beta + 1.785 \qquad [R^2 = 0.9938]$$

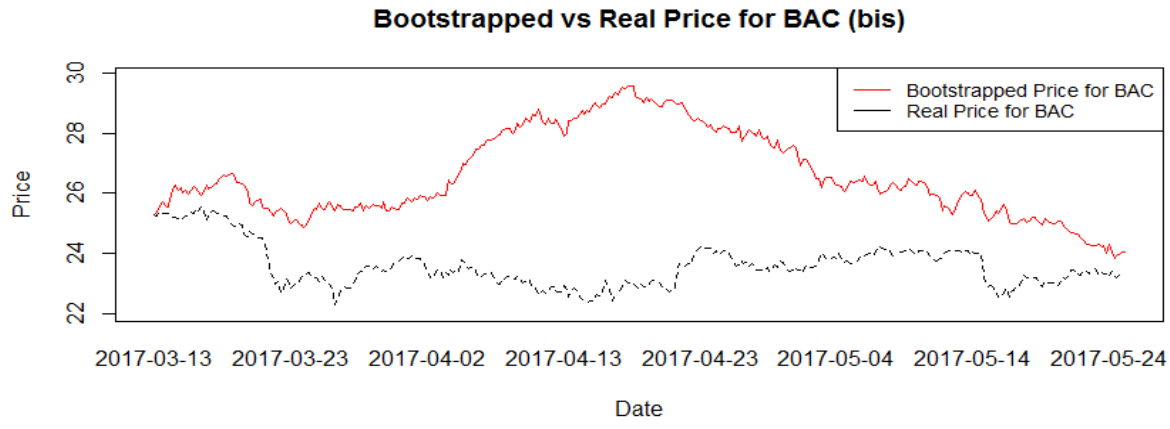
$$p = -0.816 \ln \beta + 1.092 \qquad [R^2 = 0.9876]$$

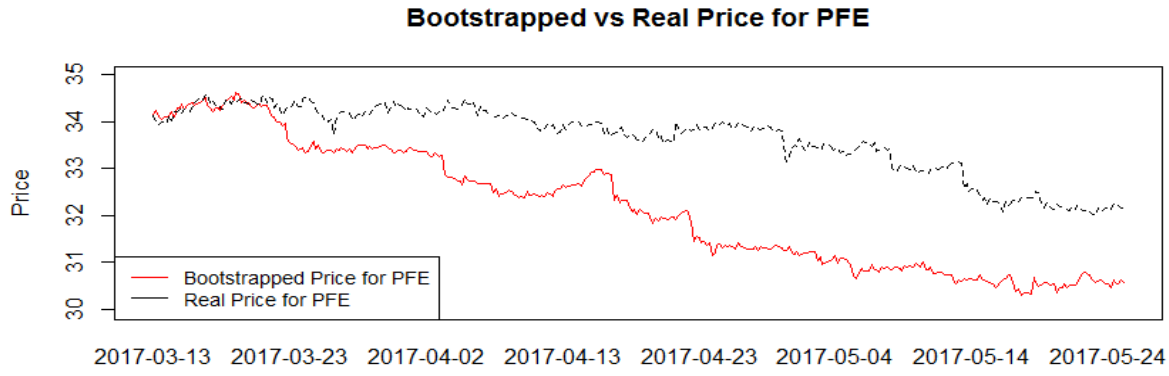
Hence, using the above heuristics for specific values of the shape parameter β, together with the above equations relating β to the other parameters, one should be able to obtain a decent range heuristic for the case of a random variable X following a Generalized Normal distribution with any combination of parameters μ, σ, β (at least for $\beta \in [0.5, 3]$, with confidence).

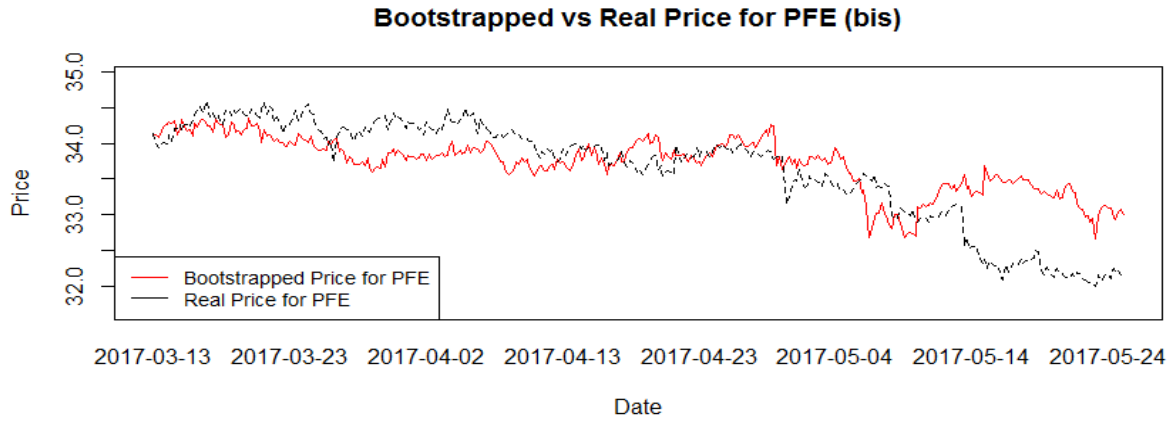
Appendix B

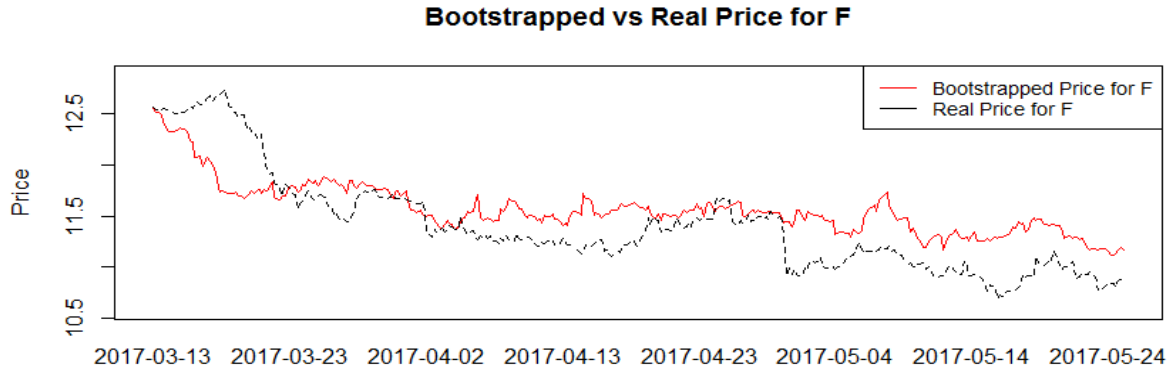
Truncated Geometric Stationary Bootstrap graphs

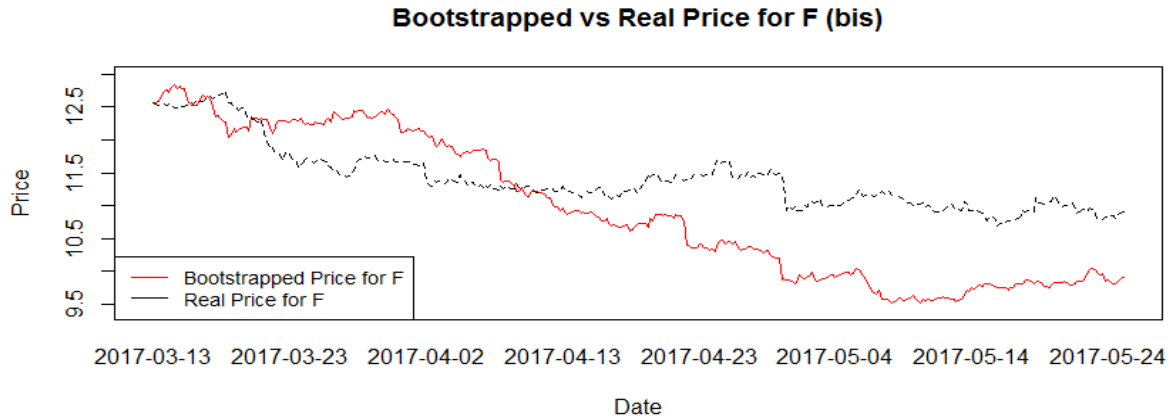


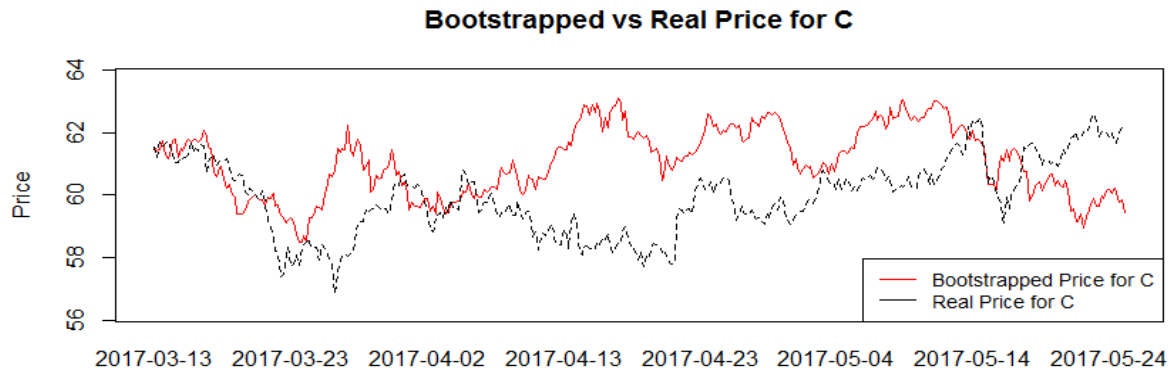


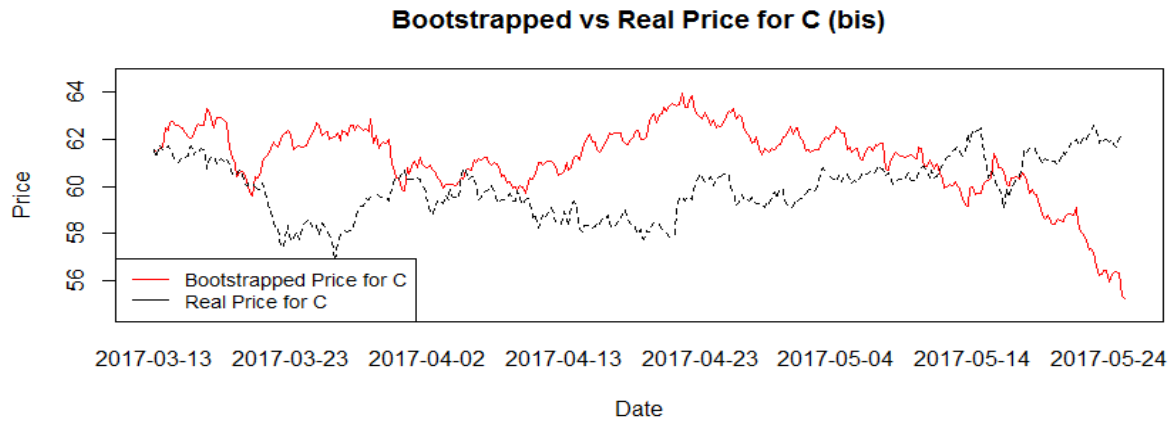

Bootstrapped vs Real Price for PFE
Price
30
31
32
33
34
35
Bootstrapped Price for PFE
Real Price for PFE
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24

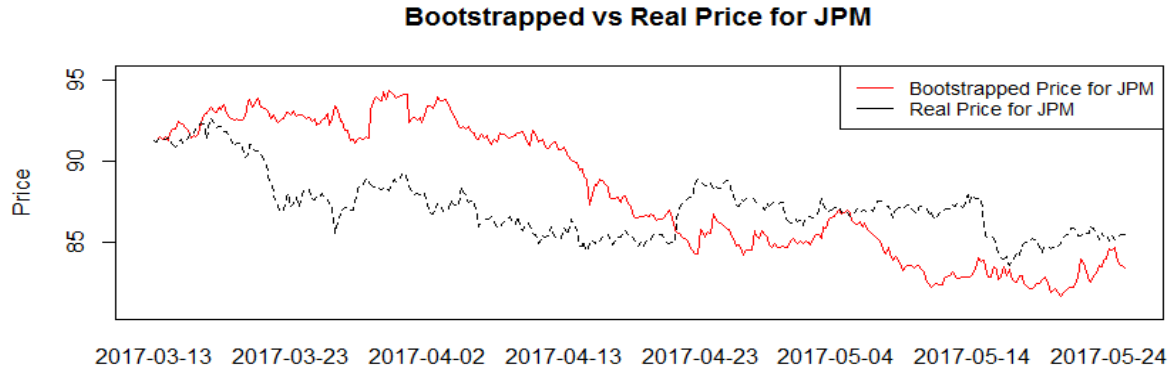

Bootstrapped vs Real Price for PFE (bis)
Price
32.0
33.0
34.0
35.0
Bootstrapped Price for PFE
Real Price for PFE
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24
Date

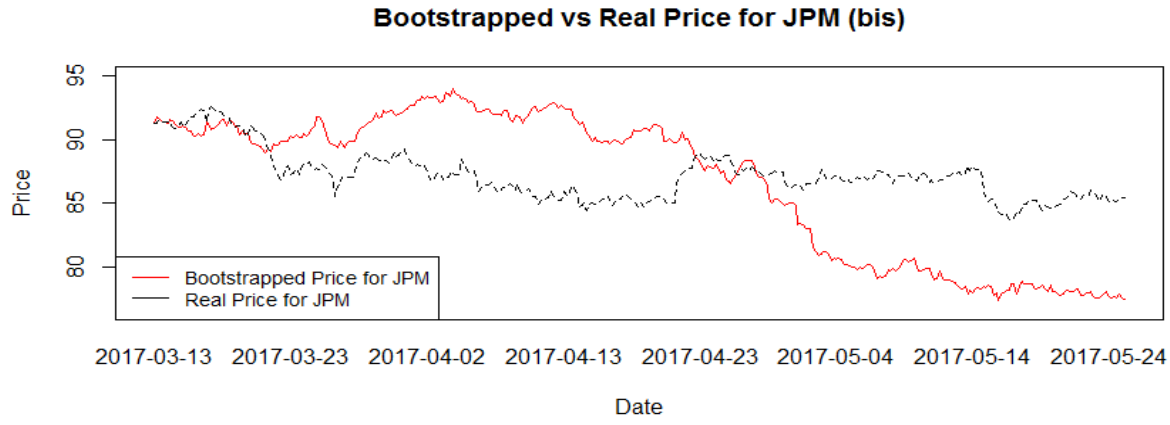

Bootstrapped vs Real Price for F
Price
10.5
11.5
12.5
Bootstrapped Price for F
Real Price for F
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24

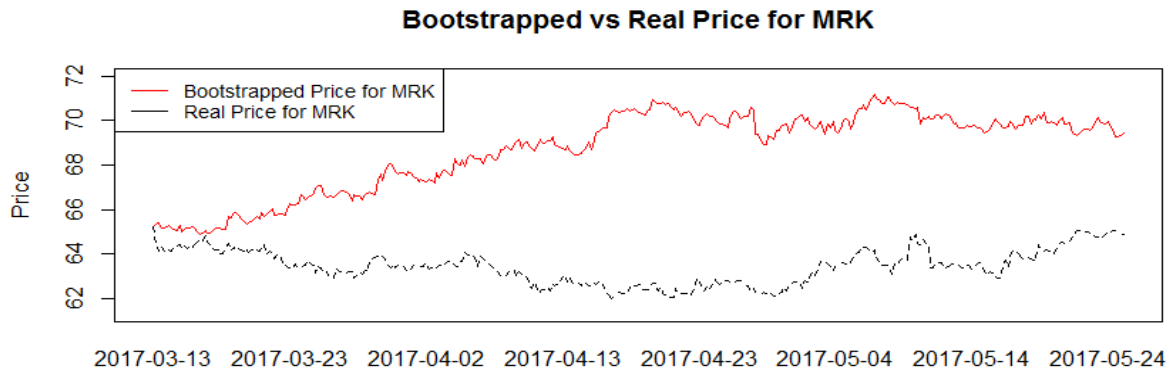

Bootstrapped vs Real Price for F (bis)
Price
9.5
10.5
11.5
12.5
Bootstrapped Price for F
Real Price for F
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24
Date

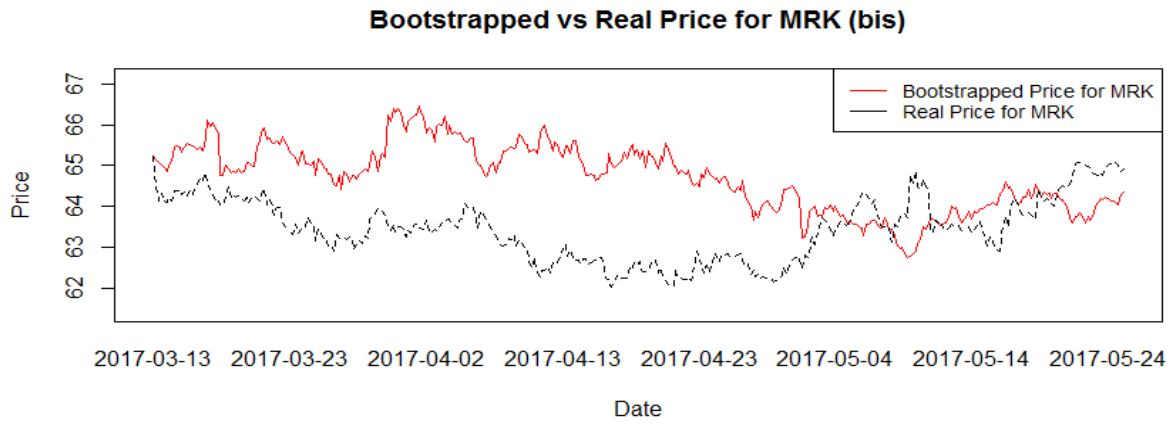

Bootstrapped vs Real Price for C
Price
64
62
60
58
56
Bootstrapped Price for C
Real Price for C
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24

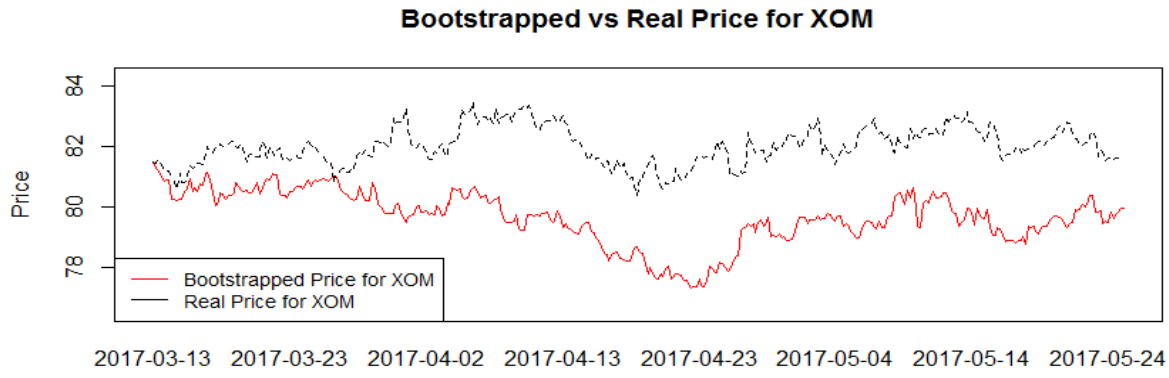

Bootstrapped vs Real Price for C (bis)
Price
64
62
60
58
56
Bootstrapped Price for C
Real Price for C
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24
Date

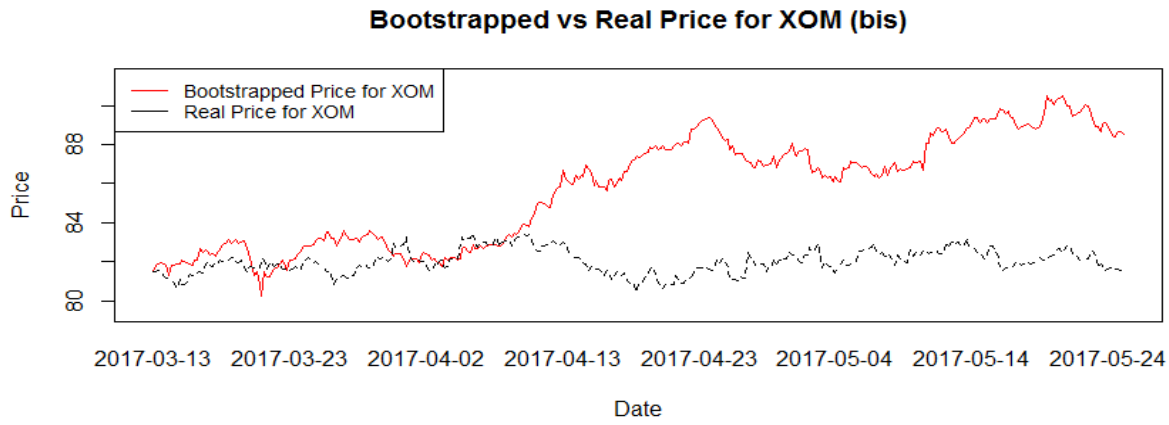

Bootstrapped vs Real Price for JPM
Price
95
90
85
Bootstrapped Price for JPM
Real Price for JPM
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24

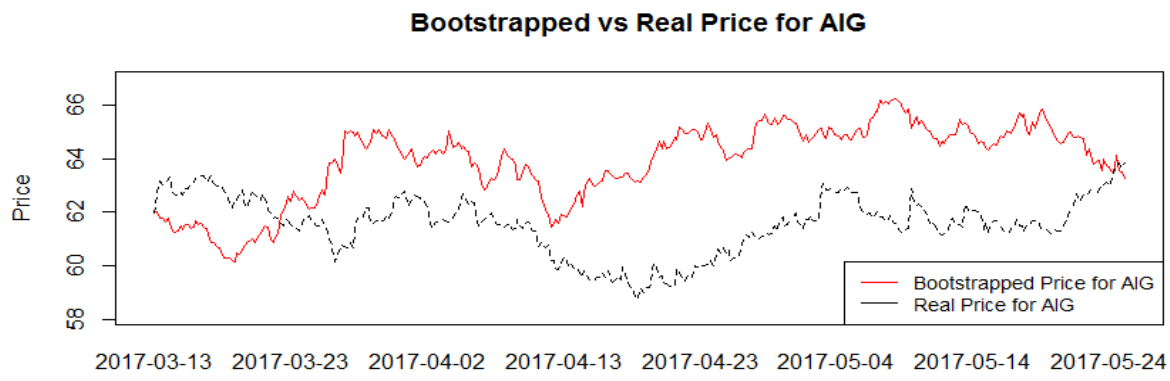

Bootstrapped vs Real Price for JPM (bis)
Price
95
90
85
80
Bootstrapped Price for JPM
Real Price for JPM
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24
Date

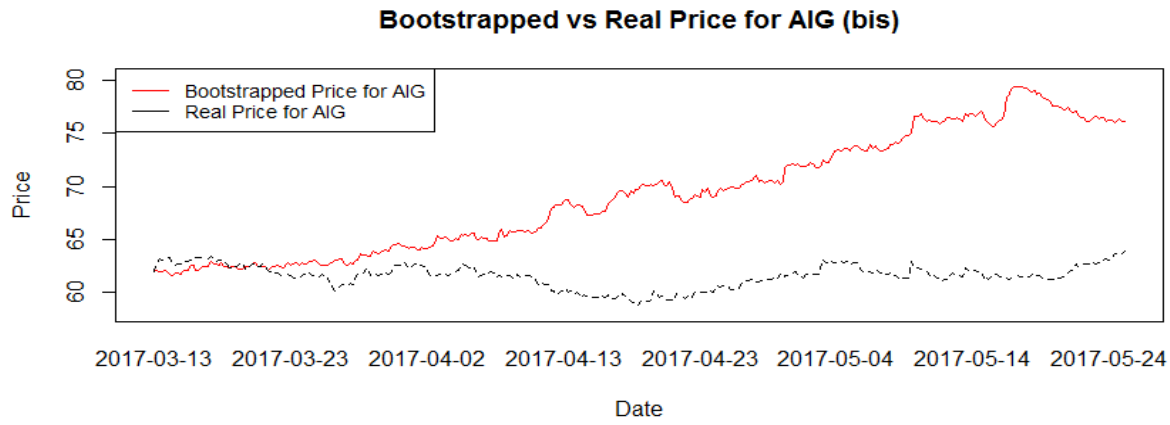

Bootstrapped vs Real Price for MRK
Bootstrapped Price for MRK
Real Price for MRK
Price
62
64
66
68
70
72
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24

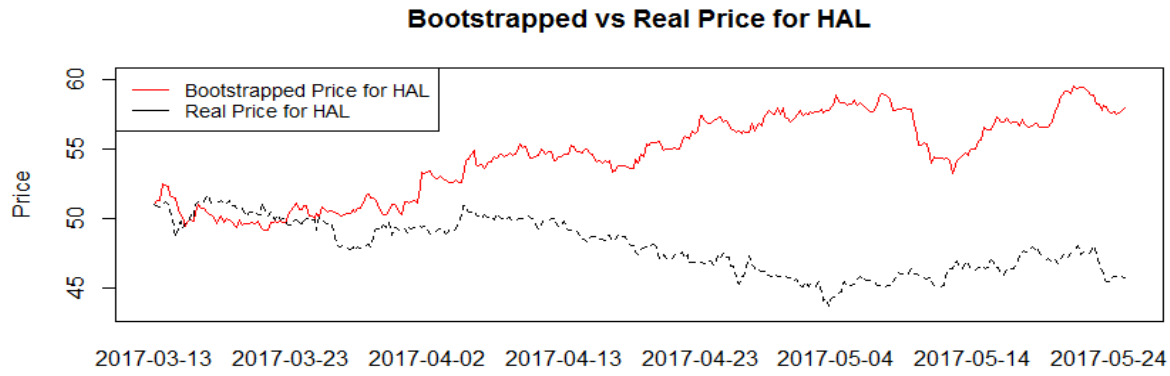

Bootstrapped vs Real Price for MRK (bis)
Bootstrapped Price for MRK
Real Price for MRK
Price
62
63
64
65
66
67
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24
Date

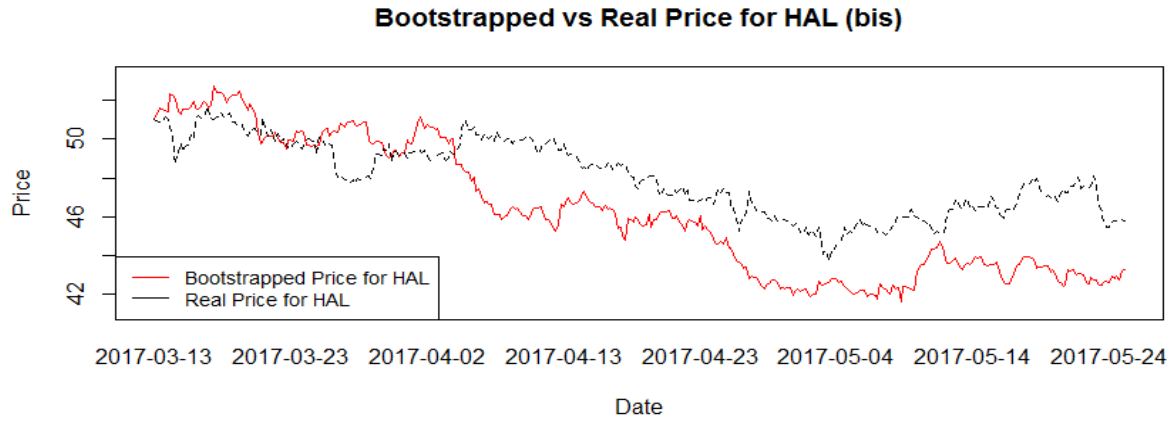

Bootstrapped vs Real Price for XOM
Bootstrapped Price for XOM
Real Price for XOM
Price
78
80
82
84
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24

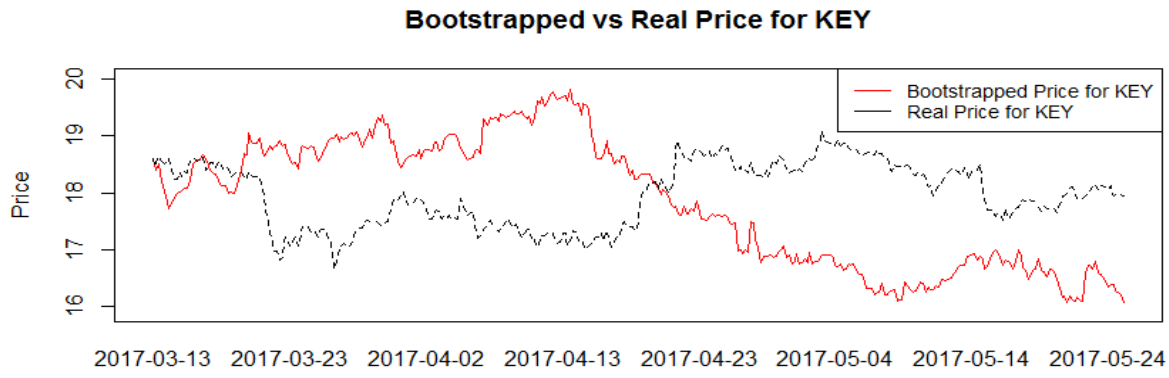

Bootstrapped vs Real Price for XOM (bis)
Bootstrapped Price for XOM
Real Price for XOM
Price
80
84
88
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24
Date

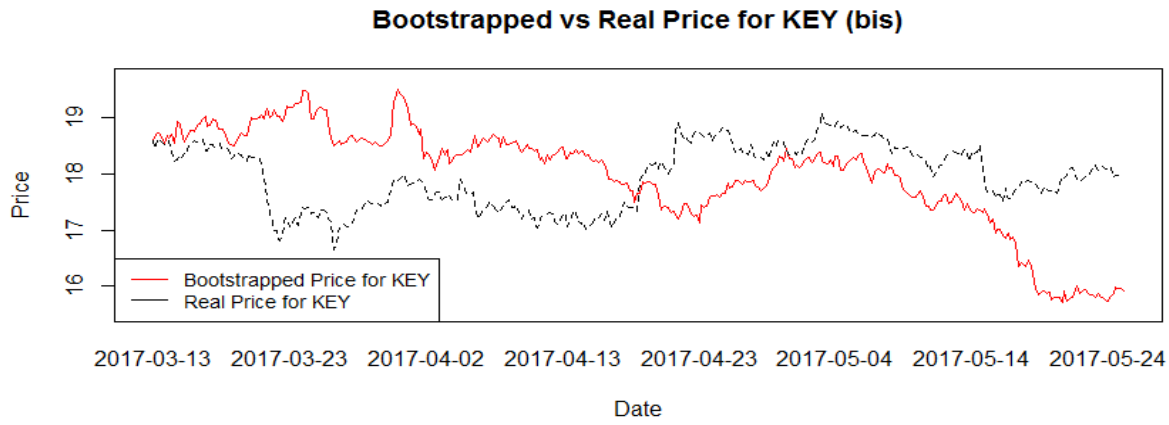

Bootstrapped vs Real Price for AIG
Price
58
60
62
64
66
Bootstrapped Price for AIG
Real Price for AIG
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24

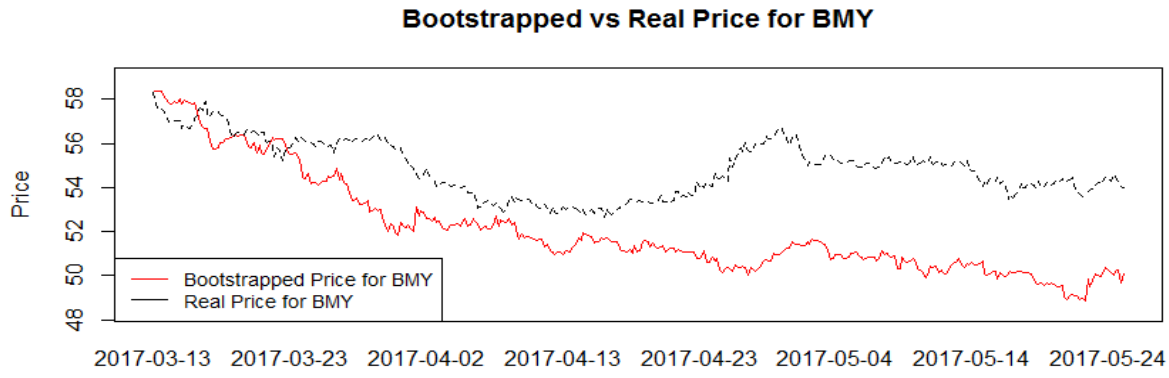

Bootstrapped vs Real Price for AIG (bis)
Price
60
65
70
75
80
Bootstrapped Price for AIG
Real Price for AIG
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24
Date

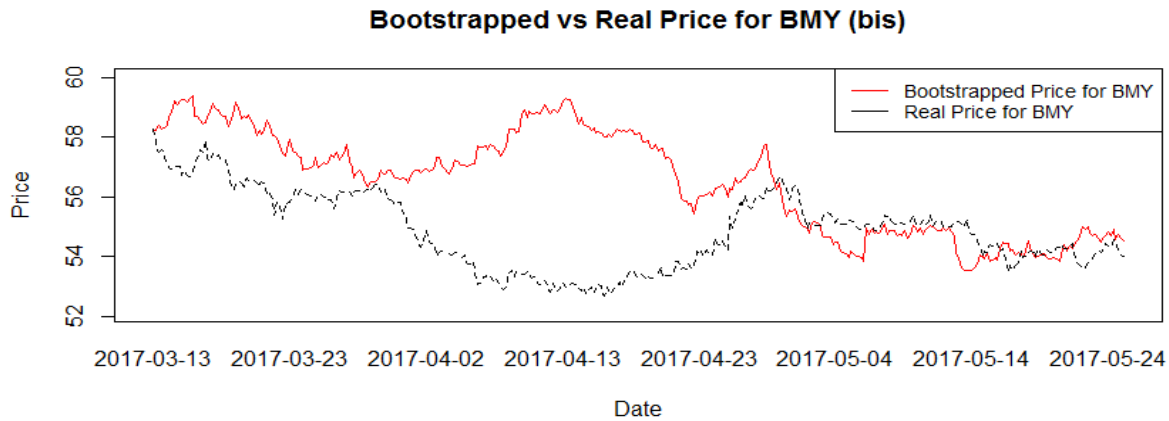

Bootstrapped vs Real Price for HAL
Price
45
50
55
60
Bootstrapped Price for HAL
Real Price for HAL
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24


Bootstrapped vs Real Price for HAL (bis)
Price
42
46
50
Bootstrapped Price for HAL
Real Price for HAL
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24
Date


Bootstrapped vs Real Price for KEY
Bootstrapped Price for KEY
Real Price for KEY
Price
16
17
18
19
20
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24


Bootstrapped vs Real Price for KEY (bis)
Bootstrapped Price for KEY
Real Price for KEY
Price
16
17
18
19
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24
Date


Bootstrapped vs Real Price for BMY
Bootstrapped Price for BMY
Real Price for BMY
Price
48
50
52
54
56
58
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24


Bootstrapped vs Real Price for BMY (bis)
Bootstrapped Price for BMY
Real Price for BMY
Price
52
54
56
58
60
2017-03-13
2017-03-23
2017-04-02
2017-04-13
2017-04-23
2017-05-04
2017-05-14
2017-05-24
Date

Appendix C

R Code

Here we show the structure of our code, and display the most relevant parts and functions that we have created in the development of this Thesis. Non-fundamental parts of the code, such as loading and cleaning data, have been abridged or omitted altogether in this appendix.

For those reasons, the code presented here is not meant to be fully reproducible, but just to give some insight into how computations were done, and to present the new functions developed during this Thesis.

The full code, which is over 2,200 lines long, is available upon request.

C.1 Preliminaries

```
#########################################################################
#          Code 0. Preliminaries: Data loading, computing Gaps          #
#                          Author: Albert Dorador                       #
#########################################################################

# For computation speed gains
library(compiler) ; enableJIT(3) # Highest level of JIT compilation
setCompilerOptions(suppressAll = TRUE)

# NYSE opens from 9:30 to 16:00, so we consider 6 (full) hours
# of trading a day, so 7 prices
n.tot = 7*252 ; n.gap = 251
n.notgap = n.tot - n.gap ; gaps_ind = seq(8,(7*n.gap+1),7)

# Function to compute empirical Gaps
gaps = function(adj.close, close, open){
  if (length(close) != length(open))
    stop("Number of opening prices do not match number of closing prices")
  n = length(close)
```

```
  #normalize opening prices
  k = adj.close/close
  adj.open = k * open

  adj.close = adj.close[-n] #remove last closing price
  adj.open = adj.open[-1] #remove first opening price

  return(1+(adj.open - adj.close)/adj.close)
}
```

C.2 Modeling overnight gaps

```
##############################################################################
#                    Code 1. Modeling overnight gaps                         #
#                        Author: Albert Dorador                              #
##############################################################################

# Range Heuristic
zeta.GED = function(n, mu, std, shape, reps, parallel = FALSE){
  if (!parallel){
    require(fGarch)
    zetas = numeric(reps)
    for(i in 1:reps){
      x = rged(n, mu, std, shape)
      Range = max(x)-min(x)
      zetas[i] = Range/std
    }
  } else {
    require(doParallel)
    num_cores = detectCores()-1
    cl = makeCluster(num_cores, type="PSOCK")
    registerDoParallel(cl)
    zetas = foreach (i = 1:reps, .combine='c', .inorder=FALSE,
                     .packages='fGarch') %dopar% {
                       x = rged(n, mu, std, shape)
                       Range = max(x)-min(x)
                       return(Range/std)
                     }
    stopCluster(cl); gc()
  }
  return(zetas) #in any case
}


# Find volatility (Relative Range) as a function of gap magnitude

library(analytics)

gap_effect_mat = matrix(nrow = num_stocks, ncol = 4)
for (j in 1:num_stocks){
  HL = as.matrix(read.xlsx('data/TFM_-_NYSE_Most_traded_stocks.xlsx',
                  sheet = j, rows = 3:6806, cols = 3:4, colNames = FALSE))
```

```
  RelRange = (HL[,1]-HL[,2])/HL[,2]
  gm = abs(1-gap_data[j,])*100
  mod = lm(RelRange ~ gm + I(gm^2) + I(gm^3))
  outlier_treatment = offliers(HL, mod, DFB = TRUE, pctg = 0.1)
  new.HL = outlier_treatment$new.dataset
  outliers = outlier_treatment$'Final outliers identified'
  new.RelRange = (new.HL[,1]-new.HL[,2])/new.HL[,2]
  new.gm = gm[-c(outliers)]
  mod2 = lm(new.RelRange ~ new.gm + I(new.gm^2) + I(new.gm^3))
  avgRelRange = mean((new.HL[,1]-new.HL[,2])/new.HL[,2])
  gap_effect_mat[j,] = mod2$coefficients/avgRelRange
}
rel.beta0 = colMeans(gap_effect_mat)[1]
rel.beta1 = colMeans(gap_effect_mat)[2]
rel.beta2 = colMeans(gap_effect_mat)[3]
rel.beta3 = colMeans(gap_effect_mat)[4]


# Weibull Gap parameters estimation
#----------------------------------

# We will explore 7 estimation methods to calibrate the Weibull parameters
Methods = c("MOM", "MLE", "MQ", "MSD", "MR", "MDR", "SCDF")
# Only show code for the first method

# (1) Method of Moments (MOM)

# range of plausible shapes
min_shape = 10
max_shape = 220
x = vgap_data
# bootstraping
Nboot = 1e5

library(doParallel)
num_cores = detectCores()-1
cl = makeCluster(num_cores, type="PSOCK")
registerDoParallel(cl)

sim = foreach(j = 1:Nboot, .combine = "cbind", .multicombine=TRUE) %dopar%
{
  require(rootSolve)

  Xboot = sample(x, replace=TRUE)
  # find shape
  rt = 1+(sd(Xboot)/mean(Xboot))^2
  rootFct = function(k) {
    gamma(1+2/k)/gamma(1+1/k)^2 - rt
  }
  shape_est = uniroot.all(rootFct, c(min_shape, max_shape))
  if (length(shape_est)!=1) stop("Change the range for shape parameter")
  scale_est = mean(Xboot)/gamma(1+1/shape_est)
  return(c(shape=shape_est, scale=scale_est))
```

```
}
stopCluster(cl); gc() #forces activation of garbage collector (free up RAM)
result_MOM = apply(sim, 1, function(x)
  c(est=mean(x), se=sd(x), quantile(x, c(0.025, 0.5, 0.975))))
result_MOM
shape_est_vec[1] = result_MOM[1,1]
scale_est_vec[1] = result_MOM[1,2]

# Kolmogorov-Smirnov's D statistic #
Dvec[1] = ks.test(vgap_data,"pweibull",shape_est_vec[1],
                  scale_est_vec[1])$statistic

# MSE #
n = length(x)
ECDF.hat = 1-exp(-((x/scale_est_vec[1])^shape_est_vec[1]))
ECDF = ecdf(x)
MSEvec[1] = sum((ECDF.hat-ECDF(x))^2)/n

Stressvec[1] = 0.3*Dvec[1]+0.7*MSEvec[1]

# (2) Maximum Likelihood (MLE)

# (3) Method 2 in Justus et al. 1977: Median and Quartiles (MQ)

# (4) Method 3 in Justus et al. 1977: Mean and SD (MSD)

# (5) Mean Rank

# (6) Median Rank

# (7) Symmetric CDF

# And the best method is:
m_star = which.min(Stressvec)
Methods[m_star]
(ranked_Stressvec = Methods[order(Stressvec)]) # full ranking
(shape_est = shape_est_vec[m_star])
(scale_est = scale_est_vec[m_star])

# Now that we have an idea about the magnitude of the optimal parameters
# let's zoom in and find even better ones
shapes = seq(shape_est-90, shape_est+10, 0.1)
scales = seq(scale_est-0.01, scale_est+0.02, 0.0005)


# Fine tuning Weibull parameters

library(beepr) ; library(doParallel)
num_cores = detectCores()-1
cl = makeCluster(num_cores, type="PSOCK")
registerDoParallel(cl)

ECDF = ecdf(x)
```

```
res = foreach(sc = scales, .combine = "cbind") %:% # nesting operator
  foreach (sh = shapes, .combine='c') %dopar% {
    D = ks.test(vgap_data,"pweibull",sh,sc)$statistic
    ECDF.hat = 1-exp(-((x/sc)^sh))
    MSE = sum((ECDF.hat-ECDF(x))^2)/n
    Stressmat = 0.3*D+0.7*MSE
    return(Stressmat)
  }
beep()

stopCluster(cl); gc()

rownames(res) = 1:length(shapes)
(Stress_optim = min(res))
star = which(res == min(res), arr.ind = TRUE)
sh.star = shapes[star[1]]
sc.star = scales[star[2]]


# Final correction: Survivor bias

# criteria:
# 1) We require a Stress at most x% bigger than optimal
# 2) If 1 is met, select the smallest shape parameter that satisfies:
#       mean(w_vec)>=1 and more severe gap than observed

mgap = mean(gap_data)
found = FALSE
thres = seq(1.05,2,0.05)
t = 1
while(t <= length(thres) && !found){
  sh = 1
  while(sh <= nrow(res) && !found){
    sc = 1
    while(sc <= ncol(res) && !found){
      if (res[sh,sc] < Stress_optim*thres[t]){
        w_vec=rweibull(1e6,shape=shapes[sh],scale=scales[sc])
        m = mean(w_vec)
        if(m >= 1 && m < mgap){
          new_sh.star = shapes[sh]
          new_sc.star = scales[sc]
          found = TRUE
        }
      }
      sc = sc + 1
    }
    sh = sh + 1
  }
  t = t + 1
}
found #check it has been found
thres[t-1] #50% increment is acceptable as initial stress was extremely low
new_sh.star #final shape parameter: 170.7193
```

```
new_sc.star #final scale parameter: 1.003302
res[sh-1, sc-1] #Final stress: 0.03627818
```

C.3 Model 1: Modeling sigma and white noise

```
#############################################################################
#             Code 2. MODEL 1: Modeling sigma and white noise               #
#                          Author: Albert Dorador                           #
#############################################################################

# Because we'll consider the average GARCH(p,q) for the 30 stocks,
# we must agree on p and q beforehand
# Considering that higher p and q may not be statistically
# significant for all stocks, we choose p=1,q=1

# Fit best GARCH (1,1) for each stock
library(fGarch)

GARCH1.1_AICc_vec = numeric(num_stocks)
distribs = c("norm", "snorm", "ged", "sged", "std", "sstd")
GARCH1.1_mean_AICc_vec = numeric(length(distribs))
k_vec = integer(length(distribs))
n = nrow(h_rets) #69
for (d in 1:length(distribs)){
  for (j in 1:num_stocks){
    tryCatch({ #ignore errors in for loop
      GARCH1.1 = garchFit(formula=~garch(1,1),data=h_rets[,j],
                          cond.dist=distribs[d],trace=FALSE)
    }, error=function(e){})
    negLL = GARCH1.1@fit$llh
    k = length(GARCH1.1@fit$par)
    k_vec[d] = k
    AIC = 2*negLL + 2*k
    GARCH1.1_AICc_vec[j] = AIC + 2*k*(k+1)/(n-k-1)
  }
  GARCH1.1_mean_AICc_vec[d] = mean(GARCH1.1_AICc_vec)
}

# And the best distribution is:
d_star = which.min(GARCH1.1_mean_AICc_vec) #4
distribs[d_star]
(ranked_distribs = distribs[order(GARCH1.1_mean_AICc_vec)]) # full ranking

#best results obtained if epsilon ~ sged, so estimate again using sged:
k_star = k_vec[d_star]
GARCH1.1_par_mat = matrix(nrow = num_stocks, ncol = (k_star-1))
for (j in 1:num_stocks){
  tryCatch({ #ignore errors in for loop
    GARCH1.1 = garchFit(formula=~garch(1,1),data=h_rets[,j],
                        cond.dist=distribs[d_star],trace=FALSE)
  }, error=function(e){})
  GARCH1.1_par_mat[j,] = GARCH1.1@fit$par[2:k_star]
```

```
}
omega = colMeans(GARCH1.1_par_mat)[1]
alpha = colMeans(GARCH1.1_par_mat)[2]
beta = colMeans(GARCH1.1_par_mat)[3]
skew = colMeans(GARCH1.1_par_mat)[4] #for the innovations, aka white noise
shape = colMeans(GARCH1.1_par_mat)[5] #for the innovations, aka white noise
```

C.4 Model 1: Return distribution estimation

```
##############################################################################
#           Code 3. MODEL 1: Return distribution estimation                  #
#                          Author: Albert Dorador                            #
##############################################################################

# Step 1: Determine best distribution (only shown for best distribution)
#a) Student-T

#b) Skewed Student-T

#c) Generalized Error Distribution
gedAICc = numeric(num_stocks)
k = 3
for (j in 1:num_stocks){
  negLL = gedFit(h_rets[,j])$objective
  AIC = 2*negLL + 2*k
  gedAICc[j] = AIC + 2*k*(k+1)/(n-k-1)
}

#d) Skewed Generalized Error Distribution
ged_par_mat = matrix(nrow = num_stocks, ncol = 3)
for (j in 1:num_stocks){
  ged_par_mat[j,] = gedFit(h_rets[,j])$par
}
mu.0 = colMeans(ged_par_mat)[1] # but we'll use NYSE's historical rate
sdv.0 = colMeans(ged_par_mat)[2]
nu.0 = colMeans(ged_par_mat)[3] #1.3; Recall: Gaussian (n=2), Laplace (n=1)

# Step 2: Refine estimation, using MLE and grid search to reduce Stress

# MLE #
ged_minusLL = function(sd, nu) {
  R = dged(x, mean = mu, sd, nu)
  -sum(log(R))
}

library(stats4)
methods = c("Nelder-Mead", "BFGS", "CG", "SANN")
k = 2
ged_params_mat = matrix(nrow = length(methods), ncol = k)
GED_AICc = numeric(4)
ged_optim_params_mat = matrix(nrow = num_stocks, ncol = k)
m_star_vec = character(num_stocks)
```

```
for (j in 1:num_stocks){
  x = h_rets[,j]
  counter = 1
  for (m in methods){
    tryCatch({ #ignore errors in for loop
      res = mle(ged_minusLL, start = list(sd = sdv.0, nu = nu.0),
                method = m)@coef
      negLL = res@min
      AIC = 2*negLL + 2*k
      GED_AICc[counter] = AIC + 2*k*(k+1)/(n-k-1)
    }, error=function(e){})
    ged_params_mat[counter,] = res
    counter = counter + 1
  }
  m_star = which.min(GED_AICc) #use best estimation method for each stock
  m_star_vec[j] = methods[m_star]
  sdev = ged_params_mat[m_star,1]
  Nu = ged_params_mat[m_star,2]
  ged_optim_params_mat[j,] = c(sdev,Nu)
}
m_star_vec #Nelder-Mead in all cases!
sdev = colMeans(ged_optim_params_mat)[1]#~ identical to uncond_vol
sdev2 = sdev^2
Nu = colMeans(ged_optim_params_mat)[2]

# Let's fine tune the parameters

sdevs = seq(sdev-0.003, sdev+0.004, 0.00001)
nus = seq(Nu-0.8, Nu+0.9, 0.005)

library(doParallel)
num_cores = detectCores()-1
cl = makeCluster(num_cores, type="PSOCK")
registerDoParallel(cl)

n = length(rets)
ECDF = ecdf(rets)
res = foreach(sdv = sdevs, .combine = "cbind",
              .packages=c("fGarch","zipfR")) %:% # nesting operator
  foreach (nu = nus, .combine='c', .packages=c("fGarch","zipfR")) %dopar% {
    D = ks.test(rets,"pged", mean = mu, sd = sdv, nu = nu)$statistic
    ECDF.hat = 0.5 + sign(rets-mu)*(Igamma(1/nu, (abs(rets-mu)/sdv)^nu, lower=TRUE)/
                                      (2*gamma(1/nu)))
    MSE = sum((ECDF.hat-ECDF(rets))^2)/n
    Stress = 0.3*D+0.7*MSE
    return(Stress)
  }
stopCluster(cl); gc()

rownames(res) = 1:length(nus)
(Stress_optim = min(res))
star = which(res == min(res), arr.ind = TRUE)
nu.star = nus[star[1]]
```

```
sdv.star = sdevs[star[2]]

# K-S D statistic #
D = ks.test(rets,"pged", mean = mu, sd = sdv.star, nu = nu.star)$statistic

# MSE #
library(zipfR)
n = length(rets)
ECDF.hat=0.5+sign(rets-mu)*(Igamma(1/nu.star,
                                   (abs(rets-mu)/sdv.star)^nu.star, lower=TRUE)/
                            (2*gamma(1/nu.star)))
ECDF = ecdf(rets)
MSE = sum((ECDF.hat-ECDF(rets))^2)/n

(Stress = 0.3*D + 0.7*MSE)
```

C.5 Model 2: Return (ARMA) and volatility (GARCH) estimation

```
##########################################################################
#  Code 4. MODEL 2: Return (ARMA) and volatility (GARCH) estimation #
#                          Author: Albert Dorador                        #
##########################################################################

# We fit the best ARMA(3,0)+GARCH(1,1) for each stock
library(fGarch)

ArmaGarch1.1_AICc_vec = numeric(num_stocks)
distribs = c("norm", "snorm", "ged", "sged", "std", "sstd")
ArmaGarch1.1_mean_AICc_vec = numeric(length(distribs))
k_vec = integer(length(distribs))
n = nrow(h_rets) #69
for (d in 1:length(distribs)){
  for (j in 1:num_stocks){
    tryCatch({ #ignore errors in for loop
      ArmaGarch1.1 = garchFit(formula=~arma(3,0)+garch(1,1),
                              data=h_rets[,j],cond.dist=distribs[d],
                              trace=FALSE)
    }, error=function(e){})
    negLL = ArmaGarch1.1@fit$llh
    k = length(ArmaGarch1.1@fit$par)
    k_vec[d] = k
    AIC = 2*negLL + 2*k
    ArmaGarch1.1_AICc_vec[j] = AIC + 2*k*(k+1)/(n-k-1)
  }
  ArmaGarch1.1_mean_AICc_vec[d] = mean(ArmaGarch1.1_AICc_vec)
}

# And the best distribution is:
d_star = which.min(ArmaGarch1.1_mean_AICc_vec)
distribs[d_star]
(ranked_distribs = distribs[order(ArmaGarch1.1_mean_AICc_vec)]) # full ranking
```

```
#best results obtained if epsilon ~ ged, so estimate again using ged:
k_star = k_vec[d_star]
ArmaGarch1.1_par_mat = matrix(nrow = num_stocks, ncol = (k_star-1))

for (j in 1:num_stocks){
  tryCatch({
    ArmaGarch1.1 = garchFit(formula=~arma(3,0)+garch(1,1),data=h_rets[,j],
                            cond.dist=distribs[d_star],trace=FALSE)
  }, error=function(e){})
  ArmaGarch1.1_par_mat[j,] = ArmaGarch1.1@fit$par[2:k_star]
}
ArmaGarch1.1_par_mat
ar1 = colMeans(ArmaGarch1.1_par_mat)[1] #ARMA
ar2 = colMeans(ArmaGarch1.1_par_mat)[2] #ARMA
ar3 = colMeans(ArmaGarch1.1_par_mat)[3] #ARMA
omega = colMeans(ArmaGarch1.1_par_mat)[4] #GARCH
alpha1 = colMeans(ArmaGarch1.1_par_mat)[5] #GARCH
beta1 = colMeans(ArmaGarch1.1_par_mat)[6] #GARCH
shape = colMeans(ArmaGarch1.1_par_mat)[7] #for the distrib of the epsilon
m = mu*(1-ar1-ar2-ar3) #Tsay, pg 39
```

C.6 Model 3: Truncated Geometric Sationary Bootstrap

```
##########################################################################
#      Code 5. MODEL 3: Truncated Geometric Sationary Bootstrap          #
#                          Author: Albert Dorador                        #
##########################################################################

# Testing weak stationarity (and weak dependence) #

# Function to determine if a time series is weakly stationary (4 criteria)
weakly.stationary = function(tseries, signific=0.05, ticker=NULL, res.df=FALSE){
  n.stocks = ncol(tseries)
  #preallocate memory
  results.df = data.frame(PP=logical(n.stocks), ADF=logical(n.stocks),
                          KPSS=logical(n.stocks), PSR=logical(n.stocks))

  for(j in 1:n.stocks){
    ret.j = tseries[,j]

    require(urca)
    # (1) Phillips-Perron with trend; if trend, stop!
    a = summary(ur.pp(ret.j,type="Z-tau",model="trend"))
    trend.pval = attributes(a)$testreg[4]$coefficients[12]
    if(trend.pval < signific){
      results.df[j,] = data.frame(PP=FALSE,ADF=FALSE,KPSS=FALSE,PSR=FALSE)
    }else{
      z.tau = attributes(a)$teststat[1]
      pp.crit.val = ifelse(signific==0.05, attributes(a)$cval[2],
                           attributes(a)$cval[1])
      PP = (z.tau < pp.crit.val)
```

```
      # (2) Augmented Dickey-Fuller
      require(car) #for DW test
      found = FALSE
      lag.order = 0
      while(lag.order <= 20 && !found){
        ADF.info = ur.df(ret.j,lags=lag.order)
        ADF.res = as.vector(ADF.info@res)
        DW = durbinWatsonTest(ADF.res)
        if(DW > 1.85){ #Verbeek 2004, p.185; don't reject H0
          found = TRUE #stop
        }else{
          lag.order = lag.order + 1
        }
      }
      b = summary(ur.df(ret.j,lags=lag.order))
      adf.stat = attributes(b)$teststat[1]
      adf.crit.val = ifelse(signific==0.05, attributes(b)$cval[2],
                            attributes(b)$cval[1])
      ADF = (adf.stat < adf.crit.val)

      # (3) KPSS
      c = summary(ur.kpss(ret.j,type="mu",use.lag=lag.order))
      kpss.stat = attributes(c)$teststat[1]
      kpss.crit.val = ifelse(signific==0.05, attributes(c)$cval[2],
                             attributes(c)$cval[4])
      KPSS = (kpss.stat < kpss.crit.val)

      # (4) PSR
      require(fractal)
      sink("NUL") #starts suppressing output until sink() is called
      PSR.info = stationarity(ret.j, significance=signific) #Ho: stat.
      sink()
      PSR = attributes(PSR.info)$stationarity[[1]]

      results.df[j,] = data.frame(PP=PP, ADF=ADF, KPSS=KPSS, PSR=PSR)
    }
  }
  if(res.df){
    return(results.df)
  }else{
    w.stat_stocks = ticker[apply(results.df, 1, all)]
    return(w.stat_stocks)
  }
}
(w.s = weakly.stationary(h_rets,ticker=tickers,signific=0.01)) # 11 are w.s


# Truncated Geometric Stationary Bootstrap algorithm

SBoot_trunc.geom = function(tseries){
  N = length(tseries)
  require(np)
```

```
  b = b.star(tseries, round=TRUE)[1]
  f = function(q){
    (N-b)*q^(N+1) + (b-1-N)*q^N + (1+b)*q + 1 - b
  }
  q = uniroot(f,c(0,1))$root
  p = 1-q
  names(tseries) = seq_along(tseries) #create named vector to recover index
  x_boot = numeric(N)

  x_boot.i = sample(x=tseries, size=1, replace=TRUE)
  n = as.integer(names(x_boot.i)) #index in original data
  x_boot[1] = x_boot.i

  for (i in 2:N){
    if(n==N){  # a)
      step4 = sample(c(0,1),size=1,replace=TRUE, prob=c(p,(1-p)))
      if(step4==0){
        x_boot.i = sample(x=tseries, size=1, replace=TRUE)
        n = as.integer(names(x_boot.i))
      }else{
        x_boot.i = tseries[1]
        n = 1
      }
      x_boot[i] = x_boot.i
    }else{  # b)
      step4 = sample(c(0,1),size=1,replace=TRUE, prob=c(p,(1-p)))
      if(step4==0){
        x_boot.i = sample(x=tseries, size=1, replace=TRUE)
        n = as.integer(names(x_boot.i))
      }else{
        x_boot.i = tseries[n+1]
        n = n + 1
      }
      x_boot[i] = x_boot.i
    }
  }
  return(x_boot)
}

# Wrapper to compute bootstrapped prices, given bootstrapped returns
SB_trunc.geom.Prices = function(boot_rets, h_prices, ticker){
  N = length(boot_rets)+1
  stock_boot.prices = numeric(N)
  stock_boot.prices[1] = h_prices[,colnames(h_prices)==ticker][[1]]
  for (i in 2:N){
    stock_boot.prices[i] = stock_boot.prices[i-1]*(1+boot_rets[i-1])
  }
  if (any(stock_boot.prices <= 0)) { #if stock price reaches 0, stays there
    firstneg = which(stock_boot.prices <= 0)[1]
    stock_boot.prices[firstneg:length(stock_boot.prices)] = 0
  }
  return(stock_boot.prices)
}
```

C.7 Simulation: Finding best parameters for fixed percentage SL-SP

```
#########################################################################
#     Code 6. SIMULATION: Finding best parameters for fixed % SL-SP     #
#                          Author: Albert Dorador                       #
#########################################################################

alphas = seq(from = 0.03, to = 0.1, by = 0.01)
betas = c(seq(from = 0.04, to = 0.15, by = 0.01), 1e6) #includes case of no SP

# Possible criteria: "ExpRet", "Sharpe", "Sortino", "R-VaR", "R-ES"

# Function to optimize parameters for the fixed % rule,
# given a model and performance criterion
pctg.barrier.param.optimizer = function(model, alphas, betas, criterion,
                                        ticker=NULL, reps = 1e4){
  require(doParallel)
  num_cores = detectCores()-1
  cl = makeCluster(num_cores, type="PSOCK")
  registerDoParallel(cl)

  if (model == 1) {
    sigma_0 = sdv.star
    sigma2_0 = sigma_0^2
    performance = foreach(a = alphas, .combine = "cbind",
                          .packages = c("doParallel", "PerformanceAnalytics",
                                        "fGarch"),
                          .export = ls(.GlobalEnv)) %:%
      foreach (b = betas, .combine='c', .packages = c("PerformanceAnalytics",
                                                      "fGarch")) %dopar% {
        oper = foreach(j = 1:reps, .combine = "rbind",
                       .multicombine=TRUE) %dopar%
        {
          gaps = rweibull(n.gap,shape=new_sh.star,scale=new_sc.star)
          P_0 = sample.int(200,1) #first price
          epsilon2_vector = numeric(n.tot)
          epsilon2_vector[1] = (rsged(1, mean = 0, sd = 1, nu = shape,
                                      xi = skew))^2
          Pvector = numeric(n.tot)
          Pvector[1] = P_0
          no_gap_multiplier = rel.beta0^2 #because SD*k --> Var*k^2
          sigma2_vector = numeric(n.tot)
          sigma2_vector[1] = sigma2_0

          # Decide which days will have a gap:
          confirmed_gaps_ind = sample(c(0,1),n.gap,replace = TRUE,
                                      prob = c(0.2207,0.7793))
                                      #77.93% chance of a gap
          confirmed_gaps = gaps[which(confirmed_gaps_ind==1)]
          confirmed_gaps_ind = gaps_ind[which(confirmed_gaps_ind==1)]
```

```
# Generate prices:
counter = 0
for(t in 2:n.tot){
  if (!(t %in% gaps_ind)){ #not a gappable day
    epsilon2_vector[t] = (rsged(1, mean = 0, sd = 1, nu = shape,
                                xi = skew))^2
    sigma2_vector[t] = (omega+alpha*epsilon2_vector[t-1]*
                          sigma2_vector[t-1]+beta*
                          sigma2_vector[t-1])*no_gap_multiplier
    r = rged(1, mean = mu, sd = sqrt(sigma2_vector[t]),
             nu = nu.star)
    Pvector[t] = Pvector[t-1]*(1+r)
  }else{ #a gappable day
    if(t %in% confirmed_gaps_ind){ #truly a gap day!
      counter = counter + 1
      epsilon2_vector[t] = (rsged(1, mean = 0, sd = 1, nu = shape,
                                  xi = skew))^2
      gm = abs(1-gaps[counter])*100
      gm_multiplier = (rel.beta0 + rel.beta1*gm + rel.beta2*gm^2 +
                         rel.beta3*gm^3)^2
      sigma2_vector[t] = (omega+alpha*epsilon2_vector[t-1]*
                            sigma2_vector[t-1]+beta*
                            sigma2_vector[t-1])*gm_multiplier
      Pvector[t] = Pvector[t-1]*gaps[counter]
    }else{
      no_gap_multiplier = rel.beta0^2
      epsilon2_vector[t] = (rsged(1, mean = 0, sd = 1, nu = shape,
                                  xi = skew))^2
      sigma2_vector[t] = (omega+alpha*epsilon2_vector[t-1]*
                            sigma2_vector[t-1]+
                            beta*sigma2_vector[t-1])*
                            no_gap_multiplier
      Pvector[t] = Pvector[t-1] #keep price the same
    }
  }
}
if (any(Pvector <= 0)){ #if stock price reaches 0, stays there
  firstneg = which(Pvector <= 0)[1]
  Pvector[firstneg:length(Pvector)] = 0
}

# We must check at each hour if we have broken the SL/SP level set
# Higher freq would mean never break
SL = P_0*(1-a)
SP = P_0*(1+b)
t = 2
while(t < n.tot && Pvector[t]>0 && Pvector[t]>SL && Pvector[t]<SP){
  SL = max(Pvector[1:t])*(1-a)
  SP = max(Pvector[1:t])*(1+b)
  t = t+1
}

leaves = t
```

```
  return(c(Pvector[t], Pvector[n.tot], P_0, leaves))
}
P.SLSP = oper[,1]
P.BH = oper[,2]
P.0 = oper[,3]
leaves = oper[,4]
rf = 0.03171
realized_rf = (1-leaves/n.tot)*rf

retSLSP = (P.SLSP*(1+realized_rf)/P.0-1)*100
retBH = (P.BH/P.0-1)*100

# Criterion 1
if (criterion == "ExpRet"){
  Expected.ret.SLSP = mean(retSLSP)
  Expected.ret.BH = mean(retBH)
  ExpRet.diff = Expected.ret.SLSP - Expected.ret.BH
  return(ExpRet.diff)
}

# Criterion 2
if (criterion == "Sharpe"){
  SharpeSLSP = (mean(retSLSP)-rf)/sd(retSLSP)
  SharpeBH = (mean(retBH)-rf)/sd(retBH)
  Sharpe.diff = SharpeSLSP - SharpeBH
  return(Sharpe.diff)
}

# Criterion 3
if (criterion == "Sortino"){
  SOR.SLSP = as.numeric(SortinoRatio(retSLSP, MAR = rf*100))
  SOR.BH = as.numeric(SortinoRatio(retBH, MAR = rf*100))
  SOR.diff = SOR.SLSP - SOR.BH
  return(SOR.diff)
}

# Criterion 4
if (criterion == "R-VaR"){
  md.SLSP = median(retSLSP)
  md.BH = median(retBH)
  RVaR.RatioSLSP = (md.SLSP-rf)/(md.SLSP-quantile(retSLSP,0.05))
  RVaR.RatioBH = (md.BH-rf)/(md.BH-quantile(retBH,0.05))
  RvaR.diff = RVaR.RatioSLSP - RVaR.RatioBH
  return(RvaR.diff)
}

# Criterion 5
if (criterion == "R-ES"){
  md.SLSP = median(retSLSP)
  md.BH = median(retBH)
  RES.RatioSLSP = (md.SLSP-rf)/(md.SLSP-
                  mean(retSLSP[retSLSP<=quantile(retSLSP,0.05)]))
  RES.RatioBH = (md.BH-rf)/(md.BH-
```

```
                    mean(retBH[retBH<=quantile(retBH,0.05)]))
      RES.diff = RES.RatioSLSP - RES.RatioBH
      return(RES.diff)
    }
  }
  stopCluster(cl); gc()
  star = which(performance == max(performance[performance != Inf]),
               arr.ind = TRUE)
  alpha.star.M1 = alphas[star[2]] #0.095 with 1e4 reps
  beta.star.M1 = betas[star[1]] #0.17 with 1e4 reps
  return(c(alpha.star.M1, beta.star.M1))

} else if (model == 2) {
  sigma_0 = sdv.star
  sigma2_0 = sigma_0^2
  performance = foreach(a = alphas, .combine = "cbind",
                        .packages = c("doParallel",
                                      "PerformanceAnalytics", "fGarch"),
                        .export = ls(.GlobalEnv)) %:%
    foreach (b = betas, .combine='c', .packages = c("PerformanceAnalytics",
                                                    "fGarch")) %dopar% {
      oper = foreach(j = 1:reps, .combine = "rbind",
                     .multicombine=TRUE) %dopar%
      {
        gaps = rweibull(n.gap,shape=new_sh.star,scale=new_sc.star)
        P_0 = sample.int(200,1) #first price
        ret_vector = numeric(n.tot)
        epsilon_vector = numeric(n.tot)
        epsilon_vector[1] = rsged(1, mean = 0, sd = 1,
                                  nu = shape, xi = skew)
        epsilon_vector[2] = rsged(1, mean = 0, sd = 1,
                                  nu = shape, xi = skew)
        epsilon_vector[3] = rsged(1, mean = 0, sd = 1,
                                  nu = shape, xi = skew)
        epsilon_vector[4] = rsged(1, mean = 0, sd = 1,
                                  nu = shape, xi = skew)
        Pvector = numeric(n.tot)
        Pvector[1] = P_0
        no_gap_multiplier = rel.beta0^2 #because SD*k --> Var*k^2
        sigma_vector = numeric(n.tot)
        sigma_vector[1] = sigma_0
        sigma_vector[2] = (omega+alpha*(epsilon_vector[1]*
                            sigma_vector[1])^2+
                            beta*sigma_vector[1]^2)*no_gap_multiplier
        sigma_vector[3] = (omega+alpha*(epsilon_vector[2]*
                            sigma_vector[2])^2+
                            beta*sigma_vector[2]^2)*no_gap_multiplier
        sigma_vector[4] = (omega+alpha*(epsilon_vector[3]*
                            sigma_vector[3])^2+
                            beta*sigma_vector[3]^2)*no_gap_multiplier
        # Decide which days will have a gap:
        confirmed_gaps_ind = sample(c(0,1),n.gap,replace = TRUE,
                                    prob = c(0.2207,0.7793))
```

```
confirmed_gaps = gaps[which(confirmed_gaps_ind==1)]
confirmed_gaps_ind = gaps_ind[which(confirmed_gaps_ind==1)]

# Generate prices:
r1 = rged(1, mean = mu, sd = sigma_vector[1], nu = nu.star)
r2 = m + ar1*r1 + (epsilon_vector[2]*sigma_vector[2])^2
r3 = m + ar1*r2+ar2*r1 + (epsilon_vector[3]*sigma_vector[3])^2
ret_vector[1] = r1
ret_vector[2] = r2
ret_vector[3] = r3

Pvector[2] = Pvector[1]*(1+r1)
Pvector[3] = Pvector[2]*(1+r2)
Pvector[4] = Pvector[3]*(1+r3)

ret_vector[4] = m + ar1*r3+ar2*r2+ar3*r1 +
  (epsilon_vector[4]*sigma_vector[4])^2

counter = 0
for(t in 5:n.tot){
  if (!(t %in% gaps_ind)){ #not a gappable day
    epsilon_vector[t] = rsged(1, mean = 0, sd = 1, nu = shape,
                              xi = skew)
    sigma_vector[t] = sqrt((omega+alpha*(epsilon_vector[t-1]*
                     sigma_vector[t-1])^2+
                     beta*sigma_vector[t-1]^2)*no_gap_multiplier)
    ret_vector[t] = m + ar1*ret_vector[t-1]+ar2*ret_vector[t-2]+
                    ar3*ret_vector[t-3] +
                    (epsilon_vector[t]*sigma_vector[t])^2
    Pvector[t] = Pvector[t-1]*(1+ret_vector[t])
  }else{ #a gappable day
    if(t %in% confirmed_gaps_ind){ #truly a gap day!
      counter = counter + 1
      epsilon_vector[t] = rsged(1, mean = 0, sd = 1, nu = shape,
                                xi = skew)
      gm = abs(1-gaps[counter])*100
      gm_multiplier = (rel.beta0 + rel.beta1*gm + rel.beta2*gm^2
                       + rel.beta3*gm^3)^2
      sigma_vector[t] = sqrt((omega+alpha*(epsilon_vector[t-1]*
                       sigma_vector[t-1])^2+beta*sigma_vector[t-1]^2)*
                         gm_multiplier)
      Pvector[t] = Pvector[t-1]*gaps[counter]
    }else{
      no_gap_multiplier = rel.beta0^2
      epsilon_vector[t] = rsged(1, mean = 0, sd = 1, nu = shape,
                                xi = skew)
      sigma_vector[t] = sqrt((omega+alpha*(epsilon_vector[t-1]*
                     sigma_vector[t-1])^2+
                     beta*sigma_vector[t-1]^2)*no_gap_multiplier)
      Pvector[t] = Pvector[t-1] #keep price the same
    }
  }
}
```

```
  if (any(Pvector <= 0)){
    firstneg = which(Pvector <= 0)[1]
    Pvector[firstneg:length(Pvector)] = 0
  }

  # We must check at each hour if we have broken the SL/SP level set
  # Higher freq would mean never break
  SL = P_0*(1-a)
  SP = P_0*(1+b)
  t = 2
  while(t < n.tot && Pvector[t]>0 && Pvector[t]>SL && Pvector[t]<SP){
    SL = max(Pvector[1:t])*(1-a)
    SP = max(Pvector[1:t])*(1+b)
    t = t+1
  }

  leaves = t
  return(c(Pvector[t], Pvector[n.tot], P_0, leaves))
}
P.SLSP = oper[,1]
P.BH = oper[,2]
P.0 = oper[,3]
leaves = oper[,4]
rf = 0.03171
realized_rf = (1-leaves/n.tot)*rf

retSLSP = (P.SLSP*(1+realized_rf)/P.0-1)*100
retBH = (P.BH/P.0-1)*100

# Criterion 1
if (criterion == "ExpRet"){
  Expected.ret.SLSP = mean(retSLSP)
  Expected.ret.BH = mean(retBH)
  ExpRet.diff = Expected.ret.SLSP - Expected.ret.BH
  return(ExpRet.diff)
}

# Criterion 2
if (criterion == "Sharpe"){
  SharpeSLSP = (mean(retSLSP)-rf)/sd(retSLSP)
  SharpeBH = (mean(retBH)-rf)/sd(retBH)
  Sharpe.diff = SharpeSLSP - SharpeBH
  return(Sharpe.diff)
}

# Criterion 3
if (criterion == "Sortino"){
  SOR.SLSP = as.numeric(SortinoRatio(retSLSP, MAR = rf*100))
  SOR.BH = as.numeric(SortinoRatio(retBH, MAR = rf*100))
  SOR.diff = SOR.SLSP - SOR.BH
  return(SOR.diff)
}
```

```
      # Criterion 4
      if (criterion == "R-VaR"){
        md.SLSP = median(retSLSP)
        md.BH = median(retBH)
        RVaR.RatioSLSP = (md.SLSP-rf)/(md.SLSP-quantile(retSLSP,0.05))
        RVaR.RatioBH = (md.BH-rf)/(md.BH-quantile(retBH,0.05))
        RvaR.diff = RVaR.RatioSLSP - RVaR.RatioBH
        return(RvaR.diff)
      }

      # Criterion 5
      if (criterion == "R-ES"){
        md.SLSP = median(retSLSP)
        md.BH = median(retBH)
        RES.RatioSLSP = (md.SLSP-rf)/(md.SLSP-
                        mean(retSLSP[retSLSP<=quantile(retSLSP,0.05)]))
        RES.RatioBH = (md.BH-rf)/(md.BH-
                        mean(retBH[retBH<=quantile(retBH,0.05)]))
        RES.diff = RES.RatioSLSP - RES.RatioBH
        return(RES.diff)
      }
    }

  stopCluster(cl); gc()
  star = which(performance == max(performance[performance != Inf]),
               arr.ind = TRUE)
  alpha.star.M2 = alphas[star[2]]
  beta.star.M2 = betas[star[1]]
  return(c(alpha.star.M2, beta.star.M2))

} else{
  N = nrow(h_prices)

  stock = h_rets[,colnames(h_rets)==ticker]

  performance = foreach(a = alphas, .combine = "cbind",
                        .packages = c("doParallel", "PerformanceAnalytics",
                                      "fGarch"),
                        .export = ls(.GlobalEnv)) %:%
    foreach (b = betas, .combine='c', .packages = c("PerformanceAnalytics",
                                      "fGarch","np")) %dopar% {
      oper = foreach(j = 1:reps, .combine = "rbind",
                     .multicombine=TRUE) %dopar%
      {
        stock_boot.ret = SBoot_trunc.geom(stock)
        stock_boot.prices = numeric(N)
        stock_boot.prices[1] = h_prices[,colnames(h_prices)==ticker][[1]]
        for (i in 2:N){
          stock_boot.prices[i] = stock_boot.prices[i-1]*
                                 (1+stock_boot.ret[i-1])
        }
        if (any(stock_boot.prices <= 0)) {
          firstneg = which(stock_boot.prices <= 0)[1]
```

```
    stock_boot.prices[firstneg:length(stock_boot.prices)] = 0
  }
  P_0 = stock_boot.prices[1]

  SL = P_0*(1-a)
  SP = P_0*(1+b)
  t = 2
  while(t < N && stock_boot.prices[t]>0 && stock_boot.prices[t]>SL
        && stock_boot.prices[t]<SP){
    SL = max(stock_boot.prices[1:t])*(1-a)
    SP = max(stock_boot.prices[1:t])*(1+b)
    t = t+1
  }
  leaves = t
  Pvector = stock_boot.prices
  return(c(Pvector[t], Pvector[N], P_0, leaves))
}
P.SLSP = oper[,1]
P.BH = oper[,2]
P.0 = oper[,3]
leaves = oper[,4]
rf = (2.5/12)*0.03171
realized_rf = (1-leaves/N)*rf

retSLSP = (P.SLSP*(1+realized_rf)/P.0-1)*100
retBH = (P.BH/P.0-1)*100

# Criterion 1
if (criterion == "ExpRet"){
  Expected.ret.SLSP = mean(retSLSP)
  Expected.ret.BH = mean(retBH)
  ExpRet.diff = Expected.ret.SLSP - Expected.ret.BH
  return(ExpRet.diff)
}

# Criterion 2
if (criterion == "Sharpe"){
  SharpeSLSP = (mean(retSLSP)-rf)/sd(retSLSP)
  SharpeBH = (mean(retBH)-rf)/sd(retBH)
  Sharpe.diff = SharpeSLSP - SharpeBH
  return(Sharpe.diff)
}

# Criterion 3
if (criterion == "Sortino"){
  SOR.SLSP = as.numeric(SortinoRatio(retSLSP, MAR = rf*100))
  SOR.BH = as.numeric(SortinoRatio(retBH, MAR = rf*100))
  SOR.diff = SOR.SLSP - SOR.BH
  return(SOR.diff)
}

# Criterion 4
if (criterion == "R-VaR"){
```

```
        md.SLSP = median(retSLSP)
        md.BH = median(retBH)
        RVaR.RatioSLSP = (md.SLSP-rf)/(md.SLSP-quantile(retSLSP,0.05))
        RVaR.RatioBH = (md.BH-rf)/(md.BH-quantile(retBH,0.05))
        RvaR.diff = RVaR.RatioSLSP - RVaR.RatioBH
        return(RvaR.diff)
      }

      # Criterion 5
      if (criterion == "R-ES"){
        md.SLSP = median(retSLSP)
        md.BH = median(retBH)
        RES.RatioSLSP = (md.SLSP-rf)/(md.SLSP-
                        mean(retSLSP[retSLSP<=quantile(retSLSP,0.05)]))
        RES.RatioBH = (md.BH-rf)/(md.BH-
                        mean(retBH[retBH<=quantile(retBH,0.05)]))
        RES.diff = RES.RatioSLSP - RES.RatioBH
        return(RES.diff)

      }
    }

  stopCluster(cl); gc()
  star = which(performance == max(performance[performance != Inf]),
               arr.ind = TRUE)
  alpha.star.M3 = alphas[star[2]]
  beta.star.M3 = betas[star[1]]
  return(c(alpha.star.M3, beta.star.M3))
 }
}

param.star.M1 = pctg.barrier.param.optimizer(1, alphas, betas, "R-ES")
alpha.star.M1 = param.star.M1[1] ; beta.star.M1 = param.star.M1[2]

param.star.M2 = pctg.barrier.param.optimizer(2, alphas, betas, "R-ES")
alpha.star.M2 = param.star.M2[1] ; beta.star.M2 = param.star.M2[2]

num.stat = length(w.s)
Mparam_expret = matrix(nrow=num.stat, ncol=2)
for(i in 1:num.stat){
  Mparam_expret[i,] = pctg.barrier.param.optimizer(3, alphas, betas,
                                                   "ExpRet", ticker=w.s[i])
}
```

C.8 Simulation: Performance (using fixed percentage optimal barriers)

```
#########################################################################
#  Code 7. SIMULATION: Performance (using fixed % optimal barriers)     #
#                          Author: Albert Dorador                       #
#########################################################################
```

```
# rules: pctg, ATR, RSI, MA

# Function to evaluate performance of a given SL rule, given a price model
SLSP.performance = function(model, rule, ticker=NULL, reps = 1e5){
  require(doParallel)
  num_cores = detectCores()-1
  cl = makeCluster(num_cores, type="PSOCK")
  registerDoParallel(cl)

  if (model == 1) {
    sigma_0 = sdv.star
    sigma2_0 = sigma_0^2
    oper = foreach(j = 1:reps, .combine = "rbind", .multicombine=TRUE,
          .packages = c("fGarch", "zoo", "TTR"),
          .export = ls(.GlobalEnv)) %dopar%
          {
            gaps = rweibull(n.gap,shape=new_sh.star,scale=new_sc.star)
            P_0 = sample.int(200,1) #first price
            epsilon2_vector = numeric(n.tot)
            epsilon2_vector[1] = (rsged(1, mean = 0, sd = 1, nu = shape,
                                        xi = skew))^2
            Pvector = numeric(n.tot)
            Pvector[1] = P_0
            no_gap_multiplier = rel.beta0^2 #because SD*k --> Var*k^2
            sigma2_vector = numeric(n.tot)
            sigma2_vector[1] = sigma2_0

            # Decide which days will have a gap:
            confirmed_gaps_ind = sample(c(0,1),n.gap,replace = TRUE,
                                        prob = c(0.2207,0.7793))
                                      # 77.93% chance of a gap
            confirmed_gaps = gaps[which(confirmed_gaps_ind==1)]
            confirmed_gaps_ind = gaps_ind[which(confirmed_gaps_ind==1)]

            # Generate prices:
            counter = 0
            for(t in 2:n.tot){
              if (!(t %in% gaps_ind)){ #not a gappable day
                epsilon2_vector[t] = (rsged(1, mean = 0, sd = 1, nu = shape,
                                            xi = skew))^2
                sigma2_vector[t] = (omega+alpha*epsilon2_vector[t-1]*
                                    sigma2_vector[t-1]+beta*
                                    sigma2_vector[t-1])*no_gap_multiplier
                r = rged(1, mean = mu, sd = sqrt(sigma2_vector[t]),
                         nu = nu.star)
                Pvector[t] = Pvector[t-1]*(1+r)
              }else{ #a gappable day
                if(t %in% confirmed_gaps_ind){ #truly a gap day!
                  counter = counter + 1
                  epsilon2_vector[t] = (rsged(1, mean = 0, sd = 1, nu = shape,
                                              xi = skew))^2
                  gm = abs(1-gaps[counter])*100
                  gm_multiplier = (rel.beta0 + rel.beta1*gm +
```

```
                                rel.beta2*gm^2 + rel.beta3*gm^3)^2
        sigma2_vector[t] = (omega+alpha*epsilon2_vector[t-1]*
                                sigma2_vector[t-1]+beta*
                                sigma2_vector[t-1])*gm_multiplier
        Pvector[t] = Pvector[t-1]*gaps[counter]
      }else{
        no_gap_multiplier = rel.beta0^2
        epsilon2_vector[t] = (rsged(1, mean = 0, sd = 1, nu = shape,
                                    xi = skew))^2
        sigma2_vector[t] = (omega+alpha*epsilon2_vector[t-1]*
                        sigma2_vector[t-1]+beta*sigma2_vector[t-1])*
                        no_gap_multiplier
        Pvector[t] = Pvector[t-1] #keep price the same
      }
    }
  }
  if (any(Pvector <= 0)) { #if stock price reaches 0, stays there
    firstneg = which(Pvector <= 0)[1]
    Pvector[firstneg:length(Pvector)] = 0
  }

  if (rule == "pctg") {
    SL = P_0*(1-alpha.star.M1)
    SP = P_0*(1+beta.star.M1)
    t = 2
    while(t < n.tot && Pvector[t]>0 && Pvector[t]>SL &&
          Pvector[t]<SP){
      SL = max(Pvector[1:t])*(1-alpha.star.M1)
      SP = max(Pvector[1:t])*(1+beta.star.M1)
      t = t+1
    }
  }

  if (rule == "ATR") {
    HLC = matrix(nrow = 252, ncol = 3)
    close_ind = seq(8,(7*252+1),7)-1
    HLC[,1] = rollapply(Pvector, 7, max, by=7)
    HLC[,2] = rollapply(Pvector, 7, min, by=7)
    HLC[,3] = Pvector[close_ind]
    atr = ATR(HLC, n=14)[15,2] #period is 7 days

    # We must check at each hour if we have broken the SL/SP level set
    # Higher freq would mean never break
    SL = max(Pvector[99]-2.5*atr, 0) #to prevent SL < 0
    SP = Pvector[99]+3*atr
    t = 100
    while(t < n.tot && Pvector[t]>0 && Pvector[t]>SL &&
          Pvector[t]<SP){ #barriers are CHECKED every period
      if (t%%7 == 1){ #UPDATE barriers once a day,i.e. every 7 periods
        SL = max(Pvector[t]-2.5*atr, 0)
        SP = Pvector[t]+3*atr
      }
      t = t+1
```

```
            }
          }

          if (rule == "RSI") {
            rsi_vec = RSI(Pvector, n=49) #period is 7 days. Computes all RSI
            rsi = rsi_vec[50]
            t = 50
            # We must check at each hour if we have RSI >= 70
            while(t < n.tot && Pvector[t]>0 && rsi < 70){
              t = t+1
              rsi = rsi_vec[t]
            }
          }

          if (rule == "MA") {
            MA.5_vec = SMA(Pvector, n=35) #Computes ALL MA.5?s
            MA.20_vec = SMA(Pvector, n=140) #Computes ALL MA.20?s
            MA.70_vec = SMA(Pvector, n=490) #Computes ALL MA.70?s

            MA.5 = MA.5_vec[490]
            MA.20 = MA.20_vec[490]
            MA.70 = MA.70_vec[490]

            t = 490
            triple.cross = FALSE
            # We must check at each hour if we have the triple crossover
            while(t < n.tot && Pvector[t]>0 && !triple.cross){
              if (MA.5 > MA.20 && MA.20 > MA.70){
                while (t < n.tot && Pvector[t]>0 && !triple.cross){
                  if (MA.5 < MA.20 && MA.20 < MA.70){triple.cross = TRUE}
                  t = t+1
                  MA.5 = MA.5_vec[t]
                  MA.20 = MA.20_vec[t]
                  MA.70 = MA.70_vec[t]
                }
              }
              t = t+1
              MA.5 = MA.5_vec[t]
              MA.20 = MA.20_vec[t]
              MA.70 = MA.70_vec[t]
            }
            t = t-2 # MA rule requires this to properly compute leaves
          }

          leaves = t
          return(c(Pvector[t], Pvector[n.tot], P_0, leaves))
        }
  } else if (model == 2) {
    sigma_0 = sdv.star
    sigma2_0 = sigma_0^2
    oper = foreach(j = 1:reps, .combine = "rbind", .multicombine=TRUE,
           .packages = c("fGarch", "zoo", "TTR"),
           .export = ls(.GlobalEnv)) %dopar%
```

```
{
  gaps = rweibull(n.gap,shape=new_sh.star,scale=new_sc.star)
  P_0 = sample.int(200,1) #first price
  ret_vector = numeric(n.tot)
  epsilon_vector = numeric(n.tot)
  epsilon_vector[1] = rsged(1, mean = 0, sd = 1, nu = shape, xi = skew)
  epsilon_vector[2] = rsged(1, mean = 0, sd = 1, nu = shape, xi = skew)
  epsilon_vector[3] = rsged(1, mean = 0, sd = 1, nu = shape, xi = skew)
  epsilon_vector[4] = rsged(1, mean = 0, sd = 1, nu = shape, xi = skew)
  Pvector = numeric(n.tot)
  Pvector[1] = P_0
  no_gap_multiplier = rel.beta0^2 #because SD*k --> Var*k^2
  sigma_vector = numeric(n.tot)
  sigma_vector[1] = sigma_0
  sigma_vector[2] = (omega+alpha*(epsilon_vector[1]*sigma_vector[1])^2+
                       beta*sigma_vector[1]^2)*no_gap_multiplier
  sigma_vector[3] = (omega+alpha*(epsilon_vector[2]*sigma_vector[2])^2+
                       beta*sigma_vector[2]^2)*no_gap_multiplier
  sigma_vector[4] = (omega+alpha*(epsilon_vector[3]*sigma_vector[3])^2+
                       beta*sigma_vector[3]^2)*no_gap_multiplier
  # Decide which days will have a gap:
  confirmed_gaps_ind = sample(c(0,1),n.gap,replace = TRUE,
                              prob = c(0.2207,0.7793))
  confirmed_gaps = gaps[which(confirmed_gaps_ind==1)]
  confirmed_gaps_ind = gaps_ind[which(confirmed_gaps_ind==1)]

  # Generate prices:
  r1 = rged(1, mean = mu, sd = sigma_vector[1], nu = nu.star)
  r2 = m + ar1*r1 + (epsilon_vector[2]*sigma_vector[2])^2
  r3 = m + ar1*r2+ar2*r1 + (epsilon_vector[3]*sigma_vector[3])^2
  ret_vector[1] = r1
  ret_vector[2] = r2
  ret_vector[3] = r3

  Pvector[2] = Pvector[1]*(1+r1)
  Pvector[3] = Pvector[2]*(1+r2)
  Pvector[4] = Pvector[3]*(1+r3)

  ret_vector[4] = m + ar1*r3+ar2*r2+ar3*r1 + (epsilon_vector[4]*
                                              sigma_vector[4])^2

  counter = 0
  for(t in 5:n.tot){
    if (!(t %in% gaps_ind)){ #not a gappable day
      epsilon_vector[t] = rsged(1, mean = 0, sd = 1, nu = shape,
                                xi = skew)
      sigma_vector[t] = sqrt((omega+alpha*(epsilon_vector[t-1]*
                      sigma_vector[t-1])^2+beta*sigma_vector[t-1]^2)*
                      no_gap_multiplier)
      ret_vector[t] = m + ar1*ret_vector[t-1]+ar2*ret_vector[t-2]+
          ar3*ret_vector[t-3] + (epsilon_vector[t]*sigma_vector[t])^2
      Pvector[t] = Pvector[t-1]*(1+ret_vector[t])
    }else{ #a gappable day
```

```
      if(t %in% confirmed_gaps_ind){ #truly a gap day!
        counter = counter + 1
        epsilon_vector[t] = rsged(1, mean = 0, sd = 1, nu = shape,
                                  xi = skew)
        gm = abs(1-gaps[counter])*100
        gm_multiplier = (rel.beta0 + rel.beta1*gm + rel.beta2*gm^2 +
                           rel.beta3*gm^3)^2
        sigma_vector[t] = sqrt((omega+alpha*(epsilon_vector[t-1]*
                    sigma_vector[t-1])^2+beta*sigma_vector[t-1]^2)*
                      gm_multiplier)
        Pvector[t] = Pvector[t-1]*gaps[counter]
      }else{
        no_gap_multiplier = rel.beta0^2
        epsilon_vector[t] = rsged(1, mean = 0, sd = 1, nu = shape,
                                  xi = skew)
        sigma_vector[t] = sqrt((omega+alpha*(epsilon_vector[t-1]*
                sigma_vector[t-1])^2+beta*sigma_vector[t-1]^2)*
                  no_gap_multiplier)
        Pvector[t] = Pvector[t-1] #keep price the same
      }
    }
  }
  if (any(Pvector <= 0)){ #if stock price reaches 0, stays there
    firstneg = which(Pvector <= 0)[1]
    Pvector[firstneg:length(Pvector)] = 0
  }

  if (rule == "pctg") {
    SL = P_0*(1-alpha.star.M2)
    SP = P_0*(1+beta.star.M2)
    t = 2
    while(t < n.tot && Pvector[t]>0 && Pvector[t]>SL &&
          Pvector[t]<SP){
      SL = max(Pvector[1:t])*(1-alpha.star.M2)
      SP = max(Pvector[1:t])*(1+beta.star.M2)
      t = t+1
    }
  }

  if (rule == "ATR") {
    HLC = matrix(nrow = 252, ncol = 3)
    close_ind = seq(8,(7*252+1),7)-1
    HLC[,1] = rollapply(Pvector, 7, max, by=7)
    HLC[,2] = rollapply(Pvector, 7, min, by=7)
    HLC[,3] = Pvector[close_ind]
    atr = ATR(HLC, n=14)[15,2] #period is 7 days

    SL = max(Pvector[99]-2.5*atr, 0) #to prevent SL < 0
    SP = Pvector[99]+3*atr
    t = 100
    while(t < n.tot && Pvector[t]>0 && Pvector[t]>SL && Pvector[t]<SP){
      if (t%%7 == 1){
        SL = max(Pvector[t]-2.5*atr, 0)
```

```
            SP = Pvector[t]+3*atr
          }
          t = t+1
        }
      }

      if (rule == "RSI") {
        rsi_vec = RSI(Pvector, n=49) #period is 7 days. Computes all RSI
        rsi = rsi_vec[50]

        t = 50
        # We must check at each hour if we have RSI >= 70
        while(t < n.tot && Pvector[t]>0 && rsi < 70){
          t = t+1
          rsi = rsi_vec[t]
        }
      }

      if (rule == "MA") {
        MA.5_vec = SMA(Pvector, n=35) #Computes all MA.5's
        MA.20_vec = SMA(Pvector, n=140) #Computes all MA.20's
        MA.70_vec = SMA(Pvector, n=490) #Computes all MA.70's

        MA.5 = MA.5_vec[490]
        MA.20 = MA.20_vec[490]
        MA.70 = MA.70_vec[490]

        t = 490
        triple.cross = FALSE
        # We must check at each hour if we have the triple crossover
        while(t < n.tot && Pvector[t]>0 && !triple.cross){
          if (MA.5 > MA.20 && MA.20 > MA.70){
            while (t < n.tot && Pvector[t]>0 && !triple.cross){
              if (MA.5 < MA.20 && MA.20 < MA.70){triple.cross = TRUE}
              t = t+1
              MA.5 = MA.5_vec[t]
              MA.20 = MA.20_vec[t]
              MA.70 = MA.70_vec[t]
            }
          }
          t = t+1
          MA.5 = MA.5_vec[t]
          MA.20 = MA.20_vec[t]
          MA.70 = MA.70_vec[t]
        }
        t = t-2 # MA rule requires this to properly compute leaves
      }

      leaves = t
      return(c(Pvector[t], Pvector[n.tot], P_0, leaves))
    }
  } else {
    N = nrow(h_prices)
```

```
if(ticker=="BAC"){s=1}
if(ticker=="PFE"){s=2}
if(ticker=="F"){s=3}
if(ticker=="C"){s=4}
if(ticker=="JPM"){s=5}
if(ticker=="MRK"){s=6}
if(ticker=="XOM"){s=7}
if(ticker=="AIG"){s=8}
if(ticker=="HAL"){s=9}
if(ticker=="KEY"){s=10}
if(ticker=="BMY"){s=11}

stock = h_rets[,colnames(h_rets)==ticker]

oper = foreach(j = 1:reps, .combine = "rbind", .multicombine=TRUE,
       .packages = c("zoo", "TTR", "np"),
       .export = ls(.GlobalEnv)) %dopar%
       {
         stock_boot.ret = SBoot_trunc.geom(stock)
         stock_boot.prices = numeric(N)
         stock_boot.prices[1] = h_prices[,colnames(h_prices)==ticker][[1]]
         for (i in 2:N){
           stock_boot.prices[i] = stock_boot.prices[i-1]*(1+stock_boot.ret[i-1])
         }
         if (any(stock_boot.prices <= 0)) { #if stock price reaches 0, stays
           firstneg = which(stock_boot.prices <= 0)[1]
           stock_boot.prices[firstneg:length(stock_boot.prices)] = 0
         }
         P_0 = stock_boot.prices[1]

         if (rule == "pctg") {
           alpha.star = Mparam_expret[s,1]
           beta.star = Mparam_expret[s,2]

           SL = P_0*(1-alpha.star)
           SP = P_0*(1+beta.star)
           t = 2
           while(t < N && stock_boot.prices[t]>0 && stock_boot.prices[t]>SL &&
                 stock_boot.prices[t]<SP){
             SL = max(stock_boot.prices[1:t])*(1-alpha.star)
             SP = max(stock_boot.prices[1:t])*(1+beta.star)
             t = t+1
           }
         }

         if (rule == "ATR") {
           HLC = matrix(nrow = 54, ncol = 3)
           close_ind = seq(8,(N+1),7)-1
           HLC[,1] = rollapply(stock_boot.prices, 7, max, by=7)
           HLC[,2] = rollapply(stock_boot.prices, 7, min, by=7)
           HLC[,3] = stock_boot.prices[close_ind]
           atr = ATR(HLC, n=14)[15,2] #period is 7 days
```

```
  SL = max(stock_boot.prices[99]-2.5*atr, 0) #to prevent SL < 0
  SP = stock_boot.prices[99]+3*atr
  t = 100
  while(t < N && stock_boot.prices[t]>0 && stock_boot.prices[t]>SL &&
        stock_boot.prices[t]<SP){
    if (t%%7 == 1){
      SL = max(stock_boot.prices[t]-2.5*atr, 0)
      SP = stock_boot.prices[t]+3*atr
    }
    t = t+1
  }
}

if (rule == "RSI") {
  rsi_vec = RSI(stock_boot.prices, n=49)
  rsi = rsi_vec[50]
  t = 50
  # We must check at each hour if we have RSI >= 70
  while(t < N && stock_boot.prices[t]>0 && rsi < 70){
    t = t+1
    rsi = rsi_vec[t]
  }
}

if (rule == "MA") {
  MA.3_vec = SMA(stock_boot.prices, n=21) #Computes all MA.3's
  MA.12_vec = SMA(stock_boot.prices, n=84) #Computes all MA.12's
  MA.30_vec = SMA(stock_boot.prices, n=210) #Computes all MA.30's

  MA.3 = MA.3_vec[210]
  MA.12 = MA.12_vec[210]
  MA.30 = MA.30_vec[210]

  t = 210
  triple.cross = FALSE
  # We must check at each hour if we have the triple crossover
  while(t < N && stock_boot.prices[t]>0 && !triple.cross){
    if (MA.3 > MA.12 && MA.12 > MA.30){
      while (t < N && stock_boot.prices[t]>0 && !triple.cross){
        if (MA.3 < MA.12 && MA.12 < MA.30){triple.cross = TRUE}
        t = t+1
        MA.3 = MA.3_vec[t]
        MA.12 = MA.12_vec[t]
        MA.30 = MA.30_vec[t]
      }
    }
    t = t+1
    MA.3 = MA.3_vec[t]
    MA.12 = MA.12_vec[t]
    MA.30 = MA.30_vec[t]
  }
  t = t-2 # MA rule requires this to properly compute leaves
}
```

```
            leaves = t
            Pvector = stock_boot.prices
            return(c(Pvector[t], Pvector[N], P_0, leaves))
          }
}
stopCluster(cl); gc()

# Result analysis, based on our 5 criteria
P.SLSP = oper[,1]
P.BH = oper[,2]
P.0 = oper[,3]
leaves = oper[,4]

if (model == 3) {
  rf = (2.5/12)*0.03171
  n.tot = N
} else {
  rf = 0.03171
}

realized_rf = (1-leaves/n.tot)*rf

retSLSP = (P.SLSP*(1+realized_rf)/P.0-1)*100
retBH = (P.BH/P.0-1)*100

# (1) Expected return
expected.retSLSP = mean(retSLSP)
expected.retBH = mean(retBH)

# (2) Sharpe ratio
SharpeSLSP = (mean(retSLSP)-rf)/sd(retSLSP)
SharpeBH = (mean(retBH)-rf)/sd(retBH)

# (3) Sortino
require(PerformanceAnalytics)
SOR.SLSP = SortinoRatio(retSLSP, MAR = rf*100)
SOR.BH = SortinoRatio(retBH, MAR = rf*100)

# (4) R-Var
md.SLSP = median(retSLSP)
md.BH = median(retBH)
RVaR.RatioSLSP = (md.SLSP-rf)/(md.SLSP-quantile(retSLSP,0.05))
RVaR.RatioBH = (md.BH-rf)/(md.BH-quantile(retBH,0.05))

# (5) R-ES
RES.RatioSLSP = (md.SLSP-rf)/(md.SLSP-
                  mean(retSLSP[retSLSP<=quantile(retSLSP,0.05)]))
RES.RatioBH = (md.BH-rf)/(md.BH-
                  mean(retBH[retBH<=quantile(retBH,0.05)]))

return(list(SL.expret = expected.retSLSP,
            BH.expret = expected.retBH,
```

```
                SL.Sharpe = SharpeSLSP,
                BH.Sharpe = SharpeBH,
                SL.Sortino = SOR.SLSP,
                BH.Sortino = SOR.BH,
                SL.RVAR = RVaR.RatioSLSP,
                BH.RVAR = RVaR.RatioBH,
                SL.RES = RES.RatioSLSP,
                BH.RES = RES.RatioBH))
}

M1.SLperf = SLSP.performance(model=1, rule="ATR")
M2.SLperf = SLSP.performance(model=2, rule="pctg")

num.stat = length(w.s)
Boot.Perf.df = data.frame()
for(i in 1:num.stat){
  Boot.Perf.aux = as.data.frame(SLSP.performance(model=3, rule="RSI",
                                                 ticker=w.s[i]))
  Boot.Perf.df = rbind(Boot.Perf.df, Boot.Perf.aux)
}

# Function to compute column means by group
rowmean = function(M, group = rownames(M), reord = FALSE){
  gr.count = unlist(lapply(split(group,f=group),length))
  M.sum = rowsum(M, group, reorder = reord)
  return(M.sum/gr.count)
}
```

Bibliography

[1] Acar, E. and Toffel, R. [2001], *Stop-loss and Investment Returns*, Financial Times / Prentice Hall. ISBN 0273650343, pp. 51-68.

[2] Amdahl, G. M. [1967], 'Validity of the single processor approach to achieving large scale computing capabilities', *AFIPS '67 (Spring) Proceedings of the April 18-20, 1967, spring joint computer conference* pp. 483–485. IBM Sunnyvale, California.

[3] Artzner, P., Delbaen, F., Eber, J. M. and Heath, D. [1999], 'Coherent measures of risk', *Mathematical Finance* **9**(3), 203.

[4] Bachelier, L. [1900], 'Theory of speculation'. translation of 1900 French edition in Cootner, P.H. [1964], *The Random Character of Stock Market Prices*, pp. 17-78.

[5] Brockwell, P. and Davis, R. [1991], *Time Series: Theory and Methods (2nd ed.)*, Springer. ISBN 0387974296, p. 273.

[6] Burnham, K. P. and Anderson, D. R. [2004], *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach (2nd ed.)*, Springer-Verlag. ISBN 0387953647, ch. 7.

[7] Canty, A. and Ripley, B. [2016], *boot: Bootstrap R (S-Plus) Functions.* R package version 1.3-18.
URL: *https://cran.r-project.org/web/packages/boot/index.html*

[8] Chen, Q. and Gerlach, R. H. [2013], 'The two-sided weibull distribution and forecasting financial tail risk', *International Journal of Forecasting* **29**(4), 527540.

[9] Chernick, M. R. and LaBudde, R. A. [2011], *An Introduction to Bootstrap Methods with Applications to R*, Wiley. ISBN 978-0-470-46704-6.

[10] Constantine, W. and Percival, D. [2016], *fractal: Fractal Time Series Modeling and Analysis.* R package version 2.0-1.
URL: *https://CRAN.R-project.org/package=fractal*

[11] Dorador, A. [2016], *powerplus: Exponentiation Operations.* R package version 3.0.
URL: *https://CRAN.R-project.org/package=powerplus*

[12] Dorador, A. [2017], *analytics: Regression Outlier Detection and Other Tools for Data Analysis.* R package version 1.0.
URL: *https://CRAN.R-project.org/package=analytics*

[13] Franco, C. and Zakoian, J. M. [2012], 'Strict stationarity testing and estimation of explosive and stationary generalized autoregressive conditional heteroscedasticity models', *Econometrica* **80**(2), 821–861.

[14] Hayfield, T. and Racine, J. S. [2008], 'Nonparametric econometrics: The np package', *Journal of Statistical Software* **27**(5).
URL: *http://www.jstatsoft.org/v27/i05/*

[15] James, J. and Yang, L. [2010], 'Stop-losses, maximum drawdown-at-risk and replicating financial time series with the stationary bootstrap', *Quantitative Finance* **10**(1), 1–12.

[16] Johnson, M. E., Tietjen, G. L. and Beckman, R. J. [1980], 'A new family of probability distributions with applications to monte carlo studies', *Journal of the American Statistical Association* **75**(370), 276–279.

[17] Justus, C., Hargraves, W., Mikhail, A. and Graber, D. [1977], 'Methods for estimating wind speed frequency distributions', *Journal of Applied Meteorology* **17**(3), 350–353.

[18] Kaminski, K. M. and Lo, A. [2014], 'When do stop-loss rules stop losses?', *Journal of Financial Markets* **18**(C), 234–254.

[19] Lim, A. and Tjhi, W. [2015], *R High Performance Programming*, Packt Publishing. ISBN 978-1783989263.

[20] McQuarrie, A. D. R. and Tsai, C.-L. [1998], *Regression and Time Series Model Selection*, World Scientific. ISBN 981-02-3242-X.

[21] Moors, J. J. A. [1986], 'The meaning of kurtosis: Darlington reexamined', *The American Statistician* **40**(4), 283–284.

[22] Nadarajah, S. [2005], 'A generalized normal distribution', *Journal of Applied Statistics* **32**(7), 685–694.

[23] Nadarajah, S. and Pogany, T. K. [2010], 'On the characteristic function of the generalized normal distribution', *Comptes Rendus Mathematique* **348**(3-4), 203–206.

[24] Nwobi, F. N. and Ugomma, C. A. [2014], 'A comparison of methods for the estimation of weibull distribution parameters', *Metodoloski zvezki* **11**(1), 65–78.

[25] Olatayo, T. O. [2014], 'Truncated geometric bootstrap method for time series stationary process', *Applied Mathematics* **5**, 2057–2061.

[26] Peterson, B. G. and Carl, P. [2014], *PerformanceAnalytics: Econometric tools for performance and risk analysis*. R package version 1.4.3541.
URL: *https://CRAN.R-project.org/package=PerformanceAnalytics*

[27] Politis, D. N. and Romano, J. P. [1994], 'The stationary bootstrap', *Journal of the American Statistical Association* **89**(428), 1303–1313.

[28] Politis, D. N. and Romano, J. P. [1995], 'Bias-corrected nonparametric spectral estimation', *Journal of time series analysis* **16**(1), 67–103.

[29] Politis, D. N. and White, H. [2004], 'Automatic block-length selection for the dependent bootstrap', *Econometric Reviews* **23**(1), 53–70.

[30] Priestley, M. B. and Rao, S. T. [1969], 'A test for non-stationarity of time-series', *Journal of the Royal Statistical Society. Series B (Methodological)* **31**(1), 140–149.

[31] R Core Team [2016], *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *https://www.R-project.org/*

[32] Ramirez, A. and Cox, C. [2012], 'Improving on the range rule of thumb', *Rose-Hulman Undergraduate Mathematics Journal* **13**(2).

[33] Sharpe, W. F. [1994], 'The sharpe ratio', *The Journal of Portfolio Management* **21**(1), 49–58.

[34] Sornette, D. [2004], *Why Stock Markets Crash: Critical Events in Complex Financial Systems*, Princeton University Press. ISBN 0691118507.

[35] Sortino, F. [1994], 'Performance measurement in a downside risk framework', *Journal of Investing* **3**, 50–58.

[36] Triola, M. F. [2010], *Elementary Statistics*, Pearson Education. ISBN 0-321-50024-5.

[37] Tsay, R. S. [2010], *Analysis of Financial Time Series*, Wiley, 3rd edition. ISBN 978-0470414354.

[38] Verbeek, M. [2004], *A Guide to Modern Econometrics*, Wiley, 2nd edition. ISBN 0-470-85773-0.

[39] Westfall, P. H. [2014], 'Kurtosis as peakedness, 1905 − 2014. r.i.p.', *The American Statistician* **68**(3), 191–195.

[40] Wooldridge, J. M. [2008], *Introductory Econometrics: A Modern Approach*, South-Western, 4th edition. ISBN 978-0324660548.

[41] Wuertz, D., with contribution from Michal Miklovic, Y. C., Boudt, C., Chausse, P. and others [2016], *fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling*. R package version 3010.82.1.
URL: *https://CRAN.R-project.org/package=fGarch*

[42] Zambelli, A. E. [2016], 'Determining optimal stop-loss thresholds via bayesian analysis of drawdown distributions', *Cornell University (under review)* .