NTNU
Norwegian University of
Science and Technology

Spread Trading in Brent Crude Futures

A stochastic approach to intraday calendar
spread trading

August Abelvik-Engmark
Daniel Vårdal Haugland

Industrial Economics and Technology Management
Submission date: June 2018
Supervisor: Sjur Westgaard, IØT

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

Abstract

In this thesis, we propose an intradaily spread trading strategy based on a stochastic process model. We go on to examine whether this strategy can be profitably applied in Brent Crude oil futures markets over the Jan-2015 to Apr-2018 period. For this purpose, tick-by-tick trading data of 63 unique Brent Crude oil futures contracts are used to construct intradaily data sets with 5-minute resolution. By considering the 9 most liquid futures contracts and by constructing 18 different calendar spreads for trading, we perform a thorough backtest of the intradaily trading strategy. Under optimistic assumptions, our strategy achieves a maximum Sharpe ratio of 4.3. Under conservative assumptions, however, Sharpe ratios are negative for all parameter choices. We conclude that intraday spread trading in Brent Crude futures based on the stochastic process model put forward in this thesis is not profitable. Although we show that such strategies may be highly profitable under optimistic assumptions, we emphasize that results are very sensitive to small changes in bid-ask spreads and the timing of trade execution. As these model parameters are difficult to estimate correctly without order book data, we conclude that a cautious approach should be taken when implementing these parameters in a backtest.

Samandrag

I denne oppgåva foreslår vi ein intra-dagleg handelsstrategi for par av framtidskontraktar basert på ein stokastisk prosessmodell. Vi held fram å undersøkja om denne strategien kan verta lønsamt anvendt i marknaden for framtidskontraktar for råolje av typen Brent, i perioden januar 2015 til april 2018. Til dette føremålet vert brukt tikk-for-tikk-handelsdata for 63 framtidskontraktar for råolje av typen Brent til å konstruera eit intra-daglig data-sett med oppløysing på 5 minutt. Ved å vurdera dei 9 mest likvide framtidskontraktane og ved å setje saman 18 ulike månads-par for handel, utfører vi ein grundig tilbakeprøving av handelsstrategien. Med optimistiske parameterval oppnår strategien vår eit maksimalt Sharpe-tilhøve på 4.3. Under konservative parameterval er likevel Sharpe-tilhøvet negativt for samtlege parameterval. Vi konkluderer med at intra-dagleg handel basert på den stokastiske prosessmodellen presentert i denne oppgåva ikkje er lønsamt for framtidskontraktar for råolje av typen Brent. Sjølv om vi viser at slike strategiar kan vere svært lønsame under optimistiske parameterval, legg vi vekt på at resultata er svært kjenslevare for små endringar i bud- og tilbudskursar, samt samtidigheita i handel. Då desse modellparametrane er vanskelege å anslå riktig utan tilgong på opne bud- og tilbudsdata, konkluderer vi med at ei varsam tilnærming bør takast ved implementering av desse parametrane i ein slik tilbakeprøving.

Preface

This thesis concludes our Master of Science in Industrial Economics and Technology Management with specialization in Financial Engineering at NTNU. Studying quantitative finance has been both inspiring and challenging. The field lies at the intersection of several academic disciplines we find interesting: finance, computer science, mathematics and statistics. We hope that the reader will find this thesis as interesting as we have found the topic of spread trading to be.

We would like to thank our supervisor, Sjur Westgaard, for his valuable input during the writing process. We would also like to thank Morten Hegna in *Montel* for granting us access to their extensive database of energy futures data, and for helping us understand the data collection process.

Trondheim, 2018-06-25

August AE — Daniel V.H.

August Abelvik-Engmark — Daniel Vårdal Haugland

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we examine whether a popular quantitative trading strategy known as spread trading[1] can be profitably applied in Brent Crude oil futures markets on an intradaily basis. We examine tick-by-tick trading data of 63 unique Brent Crude futures contracts, from which we construct designated intradaily data sets with 5-minute resolution. By considering the 9 most liquid futures contracts, and constructing 18 different calendar spreads for trading, we perform a thorough backtest of an intradaily spread trading strategy based on a stochastic process model.

Spread trading is conceptually simple: identify a pair of securities that tend to move together over time, in order to exploit any unusual deviations in their relative pricing. Profits are made by simultaneously entering short positions in relative winners and long positions in relative losers when sufficient deviations from equilibrium are observed, and by unwinding the positions upon convergence. Spread trading is thus an attempt to profit from temporary deviations from the "correct" relative pricing of securities. There is a notable number of studies on spread trading in the literature. Some of them produce exceptional results, such as Gatev, Goetzmann, and Rouwenhorst (2006), while Do and Faff (2009) prove the lack of persistence in the spectacular findings made by others. Both of the aforementioned papers study daily data of stock prices. The more recent paper of Liu, Chang, and Geman (2016) perform back-testing of high-frequency spread trading strategies in oil-related stock and achieve a Sharpe ratio of 7.2. The focus on stochastic process modelling of spreads in this thesis is largely motivated by the results achieved by Liu et al. (2016). Further, to our knowledge, there has not been conducted studies of intraday spread trading in futures markets.

A distinct trait of futures markets is that going long and going short is equally easy. For long-short strategies, this clearly favours futures markets compared to e.g. the stock market, where borrowing shares can be difficult and often entails high borrowing costs.

We look specifically at Brent Crude oil futures traded at the ICE because it is one of the most liquid commodities futures market in the world. Hundreds of thousands of "lots" change hands every day, where one lot represents one thousand barrels of crude oil - the minimum quantity of crude oil allowed for trading. In dollar figures, these volumes represent a daily turnover in the range of tens of billions of

[1] Spread trading and pairs trading are two terms describing the same concept. Both terms are used in the literature, and we will use both terms interchangeably in this thesis.

dollars. This contributes largely to the motivation for this thesis. The size of the financial trade in Brent Crude oil is - in lack of a better word - massive.

Our contribution to the existing literature is twofold: first, we conduct a detailed empirical study of intraday prices of Brent Crude futures; second, we examine whether a spread trading strategy based on a stochastic process model can be profitably applied in Brent Crude oil futures markets on an intradaily basis.

The rest of this thesis is structured as follows. Chapter two reviews selected literature on spread trading and statistical arbitrage. Chapter three introduces the dataset with descriptive statistics and explains the methodology for constructing intradaily data series from tick data. Chapter four presents our implementation of the stochastic spread trading strategy and the backtesting environment. Chapter five presents our main results and findings. Chapter six concludes and suggests topics for further research.

Chapter 2

Spread trading

2.1 Conceptual description

2.1.1 Statistical arbitrage

Statistical Arbitrage is a broad term used to describe financial trading strategies that rely on mean-reversion in the relative prices between two or more securities[1] (often within similar industries or exposed to similar risk factors). The trading strategy is motivated by the fact that prices of certain securities tend to move together. Using historical data and statistical methods, the trader identifies the typical trading patterns of the securities and use this as a basis for creating a long-short portfolio of the securities in such a way that the total exposure is market-neutral ($\beta \approx 0$). This may result in both winning and losing trades in the short-term, but if there are gains on average, it leads to profits over time. According to Gatev et al. (2006), the strategy was first made popular by Wall Street ”quant” Nunzio Tartaglia of Morgan Stanley in the mid-1980's, where he assembled a team of physicists, mathematicians and computer scientists to uncover arbitrage opportunities in the equities markets. Since then, statistical arbitrage-style trading has become wildly popular with investment banks and hedge funds, and increasingly so with quantitative hedge funds after the exponential increase in computing power and speed of execution available to market participants.

2.1.2 Pairs trading (Spread trading)

The pairs trade is a sub-category of statistical arbitrage, with only two securities involved in each statistical arbitrage portfolio (hence the name ”pairs trading”). If the relative price between the securities can be shown to be mean-reverting, a long-short strategy involving the two securities can be used. A long position is taken in the ”undervalued” security and a short position in the ”overvalued” security. The underlying assumption is that the ”mispricing” between the securities will be corrected by the market in the future. Entry and exit signals for the trade is typically generated based on historical means of the relative (log) price spreads and thresholds set by the trader. The mean-reversion of a spread is often attributed to some fundamental relationship between the series. E.g. the costs and margins of oil refineries in the case

[1] In this thesis we use the term *security* for any tradable financial asset, including: debt, equity and derivatives.

of the crack spread, or the underlying business in the case of two stocks from the same industry.

The "Law of One Price" (LOP), which is central in modern finance theory, postulates that two assets yielding identical outcomes should have the same price. In practice, few assets are truly identical and factors such as transaction costs further complicate the LOP in real-world markets. Gatev et al. (2006) suggests that random liquidity shocks can affect the market in the short-term, causing prices to diverge. Trading professionals engaging in pairs trading may consider assets which are quite different (e.g. stocks of two companies), but use statistical methods to detect historical patterns which suggest a "LOP-type" relationship between them. Two important points should be made in this context:

1. Because it is based on *relative* pricing, statistical arbitrage traders do not need to focus on the economically correct price of the underlying assets (Gatev et al., 2006). The strategy does not depend on whether the underlying securities are priced "correctly" or whether the general market goes up or down.

2. Because pairs trading *assumes* mean-reversion, the gains cannot be "locked in" from the outset as with identical assets. This is a dilemma for the professional trader, as the upside is "capped" (assuming mean-reversion), but the downside is potentially unlimited (if not mean-reverting). Because of this, risk management is of particular importance in pairs trading, as there sometimes can be large disconnects between two securities due to fundamental reasons that are not possible to predict using statistics.

2.2 Pairs trading approaches in the literature

In its general form, the overarching principle of spread trading rules can be split into a two-step algorithm (Gatev et al., 2006):

1. Find securities that "move together" (ranking pairs)

2. Take a long-short position when they diverge and unwind upon convergence (trading signals)

There are many different approaches to spread trading found in the literature, but all of them centre around the two steps outlined above. In particular, the underlying idea of mean-reversion in relative pricing is always present. Please refer to Krauss (2017) for an in-depth review of the literature on pairs trading strategies.

Two very popular approaches to pairs trading are: 1) the distance approach, and 2) the cointegration approach. In this section, we will present the most important findings in the literature from both camps. Further, we will describe a third approach, which has shown to be very promising and will be the main focus of this thesis: 3) modelling the spread as a stochastic process.

2.2.1 The Distance approach

In a much-cited paper, Gatev et al. (2006) coins what is known as the distance approach to pairs trading. Focusing on the US stock market from 1962-2002, all possible pairs of stocks are ranked based on minimizing the sum of the Euclidian squared distances (SSD) of the normalized price time series in a 12-month formation period. The top 20 pairs in terms of lowest distance are then chosen for trading in the following 6-month period. The equation for the sum of Euclidean squared distances is shown in equation (2.1), in which $P_{A,t}$ is the normalized price series of A and vice versa for B.

$$\text{ssd}(P_A, P_B) = \sum_{t=1}^{T} (P_{A,t} - P_{B,t})^2 \tag{2.1}$$

In the trading period, the spread for each pair is monitored and trades are opened when the spread deviates more than two standard deviations from the average determined in the formation period (Gatev et al., 2006). The threshold is a parameter that can significantly affect trading profits. In particular, there is a trade-off between making an increased number of trades with lower gains per trade (lower threshold), and making fewer trades with higher gains per trade (higher threshold). In practice, the threshold could be set for each pair using data-mining or machine learning methods, but this can result in obvious problems of overfitting.

The distance approach has gained traction due to its quite intuitive way of identifying price time series which stay close to each other. Some additional advantages of the distance approach are that it is easy to implement and model-free, and thus less prone to errors arising from data snooping bias or parameter optimization in-sample (overfitting). A common critique is the sub-optimal nature of the distance approach, in that its ranking can be biased towards pairs with low volatility[2]. Further, as cointegration testing is not part of the distance approach, it is vulnerable to spurious correlations between asset prices that are not in fact fundamentally related (Alexander, 2008).

For the S&P500, Gatev et al. (2006) achieve Sharpe Ratios which were 4 to 6 times larger than that of the market in the period from 1963 to 2002. This is a significant finding, but possibly an outdated one. We suspect that the risk-adjusted returns achieved with statistical-arbitrage style strategies have declined since its inception in the 70's, largely due to three factors:

1. Increased adoption of statistical arbitrage-style strategies
2. Increased data availability and processing capacity
3. Increased speed of execution

Arbitrage opportunities are simply discovered and exploited (much) faster now than in the 1970's. This is also documented in the literature: Do and Faff (2009) prove diminishing returns from pairs trading by continuing the original study of Gatev et al. (2006) through 2008. Looking into the "popular" financial literature, one can also find evidence for why spread trading profits seem to have diminished. Lewis (2014) details how high-frequency trading strategies dramatically changed American markets in the mid-00's, effectively eating into transaction costs for all parties involved in securities trading.

[2] Note that this is not always analytically true, but results from Gatev et al. (2006) indicate such an effect empirically. Refer to Krauss (2017) for calculations and more details.

2.2.2 The Cointegration approach

Vidyamurthy (2004) is among the most cited works on the cointegration approach, presenting a theoretical framework for pairs trading using the statistical concept of cointegration. Cointegration is the formal statistical concept which express that two or more time series never stray too far from each other. Given a set of time series variables $(X_1, X_2, ..., X_k)$, which are integrated of order 1, they are said to be cointegrated if a linear combination of them $(\alpha 1 X_1 + \alpha_2 X_2 + \cdots + \alpha_k X_k)$ is found to be integrated of order 0 (stationary mean-reverting). In the setting of pairs trading, the pair is said to be cointegrated if equation (2.2) have stationary residuals (ϵ_t). Equation (2.2) is often called the cointegrating regression, and the κ signifies the proportion of B which should be held for each unit of A. The log spread[3] $Y(t)$ is then defined by equation (2.3), and μ is the mean which the spread is expected to revert back to:

$$log(P_{A,t}) = \mu + \kappa \cdot log(P_{B,t}) + \epsilon_t \tag{2.2}$$

$$Y(t) = log(P_{A,t}) - \kappa \cdot log(P_{B,t}) = \mu + \epsilon_t \sim N(\mu, \sigma) \tag{2.3}$$

If the two time series variables in a spread are found to be cointegrated, we conclude that spread between them will be mean-reverting. We can thus use a spread trading strategy and go long in one contract and short in the other when the spread deviates sufficiently from the mean, expecting to make a profit when it converges. The stronger the significance of the cointegration test (and longer the time-horizon of the test), the more confident we can be that the relationship also will hold in the future. The Engle-Granger test is used for testing for cointegration among the (log) prices of pairs (Vidyamurthy, 2004), and is presented in section E of the methodology chapter[4]. The trading threshold in the cointegration approach is commonly based on a certain number of standard deviations (σ) relative to the mean (μ), estimated based on the parameters of the cointegrating regression in (2.2).

Among empirical applications, Rad, Low, and Faff (2015) conduct a large-scale study on US CRSP data[5] from 1962 to 2014, combining the cointegration and distance approaches. Following the work of Gatev et al. (2006), pairs of stocks are *first* ranked based on minimizing the 12-month SSD. *Secondly*, the Engle-Granger cointegration test is applied to all pairs and used to filter out those which do not show a statistically significant cointegration relationship. *Third*, the κ (slope coefficient of the cointegrating regression) is used to determine the proportions of units traded in B relative to 1 unit of A. Trading signals are based on a similar approach as Gatev et al. (2006). The results of Rad et al. (2015) are comparable to those of Gatev et al. (2006). One reason for this is presumably that the ranking of pairs is essentially equal in both cases, with Rad et al. (2015) only adding another "layer" of filtering using cointegration tests. Thus, Rad et al. (2015) still suffers from an inferior ranking system which is biased towards low volatility pairs, hurting the profitability of the pairs trade. Some empirical studies using the cointegration approach implement other ranking methodologies than that of Rad et al. (2015). In Dunis, Rudy, Giorgioni, and Laws (2010) and Caldeira and Moura (2013), pairs are tested for cointegration and then ranked based on

[3] Throughout the paper we consistently use $log(x)$ to reference the natural logarithm of x, i.e. $ln(x)$ in some literature.

[4] Other tests for cointegration, such as the Johansen test, is much utilized in general statistical arbitrage methods involving more than two securities. It is not used in this master thesis.

[5] CRSP: Center for Research in Security Prices at Chicago Booth School of Business

the Sharpe Ratio they achieve in the formation period, while Vidyamurthy (2004, p.104-116) focus on the frequency of mean crossovers for the spread in a given period to determine whether it will produce many trades.

2.2.3 The Stochastic Process approach

A number of papers (Elliott, Van Der Hoek, and Malcolm (2005), Do, Faff, and Hamza (2006), Avellaneda and Lee (2010), Bertram (2010), Cummins and Bucca (2012), Liu et al. (2016)) model the spread as a mean-reverting stochastic process[6]. Several models of the log spread are employed, most notably mean-reverting Gaussian Markov chains (state space model) in the discrete case and the Ornstein-Uhlenbeck (OU) process in the continuous case. As we will only use the continuous model in this thesis, we limit our focus to the OU-process.

Let the log spread be defined as $Y(t) = log(P_{A,t}) - log(P_{A,t})$ (i.e. $\kappa = 1$ from equation 2.3). The Ornstein-Uhlenbeck (OU) process is then satisfying the stochastic differential equation (2.4). θ is the mean reversion rate, taken to be strictly positive ($\theta > 0$). σ is the estimated volatility of the process, while μ is the log spread mean. dW is the increment of the continuous-time Wiener process $W(t)$.

$$dY = \theta(\mu - Y(t))dt + \sigma dW \tag{2.4}$$

The parameters can be estimated quite easily using OLS (as shown in appendix H), or by using more advanced methods such as the Kalman filter. Do et al. (2006) points out that there are several advantages in modeling the spread as an OU-process:

1. It captures mean-reversion fully as an OU-process is defined such that spread variable will be normally distributed. Because mean-reversion is the underlying assumption in pairs trading, the OU-process can be an approximation of empirical relationships.

2. Having estimated the parameters of an OU-process, forecasting is simplified. The parameters can be used directly for thresholds in generating trading signals (e.g. k standard deviations, given an estimated volatility), or in ranking spreads according to the expected half-life of mean reversion (see Chan (2013, p.46) for details).

3. The estimation of parameters is tractable (e.g. OLS or Kalman filter), making it feasible for computation on a large scale pairs trading strategy (thousands of pairs, with frequent recalculation).

The stochastic process approach also faces criticism. Both Do et al. (2006) and Cummins and Bucca (2012) note that the OU-process model of the spread is *rigid* and the assumption of a Gaussian distribution is in conflict with the well-known fat tails of financial return data.

Bertram (2010) develop analytic solutions for optimal entry and exit thresholds for pairs trading strategies, assuming that the log spread follows a zero-mean OU-process. Using stochastic calculus

[6] In Krauss (2017), these articles are categorized under "time series approaches". We find it more appropriate to name the approach after its main trait; namely the use of a theoretical stochastic process model.

and general properties of OU-processes, expressions for expected trade length, variance, expected return and Sharpe Ratios are found. Bertram (2010) also acknowledge that the Gaussian assumption of OU-processes is in conflict with the empirical behaviour of financial data, but highlights the usefulness of analytic solutions in studying the dynamics of spreads. Cummins and Bucca (2012) provide the first large-scale application of the framework developed by Bertram (2010), backtesting a total of 861 spreads in energy futures using daily data over the 2003-2010 period. The energy futures contracts considered in the paper are WTI, Brent, heating oil and gas oil traded on the NYMEX and ICE. Several types of spreads are traded, including: calendar spreads, locational spreads and crack spreads. The results are impressive, with daily mean returns in the range of $0.07\% - 0.55\%$ and Sharpe Ratios mostly larger than 2 for the top 10 strategies. Even though the focus of this master thesis somewhat overlaps Cummins and Bucca (2012), there are three important differences: *First*, we utilize intradaily data from a newer period (2015-2018) instead of daily data, and narrow our focus to Brent Crude calendar spreads only. *Second*, we target strategies with much shorter trade lengths than the averages of Cummins and Bucca (2012), which for many strategies was 10-20 days. *Third*, our methodology is inspired by the "doubly mean-reverting" framework of Liu et al. (2016) (detailed in the next paragraph) rather than the optimal thresholds developed by Bertram (2010).

In a particularly promising paper, (Liu et al., 2016) use "high-frequency" 5-minute data and introduce a new framework of "doubly mean-reverting" processes to model the spread. Using a conditional modelling approach, the log spread model is split in two: 1) A long-term spread $L(t)$, and 2) A short-term spread $Y(t)$ which is conditional on the long-term spread $L(t)$. Inspired by Fourier series expansion, this approach seeks to model "local" intraday oscillations around a long-term dynamic spread. Both $L(t)$ and $Y(t)$ are modelled as Ornstein-Uhlenbeck processes, with $Y(t)$ being conditional on $L(t)$. The OU-process parameters are re-estimated every day, to be used for ranking and intraday trading signals in the following trading day. (Liu et al., 2016) rank pairs based on two criteria: 1) highest short-term volatility, and 2) lowest long-term volatility. This ranking method seeks to find spreads that are stable in the long run but exhibit large, short-term oscillations. The empirical results from their backtest, covering spreads of 26 US oil company stocks over approximately three years (June 2013 - April 2015, and 2008), are astonishing. The authors achieve annualized Sharpe Ratios in the range of 3.9 to 7.2 after accounting for transaction costs.

Chapter 3

Data

3.1 Data description and manipulations

3.1.1 An introduction to the data set

Our main dataset consists of historical tick-data for 63 unique Brent Crude futures contracts, traded on the Intercontinental Exchange (ICE) from 2 January 2015 to 25 April 2018. To exemplify, our dataset includes tick data for e.g. ICE BRN AUG-18, the front-month contract at the time of writing; it also includes tick data for ICE BRN MAY-16, a contract that was traded up until expiry on 31 March 2016[1]. We also have daily settlement data for all Brent Crude futures contracts traded on the ICE from 2000 - 2018. The data has been retrieved via the *Montel*[2] energy data API, which in turn is connected to the ICE. In order to engage in spread trading, we need to ensure some degree of simultaneity in the prices of the contracts studied. We solve this by aggregating tick data into 5-minute bars (described in section 3.1.2). Further, we subset the trading hours of Brent Crude futures on the ICE into a 10-hour window from 9:00AM to 7:00PM in order to avoid missing data when liquidity is low (details in section 3.1.4).

By tick data, we are referring to data of all trades executed at the ICE in the given contracts over the time period studied (the left side of figure 3.3 provides an illustration). Although three years and four months of data might seem to be a short period for a backtest, we argue for the opposite: as we are studying short-term trading strategies, the dataset is in fact enormous. To illustrate: in a data set consisting of 20 years of daily observations, one would have a total of $20 \cdot 250 \approx 5000$ unique data points. By aggregating tick data to 5-minute intervals throughout a daily trading window of 10 hours (from 9 AM to 7 PM), our dataset would consist of approximately $103,000$ observations. Thus, in terms of unique data points, our dataset is about 20 times larger than a data set consisting of 20 years of daily data.

The ICE Brent Crude futures contract

The ICE Brent Crude futures contract is a deliverable contract based on EFP[3] delivery with an option to cash settle. All contracts are specified with EFP delivery in a particular month (e.g. December), and

[1] An overview of all contracts studied in this thesis is found in appendix B.

[2] Montel is a data provider and news agency for the European energy markets. Montel is an authorized distributor of ICE data.

[3] Exchange Futures for Physical. Details of the settlement are found on the ICE website.

synthetic positions such as contracts with "yearly delivery" can be financially engineered with a basis in the monthly contracts. The minimum unit of trading is one (1) lot, which is equal to 1,000 barrels. It should thus be noted that the minimum tradeable position size in April 2018 is approximately $\sim 75USD/bbl \cdot 1000bbl = 75,000USD$. The prices are quoted in US Dollars and the minimum tick size is one cent (0.01 USD) per barrel. All open contracts are marked-to-market and settled in cash on a daily basis; the daily settlement price is calculated as the volume-weighted average price (VWAP) of trades during a two minute settlement period from 7:28:00 PM, British Summer Time (BST). The contracts are traded a total of 22 hours each day in London, New York and Singapore. Trading hours in London are from 1 AM to 11 PM, BST.

The Brent Crude futures price dynamics in 2015-2018

The price of crude oil is highly sensitive to fundamental supply and demand factors. Factors impacting the price of crude oil can be factors such as geopolitical tension and events, changes in global demand for petroleum products, increases in technological productivity, the rise of unconventional oil production (e.g. shale) and so forth. Looking at figure 3.1, we observe that the market for Brent Crude futures has been turned "upside down" in the 2015-2018 period. *Firstly*, the front-month contract (ICE BRN M1) went through a large decline from 55 USD/bbl in January 2015 to 28 USD/bbl in January 2016[4], before "steadily" rising to about 74 USD/bbl at the end of April 2018. A similar range of prices is observed in the other contracts (M1-M30), though with lower realized volatility. *Secondly*, the forward curve has gone from being highly contangoed in September 2015 to being highly backwardated in April 2018. This shift in the term structure is also reflected in both log spreads and the implied roll yields of most contracts traded.

As detailed in appendix G, the log spread of futures prices should theoretically be given by the roll yield multiplied with the difference in maturities. Figure 3.1 thus illustrates an important detail for the application of a pairs trading strategy in Brent Crude futures: the mean of the spread seems to be somewhat stable over a short-to-medium term horizon of 1 month but can change substantially over time due to changes in the term structure. We are careful about drawing early conclusions (tests for cointegration over both the long- and short-term are presented in section 3.2.2), but the overview provided in figure 3.1 serves as motivation to take a short-term approach to spread trading in Brent Crude futures.

3.1.2 Data aggregation

The original data series contains ∼ *235 million* trades in total, in the period from January 2015 to April 2018. Individual contracts are traded as much as tens of millions of times during their lifespan. The sheer size of the dataset, combined with the event-driven nature of tick data, makes the dataset difficult to analyze. It also poses two main problems in the context of pairs trading. *Firstly,* simultaneity in prices is necessary to ensure that spreads are indeed tradeable. Because tick data is not comparable across different contracts (ticks do not occur simultaneously in both contracts), we need to aggregate the data in

[4] To put the oil price crash of early 2016 in perspective: in late June 2014, the front-month contract was trading at 115 USD/bbl. 18 months after, in January 2016, the oil price had fallen by 87 dollars since its peak - a decline of more than 75%

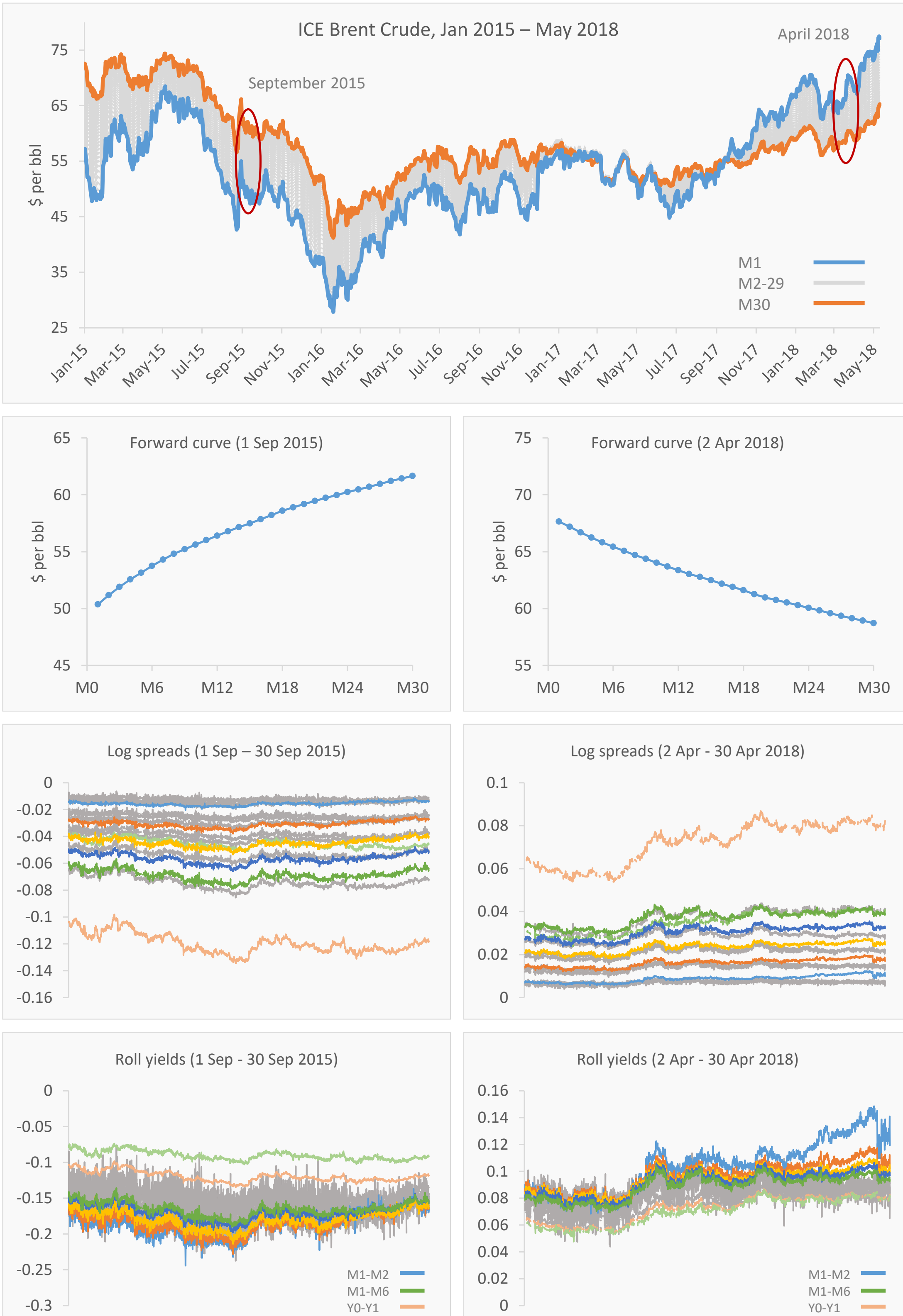


Figure 3.1: ICE Brent Crude futures overview, January 2015 to May 2018

some manner. *Secondly*, as historical order book data (bid-ask prices and volumes) has not been available for this thesis, we argue that a high-frequency approach with very granular time resolution or a tick based approach would not give meaningful results, even for highly liquid securities (e.g. sub-1-minute resolutions). Market microstructure would almost certainly have a big impact on the results, which we would not be able to control for without order book data.

We aggregate raw tick-data into a format more suitable for analysis and backtesting. In the literature, data logged over the course of some predefined time interval (e.g. every 5 or 10 minutes) seems to be the norm, and we choose to follow this norm as well. Our procedure involves looping through all rows of tick data and in order to map the aggregated data onto a new time axis as illustrated in figure 3.3, with resulting 5-minute bars on the right side. The resulting time series include the following information for each bar: opening price, closing price, volume-weighted average price (VWAP), traded volume and number of trades. As an example, the ”close” price for an interval is simply the last traded price in the interval. The VWAP, on the other hand, is the volume-weighted average price for all trades in the interval. The curious reader may verify the procedure of creating open and close prices for an interval by comparing tick data and aggregated data in figure 3.3.

We aggregate data into bars of 5-, 10-, 30- and 60-minute resolution. We perform this aggregation for all contracts, resulting in 63 individual time series for each specific time resolution.

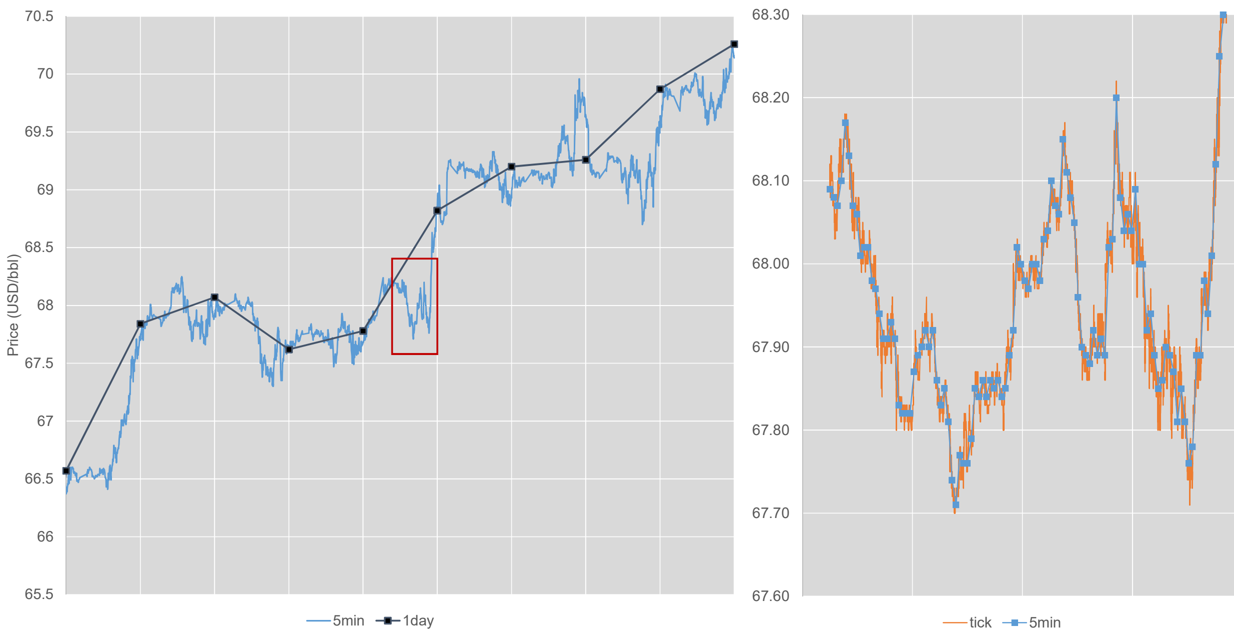
Artificial volatility in spreads

The use of aggregated data with open/close prices can cause ”artificial” volatility in the log spreads. The reason for the phenomenon is that the closing and opening prices of a 5-min interval (or any other choice of an interval) might not be simultaneous in both contracts. The close in one contract might be a trade twenty seconds ago, while for the other contract it might be a trade one second ago. This does not imply that there is a change in the tradable spread between them, which could only be identified by looking at bid-ask data. While we acknowledge the risk of basing some of our trading decisions on short-term volatility that might not be tradable, this can be mitigated by setting appropriate thresholds for the strategy (details in methodology section 4.2.2).

3.1.3 Rolling of contracts and expiry-related concerns

Absolute and relative contracts

In this thesis, we will refer to contracts on both an ”absolute” and a ”relative” basis. By *relative*, we refer to the contract which is currently in a given distance from maturity, e.g. the front month is referred to as ICE BRN M1. When referring to a time series of e.g. ICE BRN M1, this is the continuous time series of rolled contract positions such that it always represents the front month, depending on the schedule for rolling. By *absolute*, we refer to a specific contract of a given maturity, e.g. ICE BRN MAY-18. These contracts are the actual contracts traded on the exchange. By treating contracts on an absolute basis in our backtesting model, we avoid the pitfalls related to including false returns across roll dates in our performance metrics. When reaching the chosen roll date of a contract, all open positions are sold out.



(a) Daily settlement data vs. 5-min data

(b) 5-min data vs. tick-data

Figure 3.2: Comparison of time resolutions. Figure 3.2a show daily and 5-min data for the period 2-Jan-18 to 15-Jan-18. The red box indicate the time-interval for figure 3.2b.

Rolling procedures

Selecting the exact time for contract rolling might seem trivial, as one could just trade contracts until they expire and then move on to the next. We choose to roll over contracts on the penultimate day of trading, as there are several unfortunate effects related to last-day trading in Brent Crude futures. *Firstly*, any open positions at the time of expiry are physically settled unless the option to settle in cash is exercised. Delivery is unwanted from a short-term trading strategy perspective, as the cash-settlement price is not published until the next trading day following expiry. For this reason, we stay clear of final settlement both in cash and in physical crude oil. *Secondly*, open interest and volumes in the front-month contract decreases rapidly towards expiry after topping out at some point during the final two months of the contract's lifespan. This makes trading more risky closer to expiry, as liquidity quickly runs thin. The development of open interest and volume in contracts approaching maturity for the ICE BRN MAY-18 contract over the last year of trading is shown in figure 3.4a and open interest for M1-M6 in its last month of trading is shown in figure 3.4b. Similar stylized patterns as these are found in most contracts when they are "in front" (traded as M1).

3.1.4 Choosing a subset of the data for trading

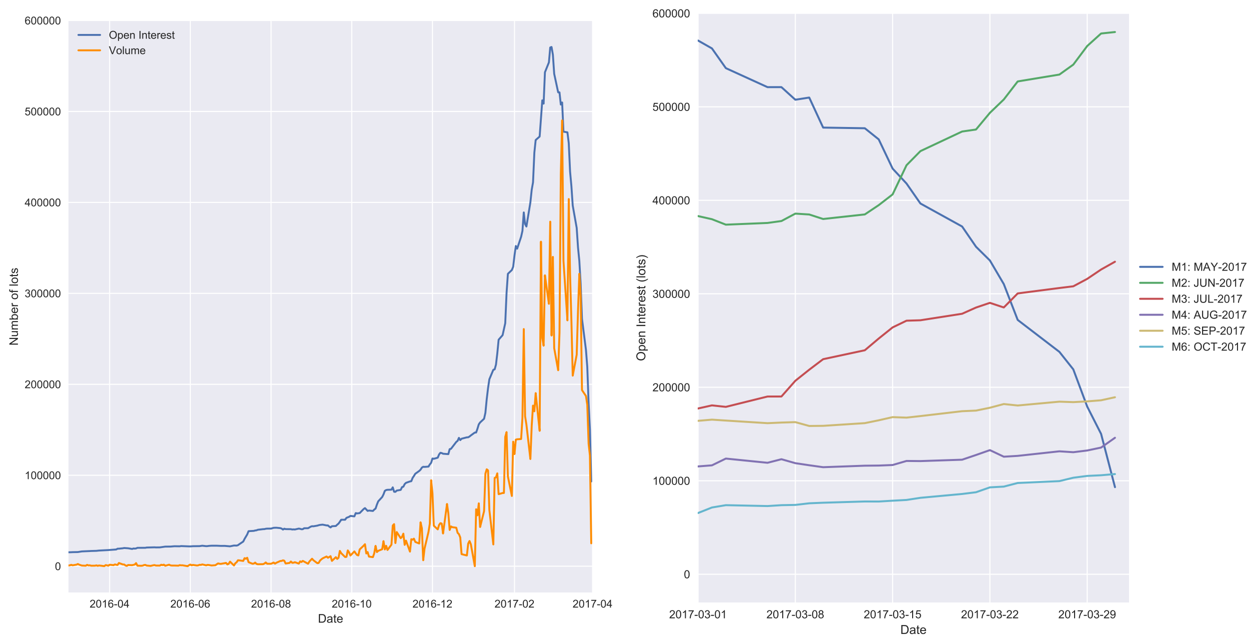
Why we subset the aggregated data

As briefly mentioned in earlier sections, missing data is a problem which occurs when aggregating tick-data into e.g. 5-minute bars and there are no trades in the interval. In fact, this is a fundamental problem

TradeInDay	TradeID	TradingTime	Price	Volume	ContractName
4086	1211784482	2018-01-09 09:00:00	68.09	2	ICE BRN MAR-2018
4087	1211784481	2018-01-09 09:00:00	68.09	1	ICE BRN MAR-2018
4088	1211784478	2018-01-09 09:00:00	68.10	1	ICE BRN MAR-2018
⋮	⋮	⋮	⋮	⋮	⋮
4379	1211785472	2018-01-09 09:04:16	68.08	1	ICE BRN MAR-2018
4380	1211785468	2018-01-09 09:04:16	68.08	1	ICE BRN MAR-2018
4381	1211785496	2018-01-09 09:04:30	68.08	1	ICE BRN MAR-2018
4382	1211785495	2018-01-09 09:04:30	68.08	4	ICE BRN MAR-2018
4383	1211785494	2018-01-09 09:04:30	68.08	1	ICE BRN MAR-2018
4384	1211785501	2018-01-09 09:04:51	68.07	1	ICE BRN MAR-2018
4385	1211785568	2018-01-09 09:05:00	68.08	4	ICE BRN MAR-2018
4386	1211785569	2018-01-09 09:05:00	68.08	1	ICE BRN MAR-2018
4387	1211785570	2018-01-09 09:05:00	68.08	1	ICE BRN MAR-2018
4388	1211785567	2018-01-09 09:05:00	68.08	2	ICE BRN MAR-2018
⋮	⋮	⋮	⋮	⋮	⋮
6075	1211791231	2018-01-09 09:24:24	68.13	1	ICE BRN MAR-2018
6076	1211791235	2018-01-09 09:24:48	68.13	1	ICE BRN MAR-2018
6077	1211791234	2018-01-09 09:24:48	68.13	1	ICE BRN MAR-2018
6078	1211791247	2018-01-09 09:24:49	68.13	1	ICE BRN MAR-2018
6079	1211791246	2018-01-09 09:24:49	68.14	1	ICE BRN MAR-2018
6080	1211791250	2018-01-09 09:25:09	68.13	1	ICE BRN MAR-2018
6081	1211791252	2018-01-09 09:25:11	68.13	1	ICE BRN MAR-2018
6082	1211791254	2018-01-09 09:25:15	68.13	1	ICE BRN MAR-2018
⋮	⋮	⋮	⋮	⋮	⋮
7486	1211794109	2018-01-09 09:29:58	68.05	1	ICE BRN MAR-2018
7487	1211794108	2018-01-09 09:29:58	68.05	1	ICE BRN MAR-2018
7488	1211794105	2018-01-09 09:29:58	68.06	1	ICE BRN MAR-2018
7489	1211794106	2018-01-09 09:29:58	68.06	1	ICE BRN MAR-2018
7490	1211794107	2018-01-09 09:29:58	68.06	1	ICE BRN MAR-2018
7491	1211794104	2018-01-09 09:29:58	68.06	1	ICE BRN MAR-2018
7492	1211794115	2018-01-09 09:29:59	68.05	1	ICE BRN MAR-2018
7493	1211794175	2018-01-09 09:30:00	68.07	1	ICE BRN MAR-2018
7494	1211794176	2018-01-09 09:30:00	68.07	1	ICE BRN MAR-2018
⋮	⋮	⋮	⋮	⋮	⋮

TradingTime	Open	Close	VWAP	Volume	N Trades
2018-01-09 09:00:00	68.09	68.07	68.1015	515	299
2018-01-09 09:05:00	68.08	68.07	68.052	618	381
2018-01-09 09:10:00	68.07	68.11	68.1179	848	495
2018-01-09 09:15:00	68.1	68.17	68.1348	604	419
2018-01-09 09:20:00	68.17	68.14	68.1514	595	400
2018-01-09 09:25:00	68.13	68.05	68.0799	2594	1413
2018-01-09 09:30:00	68.07	68.07	68.0609	1710	786
2018-01-09 09:35:00	68.06	68.02	68.05	911	451
2018-01-09 09:40:00	68.01	68.02	68.0122	1209	599
2018-01-09 09:45:00	68.02	68.01	68.0079	395	171
2018-01-09 09:50:00	68.02	67.98	68.0129	797	349
2018-01-09 09:55:00	67.98	67.96	67.9852	485	277

Figure 3.3: Example of aggregation process of tick data into 5-min data, for the 9th of January 2018 in the ICE BRN MAR-2018 contract. Tick data on the left and 5-min data on the right. TradeInDay is a chronological counter showing the trader's position in the particular day. TradeID is generated from the data vendor and is not necessarily chronological in a single contract. VWAP is the volume-weighted average price in the 5-min interval.



(a) Open interest and daily volume in MAY-17 contract

(b) Open interest for M1-M6 in March 2017

Figure 3.4: Example of long-term development in open interest of the ICE BRN MAY-17 contract, and comparison of open interest in M1-M6 when MAY-17 is front contract. Number of lots on vertical axis.

of all trading strategies when handling high-frequency intraday data. Uncertainty in the time domain (when the next trade will happen) is not captured in most models used in the pairs trading literature[5], and the problem of missing data is commonly avoided by subsetting the data or interpolating it in order to remove blanks. In Liu et al. (2016), the authors remove 5 stocks from their data set due to high numbers of blanks and subsequently interpolate over blank data points for the remaining stocks. We choose a middle ground: we remove contracts with too high numbers of blanks, but we do not interpolate over remaining blanks. The treatment of blanks in our backtest is explained thoroughly in section 4.3.5.

Subsetting procedure

Subsetting is approached in a systematized manner, in which we study the relative number of missing data points of different subsets. The twelve closest monthly contracts (M1-M12), the half-year contract (H1) and the two closest yearly contracts (Y0-Y1) are examined over the course of four one-month periods: April 2015, June 2016, November 2017 and February 2018. We start out looking at 60-minute data throughout the entire trading hours (1 AM to 11 PM) and then narrow it down both in terms of time resolution and trading hours. This process is illustrated in figure 3.5. By averaging the four periods, we see

[5]The log-Brownian paradigm for modelling securities prices is both easy to apply and it produces sensible results, but it fails to capture a crucial property of intraday price movements, namely uncertainty in the time domain. Whereas daily data for most securities have no time uncertainty (price data arrives at a constant rate of once every trading day), this is not the case for intraday data. The arrival of a tick, or a trade, could arguably be modelled as a Poisson process of some given intensity, with the realized prices being drawn from some independent distribution. This type of model is out of scope for this master thesis, but the interested reader is referred to Rogers and Zane (1998) for a detailed description of such a model.

that M1-M6, Y0, H1 and Y1 achieve filling-rates well above 90%, and for M1-M5 the percentage of missing data points is approximately 0%. By evaluating a large number of subsets, both in terms of time resolution and trading hours, we have found 5-minute data from 9 AM to 7 PM to yield satisfactory filling-rates. Our resulting 5-min dataset includes 120 data points per day, where the time stamp indicates the opening time of the interval.

In selecting contracts for trading, our objective is twofold. We want to: 1) keep periods with missing or no data at the lowest level possible to ensure sufficient robustness in the backtest; while also 2) maintaining the most detailed time resolution possible, to ensure sufficient granularity for intraday trading. This promotes an "uncertainty principle": we want both *robustness* and *granularity*, but must compromise on one in order to achieve the other. We know from section 3.1.3 that volume and open interest in Brent Crude futures increase rapidly during the last six months of a contract's lifespan. From this, we hypothesize that distant-maturity contracts might not be as frequently traded as i.e. M1 or M2. Combined with the subsetting approach illustrated in 3.5, we choose to trade the relative contracts M1-M6, Y0-Y1 and H1 (an overview of all pairs considered are found in appendix C). When looking at the M7-M12 contracts, intraday liquidity is rapidly decreasing with a high number of missing data points.

3.1.5 Summary of data aggregation and subsetting

Before continuing with the empirical analysis of the data set, we summarize our choices of data aggregation and subsetting:

1. To achieve price simultaneity, tick data has been aggregated to 5-minute bars. Also, because we do not have historical order book data, we argue that a very granular resolution (e.g. sub-1-minute) would not be meaningful due to the potential impact of market microstructure.

2. Time resolution is set to 5-minutes, as this allows us to test a true ”intraday” strategy.

3. The considered pairs for pairs trading are combinations of the relative contracts M1-M6, Y0-Y1 and H1 (an overview of pairs are found in appendix C), as these are the most liquid contracts.

4. We limit trading hours to 9:00 AM until 7:00 PM, BST. We do this to avoid large periods without trade data in the contracts selected for trading.

3.2 Empirical study of the intraday data set

In this section we conduct an empirical study of the data subset chosen in section 3.1.4 (5-min resolution between 9AM and 7PM for pairs listed in appendix C). We give a detailed description of return distributions, autocorrelation in returns, Conditional Value at Risk (CVaR) for long positions and the covariation of contracts (both correlation and cointegration). By comparing 5-min data with daily data, we highlight several interesting features that in our opinion justifies a thorough backtest of an intradaily spread trading strategy.

We study contract time series on a *relative* basis for our empirical study. We study log returns, and we exclude log returns across roll-dates to avoid false returns arising from the rolling of contracts. Log

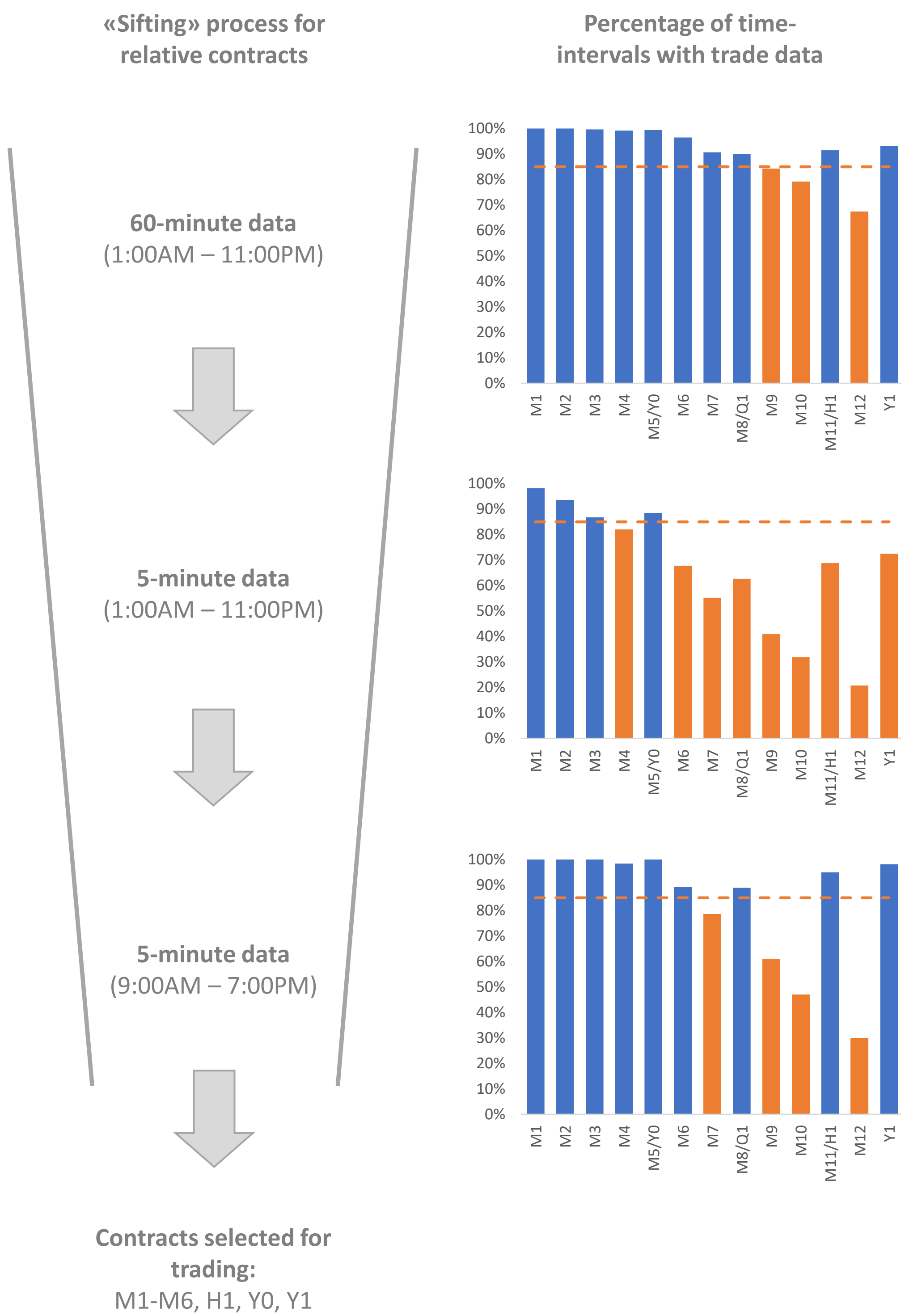


Figure 3.5: Selecting relative contracts for trading

returns for contract i are calculated using equation (3.1). We observe 'Close' prices for 5-min data and 'Settlement' prices for daily data. We use the daily settlement price as this is the *de facto* daily closing price, used by brokerages when marking positions to market.

$$R_{i,t} = log(P_{i,t}) - log(P_{i,(t-1)}) \tag{3.1}$$

When analyzing 5-min bars, we exclude overnight log returns (in our case from 7 PM until 9 AM the following day). Overnight returns are excluded because they represent returns over a different time interval than the 5-minute intervals we have partitioned our data into. This is a subtle detail, but an important distinction between daily and intradaily data. For daily data, data points are evenly spaced out with one trading day between them; for intradaily data, overnight returns differ from returns over e.g. a 5-min interval. Further, log returns related to missing data points are considered missing, and thus excluded from the following analysis [6]. A note of caution: aggregating 5-min return series will not yield true daily returns due to the treatment of blanks and overnight returns. Despite this, the partitioning of the data into 5-min intervals gives us a solid basis for describing the return characteristics over short time intervals.

3.2.1 Statistical properties of Brent Crude futures contracts

Descriptive statistics

In this subsection, we present descriptive statistics for the log returns of 5-min data and daily data. Descriptive stats for 5-min data are found in table 3.1, and for daily data in table 3.2. We highlight the following from the tabulated data:

1. **Negative intraday returns:** Mean values for intraday returns are all negative. This is in contrast to daily returns, which show positive mean returns for the entire period studied. The number of observations in both samples is high. From this, we conclude that on average, intraday returns have been negative while overnight returns have been positive[7].

2. **Wide confidence bounds:** Confidence bounds are quite wide relative to the mean for both 5-min returns and daily returns. The 95% CB for daily returns in the M1 contract ranges from $\sim -32\%$ to $\sim 47\%$, on an annualized basis.

3. **Non-Gaussian return distributions:** Jarque Bera for both 5-min and daily data are very high, indicating that both distributions are non-Gaussian. 5-min JB is much much higher than daily JB. Further, 5-min returns show low skewness but very high excess kurtosis. In other words, the 5-min return distribution has very fat tails. For daily data, there is a slight positive skew for all contracts.

4. **Missing values:** The number of observations is significantly lower for H1 and Y1 than for the rest of the contracts studied. This is caused by blanks in the dataset, which leads to undefined log returns and fewer observations. The return over periods with missing data is not included. This is

[6] Log returns are calculated *before* removing the missing data. If blanks were removed first, it would certainly impact the distribution of 5-min log returns as the time-delta no longer would be exactly 5 minutes for all remaining data points.

[7] Overnight returns are here defined as the returns from 7.00PM to 9.00AM the following trading day.

undeniably a source of error, as these returns might "correct" any new information arriving during the blank period. However, it should be noted that the number of total observations is very large and thus we consider these effects to be small.

	5-minute			**Intraday aggregated***		
Contract	Mean	CB	SD	Mean	CB	SD
M1	-0.0001987%	0.0010407%	0.1691236%	-0.0236%	0.1238%	1.8449%
M2	-0.0001693%	0.0010200%	0.1657124%	-0.0201%	0.1214%	1.8077%
M3	-0.0001269%	0.0010087%	0.1632578%	-0.0151%	0.1200%	1.7809%
M4	-0.0001514%	0.0010130%	0.1616730%	-0.0180%	0.1205%	1.7636%
M5	-0.0001922%	0.0010420%	0.1619150%	-0.0229%	0.1240%	1.7663%
M6	-0.0002995%	0.0010796%	0.1619711%	-0.0356%	0.1285%	1.7669%
Y0	-0.0003001%	0.0009500%	0.1529322%	-0.0357%	0.1131%	1.6683%
H1	-0.0005757%	0.0010838%	0.1522441%	-0.0685%	0.1290%	1.6608%
Y1	-0.0004800%	0.0009431%	0.1384623%	-0.0571%	0.1122%	1.5104%
				5–minute		
Contract	N.obs.	Min	Max	Kurtosis	Skewness	Jarque-Bera
M1	101 458	-3.64%	2.37%	10.82	-0.07	495 166
M2	101 405	-3.45%	2.33%	10.52	-0.06	467 657
M3	100 634	-3.39%	2.27%	10.16	-0.06	432 608
M4	97 853	-3.07%	2.18%	9.15	-0.06	341 390
M5	92 756	-2.92%	2.17%	8.55	-0.04	282 315
M6	86 473	-3.00%	2.05%	8.62	-0.05	267 623
Y0	99 545	-2.40%	2.05%	8.51	-0.05	300 223
H1	75 801	-2.00%	1.76%	6.66	-0.04	140 182
Y1	82 801	-1.76%	1.59%	6.17	-0.01	131 262

Table 3.1: Descriptive statistics for log returns of *intradaily 5-min* data. Overnight returns are excluded. CB is the 95% confidence bound of the mean. *Note: Aggregated over the (T = 120) 5-min periods from 9:00AM to 7:00PM, in similar fashion as when annualizing daily returns. This return only measures the *intradaily* component of daily returns, as overnight returns are excluded from the 5-min dataset.

Distributions of returns in different time resolutions

From the empirical finance literature, a well-known *stylized fact* is that daily return series have fat tails. Figure 3.6 show histograms of 5-min (A) and daily (B) log returns in the ICE BRN M1 *relative* contract. We observe in histogram A that the distribution of 5-min returns has (much) fatter tails than the distribution of daily returns. It is clear that 5-min returns arrange themselves in a highly leptokurtic fashion. Daily returns also show a leptokurtic tendency. This resonates well with the high levels of excess kurtosis observed in tables 3.1 and 3.2.

We take a closer look at the return distribution of the front-month contract by considering four different time resolutions. For this purpose, we construct Quantile-Quantile (Q-Q) plots, presented in figure 3.7. Looking at plots A through D, the general conclusion is that intraday data have much fatter tails than what would be expected from a normal distribution. We also notice that return distributions seem to become more leptokurtic the shorter the time resolution and that the 30- and 60-min data presented in plots B and C nicely bridges the gap between the 5-min data and the daily data already discussed. Explanations for the highly leptokurtic nature of short-term return distributions can be many. Price shocks

	Daily			Annualized*		
Contract	Mean	CB	SD	Mean	CB	SD
M1	0.0312%	0.1587%	2.3677%	7.79%	39.68%	37.44%
M2	0.0279%	0.1551%	2.3145%	6.97%	38.79%	36.60%
M3	0.0250%	0.1514%	2.2588%	6.25%	37.85%	35.71%
M4	0.0222%	0.1476%	2.2017%	5.55%	36.90%	34.81%
M5	0.0196%	0.1438%	2.1450%	4.89%	35.95%	33.92%
M6	0.0173%	0.1402%	2.0909%	4.32%	35.04%	33.06%
Y0	0.0116%	0.1407%	2.0998%	2.89%	35.19%	33.20%
H1	0.0005%	0.1219%	1.8186%	0.13%	30.47%	28.75%
Y1	-0.0080%	0.1085%	1.6181%	-2.01%	27.12%	25.59%
				Daily		
Contract	N.obs.	Min	Max	Kurtosis	Skewness	Jarque-Bera
M1	855	-8.80%	11.13%	2.16	0.36	185
M2	855	-8.66%	10.45%	2.14	0.35	181
M3	855	-8.60%	9.87%	2.11	0.33	174
M4	855	-8.55%	9.67%	2.09	0.32	170
M5	855	-8.49%	9.57%	2.08	0.30	167
M6	855	-8.40%	9.51%	2.08	0.29	167
Y0	855	-8.66%	14.97%	4.39	0.59	736
H1	855	-8.16%	9.98%	2.89	0.33	313
Y1	855	-7.73%	7.73%	2.57	0.20	241

Table 3.2: Descriptive statistics for log returns of *daily* data. In contrast to table 3.1, this include overnight returns. CB is 95% confidence bound of mean. *Note: Annualized using the standard approach of multiplying daily log returns with $\sqrt{T}$ and std.dev. with $\sqrt{T}$ (trading days assumed to be $T = 250$).

might pose the most fruitful explanation: the market for crude oil futures is (very) liquid, and as a consequence, any new information is quickly assimilated in the market. In the event of a price shock, we infer from the distributions of returns that the most extreme movements in prices happen over a short period of time. The relative size of the shock is much larger for 5-min data than for daily data, simply because on average, prices change less in 5 minutes than in a day. This is a possible explanation for the highly leptokurtic return distribution.

We have studied the other relative contracts in a similar fashion, from which identical insights can be drawn.

Time series and autocorrelation of returns and absolute returns

Figure 3.8 show time series of log returns and absolute values of log returns for 5-min (A) and daily (B) data. By looking at the top panel for A and B, we can verify the shape of the return distributions previously observed in figure 3.6. For 5-min data, we verify that returns usually fall in the range of $\pm 0.5\%$, but sometimes experience large 'shocks' with returns of e.g. $< -2\%$ in a single 5-minute interval. From the bottom panels in both A and B, we notice that *volatility clustering* is an evident feature of returns - this is a potential risk factor in spread trading. However, as we will later show in figures 3.11 and 3.12, the relationship between returns in Brent Crude futures contracts of different maturities is quite strong. This serves as a mitigating factor on the aforementioned risk. We also note that the second half of our sample seems to have lower variability in returns than the first half. This may also be verified from the chart of

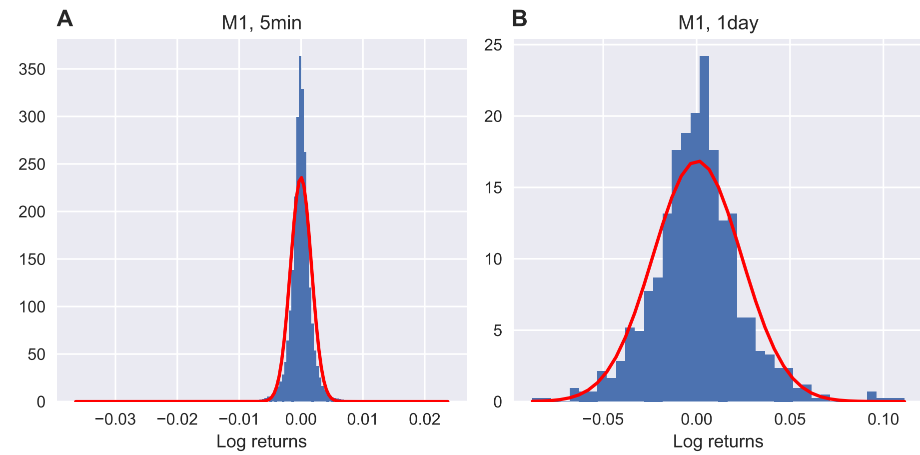


Figure 3.6: Histograms of 5-min and daily returns for the *relative* contract ICE BRN M1. The red line is the Gaussian fit of the data observed. Note that the scale of the x-axis is not the same for the two histograms.

daily data over the period from January 2015 to April 2018 in section 3.1.

To look closer at the time dependence of returns and volatility, we plot the Autocorrelation Function (ACF) for both returns and absolute returns in figure 3.9. In plots A and C, we note that persistence in returns seems to be low or non-existent (correlation coefficients for lags are small in absolute terms and does not follow any notable pattern). Looking at the 5-min ACF in A, we note that several lags achieve correlations that fall outside of the confidence boundary. To us, however, this seems to be merely a consequence of the high number of lags studied (600) - some lags will exceed the confidence bound out of pure chance. Further, as correlation coefficients are small in magnitude (~ 0.02 at most), this indicates that there is little persistence in 5-min returns. From C, we note a very slight single-day persistence of returns for the daily data, as the first lag shows a statistically significant correlation coefficient of negative ~ 0.075. This is however low, and thus the ACF plots do not indicate significant persistence in returns for either 5-min or daily return series. Because the ACF plot does not verify the joint hypothesis that all correlation coefficients for a given number of lags are zero, we also use the Ljung-Box test on the return data. Test results for the ICE BRN M1 contracts are found in table 3.3. Interestingly, we now find the null hypothesis of no autocorrelation in 5-min return series to be rejected as p-values for all lags are significant at a 1%-level. ($p<0.01$). For the daily return series, we still have no concluding evidence for autocorrelation for lags more than 1 day.

The lack of autocorrelation in returns does not mean that returns are independent over time - by plotting the ACF of absolute values of returns we see in B and D that non-linear time dependencies are highly present. Because absolute returns (or squared returns) are linked to volatility, this confirms the presence of volatility clustering in both 5-min and daily data. The shape of the 5-min ACF plot in B looks strange at first but has also been found in the early literature on volatility persistence in Andersen and Bollerslev (2014, Fig. 4, p.123). Looking closer, we find that all the tops of the ACF in B is centred around multiples of 120, which is the number of 5-minute periods each day in the sample we study. So what does B really tell us? It tells us that for 5-min data, persistence in absolute returns is strongest for lags spaced

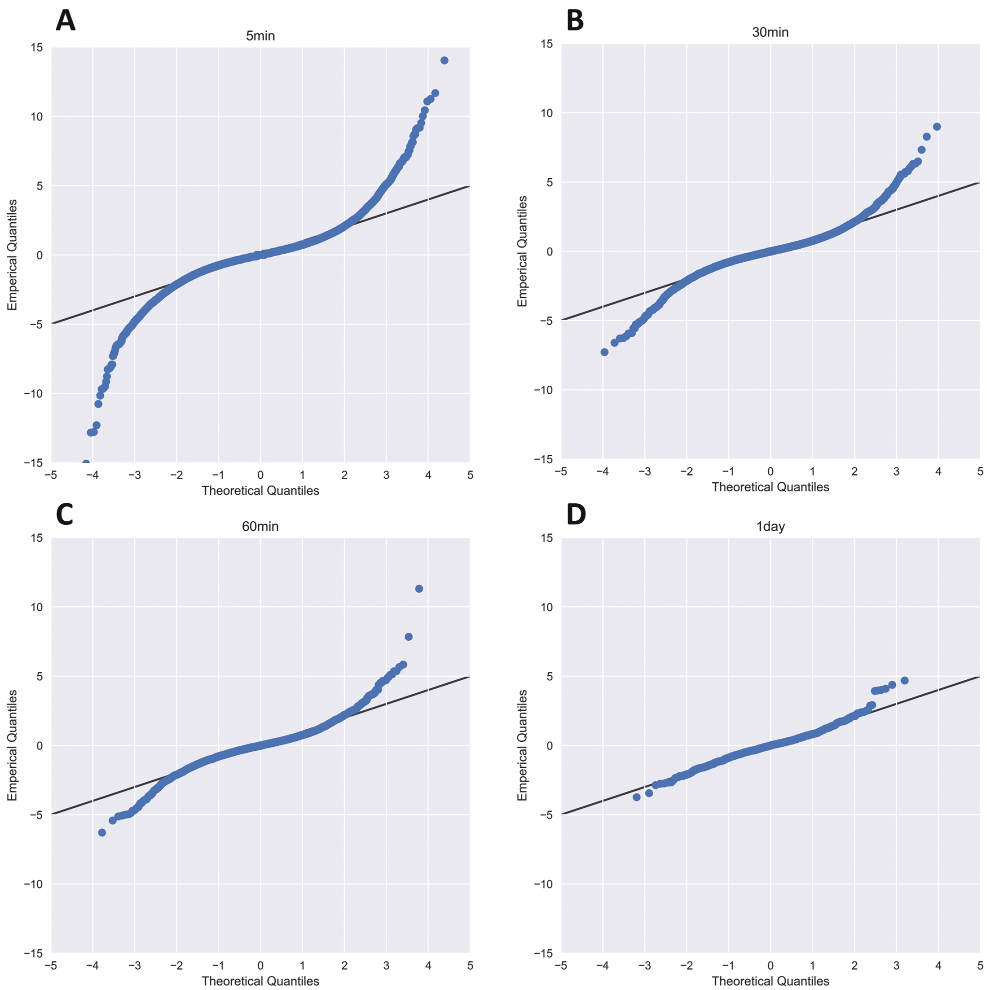


Figure 3.7: Q-Q plots for 5-min, 30-min, 60-min and daily time resolutions, for the relative ICE BRN M1 contract. All data series have been transformed into standard normal variables for easier comparison.

exactly one trading day apart. From D, we conclude that there exists significant persistence in absolute daily returns. We also note that the periodic effect observed for 5-min data has disappeared.

Conditional Value at Risk (CVaR) of long positions in Brent Crude futures

The CVaR metric is intrinsically linked to the time horizon and resolution of the data studied. Figure 3.10 shows the sensitivity of CVaR for different quantile-levels α and different time resolutions.

Firstly, we observe that the CVaR is relatively equal across contracts. The closer to maturity, the higher the CVaR. As seen in section 3.2.1, the tails of 5-min returns are considerably "fatter" than that of other time resolutions. A disproportionate amount of losses occur over very short time-intervals for Brent Crude futures, which further highlights the importance of risk management in intradaily trading strategies.

	Lags = 3		Lags = 5		Lags = 20		Lags = 40	
	5-min	*daily*	*5-min*	*daily*	*5-min*	*daily*	*5-min*	*daily*
Q-statistic	37.154	6.4742	37.679	7.3414	76.706	26.717	111.85	38.594
p-value	0.0000*	0.0907	0.0000*	0.1965	0.0000*	0.1434	0.0000*	0.5336

Table 3.3: Ljung-Box test applied to the ICE BRN M1 time series. The joint null hypothesis are rejected for all 5-min data tests, and we conclude that autocorrelation is present. Statistically significant results at the 1% level is indicated by: *p<0.01. The maximum number of lags of 40 are based on the default specification lags = $min(\lfloor n/2 \rfloor - 2, 40)$ in the Stata 15 software package (StataCorp, 2017). In our case $n >> 84$, and thus the default choice is lags = 40.

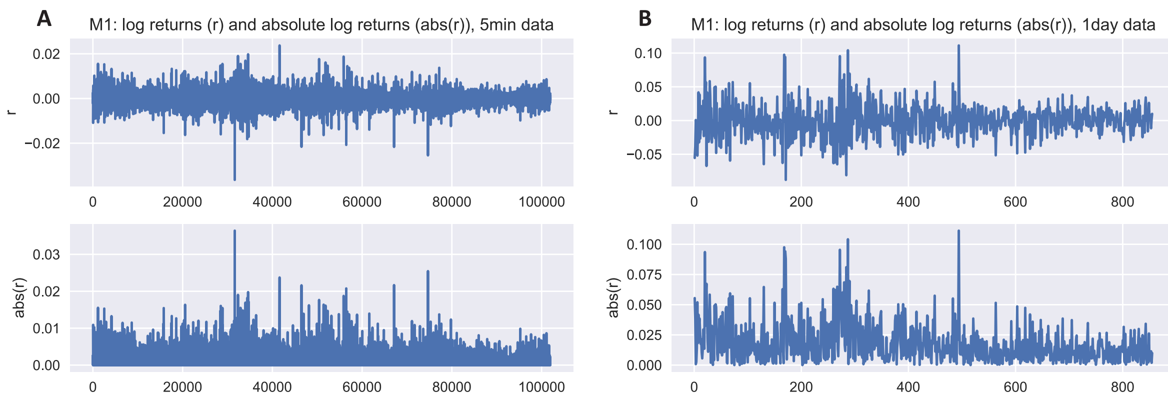


Figure 3.8: Comparison of log returns and absolute log returns for 1-day data and 5-min data, for the relative ICE BRN M1 contract.

Covariation of contracts

In figure 3.11 and 3.12, we present scatter plot matrices for 5-min and daily log returns for the M1-M6, Y0, H1 and Y1 contracts. We also show the return distributions on the diagonal. Clearly, returns of different contracts exhibit some strong form of interdependency. This further motivates our spread trading backtest. As explained in appendix G, the log returns of calendar spreads are related to roll-yields (cost of storage and convenience yield) and spot price returns. Because the spot price is common for both contracts, the relationship between returns for the contracts should be very strong. Comparing the scatter plot matrices, we observe that the relationship between log returns of contracts of different maturities are strong for both 5-min and daily data. For 5-min data, however, correlation[8] of returns is lower than for daily data. From the return distributions, we know that 5-min returns have fatter tails than daily returns, and we believe this might be part of the explanation for the lower correlation. The other part of the explanation might be that for 5-min intervals, details of micro market structure and order flow become important factors affecting the price. If one contract is being accumulated aggressively while another is being dumped, chances are that their prices will be affected accordingly. In other words, price movements over very short time periods might be driven by other factors than the underlying fundamentals.

A natural question when studying the covariation of contracts is to look for lead/lag relationships. We restrict our analysis to the bivariate case, and use a Granger causality test to check for statistically

[8] Correlation matrices for both 5-min and daily data are found in appendix D.

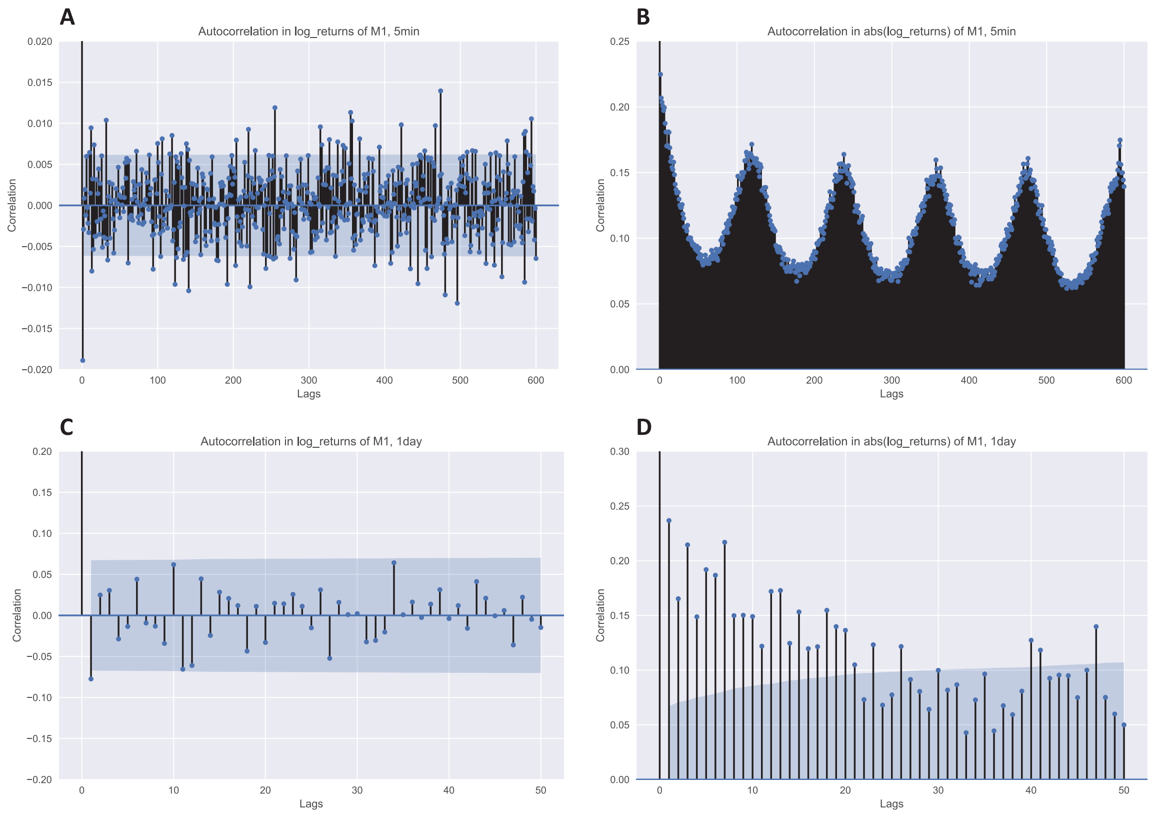


Figure 3.9: Plots of the Autocorrelation Function (ACF) for log returns (left) and absolute log returns (right) in 5-min (top) and daily data (bottom), for the relative ICE BRN M1 contract. 95% confidence bounds are shown in shaded blue background.

significant lead/lag relationships in the 5-min log returns series. The results for all pairs are found in table 3.4 and the test procedure is shortly described in the captions of the table. From the p-values, we see that all test statistics are significant at a 5% significance level, in both directions. The null hypothesis is thus rejected, and we conclude that lagged values of x are shown to explain some of the variations in y. Because all pairs show bi-directional Granger causality, we interpret this as another confirmation of the strong relationship between the contracts.

We argue that this makes an interesting case for short-term spread trading in Brent Crude futures: the underlying fundamental relationship between contracts is very strong, as will be further demonstrated in section 3.2.2, but temporary deviations from this relationship may occur due to idiosyncratic factors affecting contracts in the short-term. This is apparent from the lower correlation of 5-minute returns relative to daily returns.

3.2.2 Cointegration of contracts

Long-term relationship of contracts

We perform the Engle-Granger routine to test for cointegration in Brent Crude calendar spreads. We use daily data of log spreads in the period from January 2000 to April 2018.

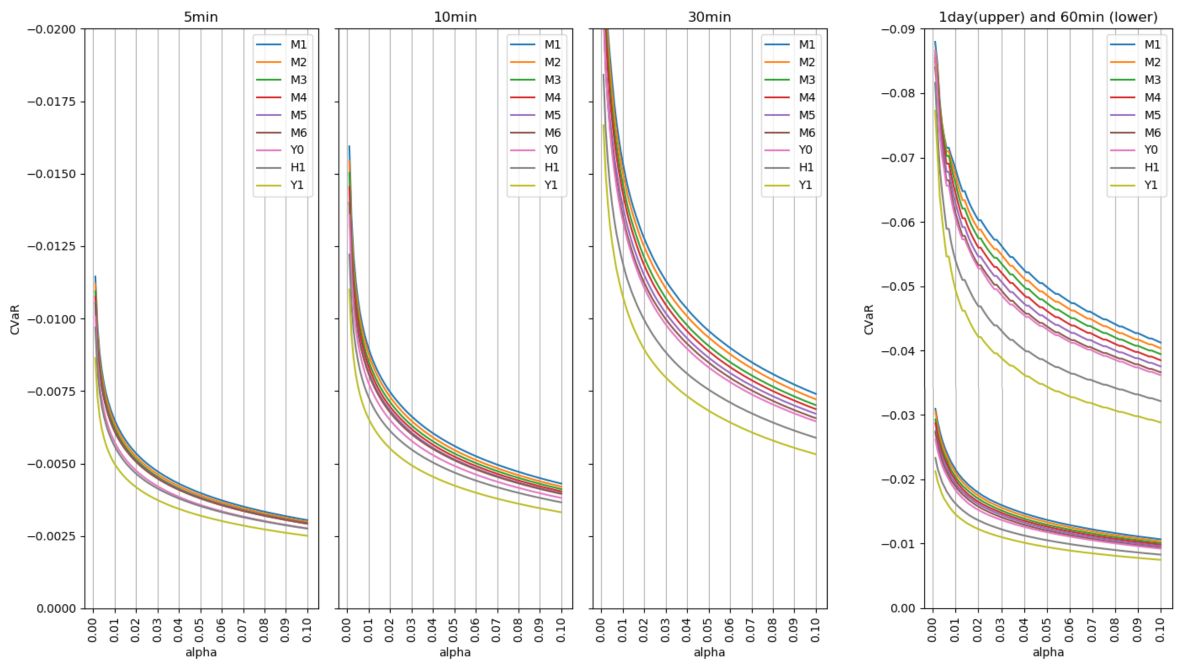


Figure 3.10: Sensitivity analysis for CVaR when changing the percentile alpha.

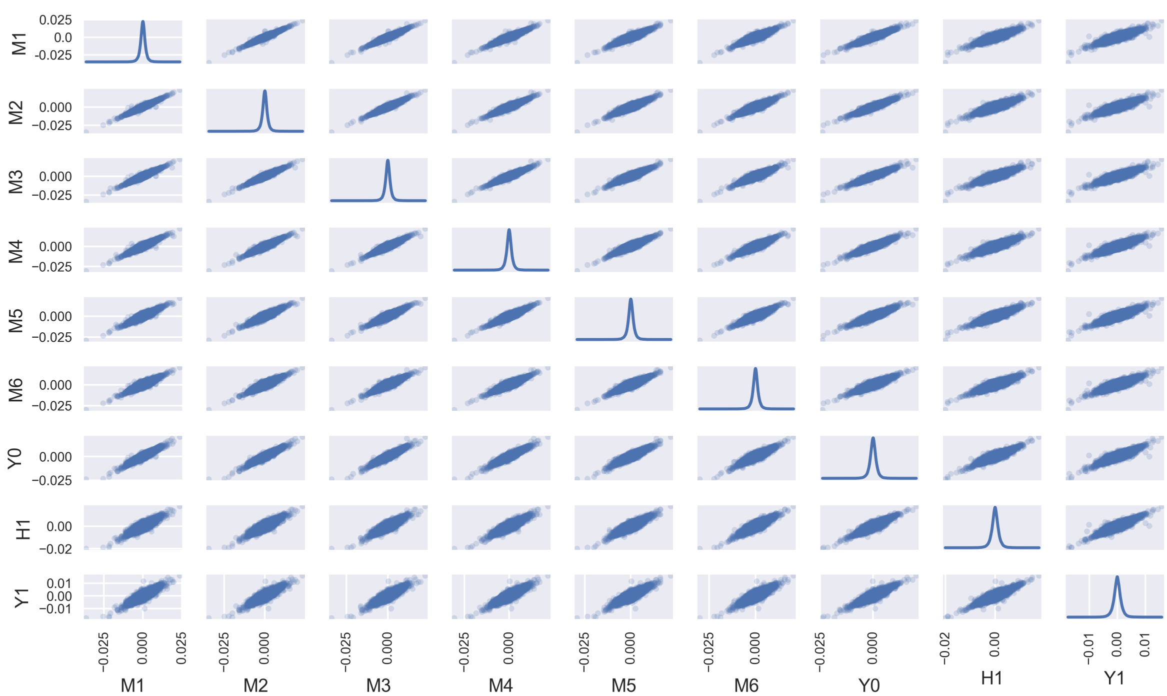


Figure 3.11: Scatter plot matrix for 5-min log return series. Correlation matrix is found in appendix D.

			B is Granger causing A			A is Granger causing B			
PairsID	*Contract A*	*Contract B*	*F-statistic*	*p-val*	*Lags*	*F-statistic*	*p-val*	*Lags*	*Nobs*
1	M1	M2	53.02	0.000	2	139.2	0.000	6	101392
2	M1	M3	29.59	0.000	1	329.3	0.000	6	100621
3	M1	M4	51.90	0.000	1	454.2	0.000	6	97840
4	M1	M5	8.556	0.003	1	743.9	0.000	5	92743
5	M1	M6	4.645	0.031	1	689.0	0.000	6	86463
6	M2	M3	44.18	0.000	3	273.8	0.000	6	100598
7	M2	M4	67.01	0.000	2	420.4	0.000	6	97829
8	M2	M5	22.03	0.000	2	592.1	0.000	6	92736
9	M2	M6	18.18	0.000	1	672.1	0.000	6	86435
10	M3	M4	225.3	0.000	2	243.9	0.000	7	97311
11	M3	M5	96.99	0.000	2	461.9	0.000	6	92387
12	M3	M6	37.34	0.000	2	577.4	0.000	6	86208
13	M4	M5	132.1	0.000	3	412.1	0.000	5	90763
14	M4	M6	91.20	0.000	2	464.9	0.000	6	84829
15	M5	M6	114.4	0.000	5	288.2	0.000	6	82037
16	Y0	Y1	72.56	0.000	1	560.5	0.000	4	82695
17	Y0	H1	58.25	0.000	1	737.6	0.000	4	75751
18	H1	Y1	287.8	0.000	4	122.3	0.000	5	71856

Table 3.4: Results of a bivariate Granger causality test for all pairs considered on 5-minute data. The hypothesis tested is: H_0: Lagged values of x do not explain the variation in y, i.e. coefficients of x-lags are all equal to zero. H_A: Lagged values of x have a statistically significant effect on y, i.e. at least one of x-lag coefficients are not zero. The test is computed as a Wald test comparing the unrestricted model (y is explained by lags of both y and x) and the restricted model (y is explained by lags of y only). Lag length is chosen using AIC.

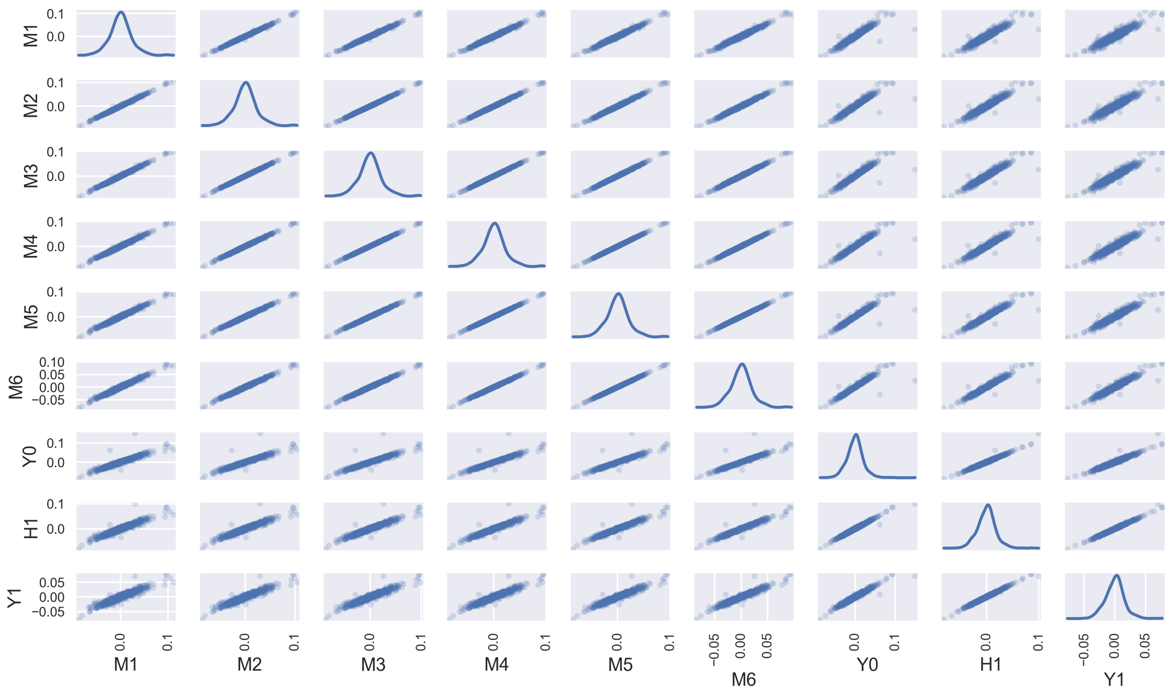


Figure 3.12: Scatter plot matrix for daily log return series. Correlation matrix is found in appendix D.

The first step in the Engle-Granger test, i.e. the test for unit roots in the individual log price series, is highly significant for all contracts. This means that log price series are non-stationary, as expected. Proceeding to test for unit roots in the first difference of log prices (log returns), these are proven to be stationary. As a result, all log price series are proven to be integrated of order 1, I(1).

Step two in the Engle-Granger test is to regress one of the series on the other and run another unit root test on the residuals of the regression. Test results are presented in table 3.5, while the full procedure for the EG test routine is explained in appendix E. We conclude that all spreads are cointegrated with high levels of statistical significance, as all p-values are below 0.02 in the Augmented Dickey-Fuller (ADF) test on the residuals of the cointegrating regression. We also note that the estimated cointegration coefficient (κ) is approximately one (1) for all series, indicating that an "energy-neutral"[9] position should be held.

These results suggest that log calendar spreads are in fact mean-reverting in the long run. Figure F.1 in the appendix shows the cointegration regression residuals (i.e. the log spreads, given the resulting κ of the cointegration regression) of all pairs throughout the period (2000-2018). These plots seem to confirm our conclusion of long-term mean reversion, but they also show that the spreads have large deviations from their mean during the financial crisis of 2009 and oil price crash of 2014-2015. This is due to the rapid changes in the term structure of oil futures at those points in time, which is reflected in the log spread (see appendix G for theoretical details).

From the analysis above with support in table 3.5, we see two problems emerging when trying to trade the long-term spread:

[9]An energy-neutral position is a long-short position with net exposure of zero units of the underlying commodity, e.g. long one lot and short one lot of crude oil

			Stationarity of log series				Stationarity of differenced log series				Cointegrating regression*				
PairsID	A	B	ADF, A	ADF, B	p-val, A	p-val, B	ADF, A	ADF, B	p-val, A	p-val, B	μ	κ	ADF	p-value	N. obs.
1	M1	M2	0.44	0.49	0.8105	0.8226	-15.66	-15.73	0.0000	0.0000	0.0140	0.9962	-4.79	0.0004	4684
2	M1	M3	0.44	0.54	0.8105	0.8338	-15.66	-15.86	0.0000	0.0000	0.0393	0.9898	-4.15	0.0043	4684
3	M1	M4	0.44	0.58	0.8105	0.8424	-15.66	-15.88	0.0000	0.0000	0.0663	0.9832	-4.00	0.0071	4684
4	M1	M5	0.44	0.68	0.8105	0.8630	-15.66	-21.50	0.0000	0.0000	0.0959	0.9760	-4.67	0.0006	4684
5	M1	M6	0.44	0.72	0.8105	0.8697	-15.66	-21.44	0.0000	0.0000	0.1250	0.9691	-4.31	0.0024	4684
6	M2	M3	0.49	0.54	0.8226	0.8338	-15.73	-15.86	0.0000	0.0000	0.0237	0.9940	-4.36	0.0021	4684
7	M2	M4	0.49	0.58	0.8226	0.8424	-15.73	-15.88	0.0000	0.0000	0.0493	0.9878	-4.48	0.0013	4684
8	M2	M5	0.49	0.68	0.8226	0.8630	-15.73	-21.50	0.0000	0.0000	0.0778	0.9809	-4.29	0.0027	4684
9	M2	M6	0.49	0.72	0.8226	0.8697	-15.73	-21.44	0.0000	0.0000	0.1057	0.9742	-4.28	0.0028	4684
10	M3	M4	0.54	0.58	0.8338	0.8424	-15.86	-15.88	0.0000	0.0000	0.0244	0.9940	-4.27	0.0028	4684
11	M3	M5	0.54	0.68	0.8338	0.8630	-15.86	-21.50	0.0000	0.0000	0.0515	0.9875	-4.19	0.0038	4684
12	M3	M6	0.54	0.72	0.8338	0.8697	-15.86	-21.44	0.0000	0.0000	0.0785	0.9810	-4.16	0.0042	4684
13	M4	M5	0.58	0.68	0.8424	0.8630	-15.88	-21.50	0.0000	0.0000	0.0262	0.9937	-3.95	0.0084	4684
14	M4	M6	0.58	0.72	0.8424	0.8697	-15.88	-21.44	0.0000	0.0000	0.0522	0.9875	-4.09	0.0054	4684
15	M5	M6	0.68	0.72	0.8630	0.8697	-21.50	-21.44	0.0000	0.0000	0.0252	0.9940	-3.72	0.0172	4684
16	Y0	Y1	0.70	1.03	0.8665	0.9204	-21.34	-49.70	0.0000	0.0000	0.2514	0.9426	-3.69	0.0190	4684
17	Y0	H1	0.70	0.87	0.8665	0.8964	-21.34	-21.19	0.0000	0.0000	0.1583	0.9634	-3.68	0.0192	4684
18	H1	Y1	0.87	1.03	0.8964	0.9204	-21.19	-49.70	0.0000	0.0000	0.0839	0.9816	-3.93	0.0091	4684

Table 3.5: Results from Engle-Granger test for cointegration in daily data from January 2000 to April 2018.

1. It can take a long time for the spread to mean-revert, resulting in low annualized profits and long trade lengths.

2. Contracts expire each month, and positions must be rolled over in order to be kept alive. This entails costs (commissions, bid-ask spreads) and can further reduce trading profits.

Both problems highlighted above are reasons why we wish to focus on short-term spread trading.

Looking for a short-term relationship

As we seek to profit from short-term mean reversion, we now test for cointegration on 5-min log spreads using the Engle-Granger procedure. To avoid rolling concerns, we test for cointegration in one month of absolute contract data at the time. We thus run the EG test for all 18 pairs over the 40 consecutive one-month intervals in our intraday dataset, which results in $18 \cdot 40 = 720$ tests in total. The results form a very large amount of data, and the main conclusion is that cointegration is not possible to prove in the short-term. In figure 3.13 we present a strip plot (one-dimensional scatter plot) of the p-values for all tests, across all pairs. The vertical axis represents different spreads, and the horizontal axis represents the distribution of p-values. Each dot represents a test for a given pair and month in our intraday sample. We quickly see that p-values are distributed all over the scale. Some tests show significant p-values, but most tests show high, insignificant p-values. As a consequence, we must conclude that spreads are not cointegrated on a 5-minute basis, and the cointegration approach (presented in section 2.2.2) to short-term spread trading is ruled out. In appendix F.2 we have plotted spreads and histograms of the ICE BRN M1 contract for all 40 consecutive one-month periods. In summary, we observe that a large number of the spreads have some sort of drift or jumps in them, which indicate that they are non-stationary. Further, most of the histograms do not exhibit a Gaussian form. But even though we can not apply the cointegration approach to a short-term strategy, we see from the spread plots in appendix F.2 that many series seem to have a high degree of oscillations around some kind of ”rolling mean”. This motivates a similar approach as Liu et al. (2016), in which we model the oscillations around a time-varying mean in order to make profitable trades in the short-term. In fact, we think that the presence of short-term divergence from some longer-term fundamental relationship provides an interesting backdrop for an intradaily trading strategy, and thus we proceed to implement the stochastic approach to spread trading in chapter 4.

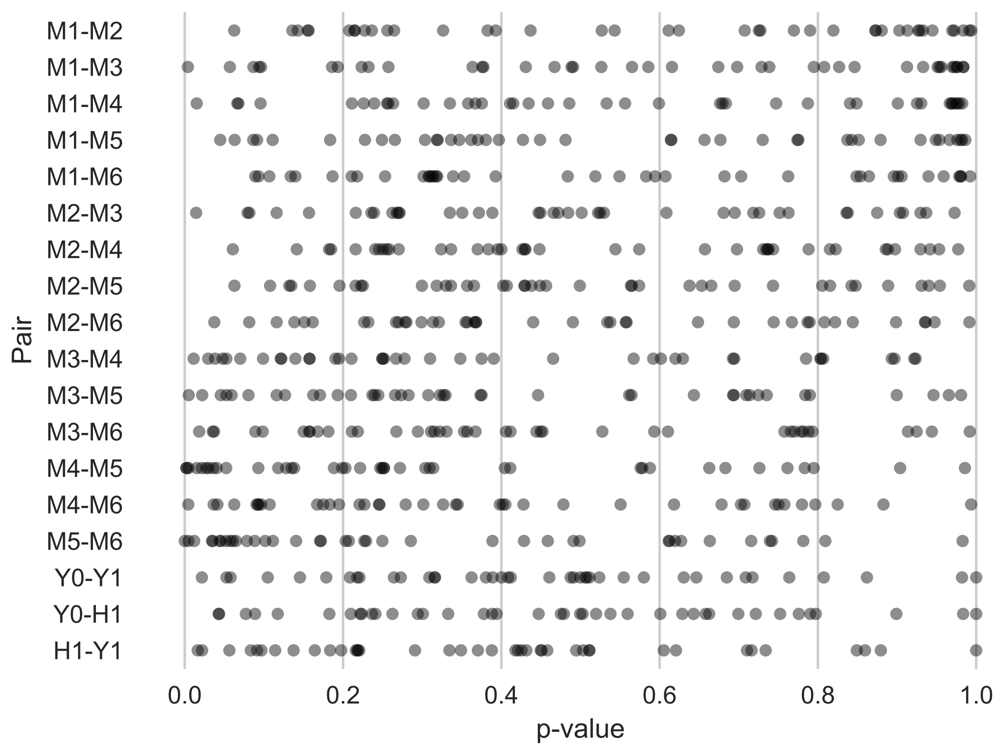


Figure 3.13: Stripplot (one-dimensional scatterplot) of p-values from ADF test of residuals in Engle-Granger procedure. Each pair has been tested in 40 consecutive one-month periods. One dot in diagram represents one test. AIC is used for numbers of lags in ADF test.

Chapter 4

Methodology

4.1 An overview of the backtesting model

To backtest the intraday spread trading strategy, we have built an event-driven backtesting environment in Python[1]. The two most important parts of the model is: 1) ranking and 2) trading (signal generation & execution). These are described in detail in sections 4.2.1 and 4.2.2. The building blocks and flow of the model are shown in figure 4.1, and the Python code for all functions used are found in appendix J.2. A short explanation of the model flow follows.

The entire data set from January 2015 to April 2018 is split into trading periods based on a pre-determined trading period length ($N = \{1, 5, 20\}$ days). At the start of each trading period, all possible pairs are ranked based on volatility and frequency of mean crossovers in the formation period (section 4.2.1 provide details). In our strategy, we trade K accounts. Depending on the number of accounts traded (K), we then backtest the top K pairs for the given trading period. The model runs through each timestamp in the trading period using intraday 5-minute data. Trading signals are generated based on prices from the underlying contracts and thresholds determined from the formation period (section 4.2.2 provide details). Orders are executed and accounted for, including transaction costs and bid-ask spreads. Combined ranking and trading procedure is repeated for all trading periods. Rolling is handled by not allowing a trading period to overlap the rolling date; i.e. contracts are re-ranked the day before maturity of currently traded contracts. Account balances are logged and accumulated return series are produced for each account at each time step, and this is used for calculating performance metrics at the end.

4.2 Spread trading strategy - the stochastic approach

We adopt a similar approach to spread trading as the one introduced in Liu et al. (2016). The approach outlined in the paper is based on the stochastic approach (details in section 2.2.3), but two stochastic processes are used instead of one. For each contract pair, we create a long-term log spread based on daily settlement prices and a short-term log spread based on 5-minute data. In the formation period, we model the short-term spread as an Ornstein-Uhlenbeck (OU) process with a time-varying mean based on

[1] For more details on such a structure, we refer the reader to Chan (2013).

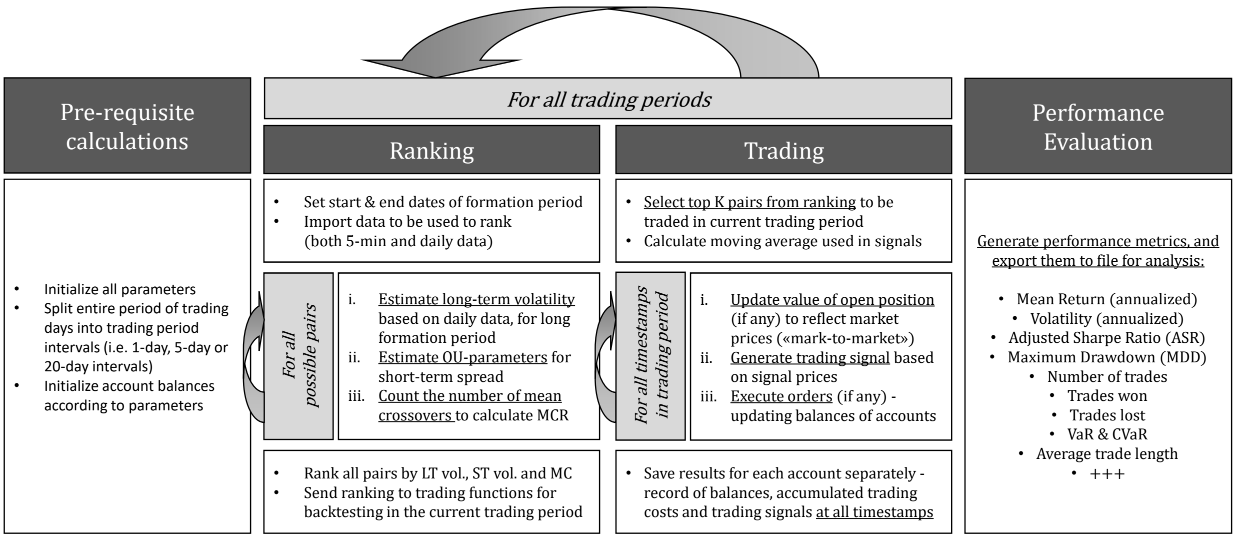


Figure 4.1: Boxmodel of the backtesting model implementation. Python code for the entire program is found in appendix J.2.

a trailing moving average. We then use the statistical properties of the short-term and long-term spreads in the formation period to: 1) rank pairs, and 2) generate thresholds for trading signals.

Why do we favour the stochastic approach? An important critique of the distance and classical cointegration approaches to pairs trading are that relationships between securities often are *dynamic* rather than static. Based on the initial discussion in Liu et al. (2016), two simple examples are provided:

1. *Firstly*, the log spread of two securities might experience a (quick) change in mean due to fundamental factors. In this case, it would not be identified as a candidate for pairs trading in the distance approach no matter their later co-movements. Because the oil price is heavily dependent on shocks in supply (such as geopolitical events), sudden changes in spreads can happen from time to time, reflecting changes in the forward curve.

2. *Secondly*, if two securities experience periods of strong co-movements but in some periods this does not hold, it would not be selected in a cointegration approach - even though profits could be made in periods. We would also add that the classical cointegration approach described in 2.2.2 is based on the spread returning to a constant mean. In section 3.2.2 we show that short-term cointegration cannot be proven for the contracts in our data set, and given that the calendar log spread is dependent on the roll yield (see appendix G for details), the mean of the log spread is not necessarily constant[2].

4.2.1 Ranking of pairs in formation periods

The ranking is performed at the start of each trading period and is held constant until the next trading period (i.e. 1 day, 5 days or 20 days in our study). Ideally, we want pairs with low long-term variance

[2]Note that we still argue for mean-reversion in the log-spread, but with a time-variant mean instead of a constant one. As pointed out in Liu et al. (2016), this corresponds to modelling the short-term oscillations around a dynamic long-term mean.

and high short-term variance. In addition, we want pairs that cross the mean as many times as possible in the formation period. In this case, we could make lots of trades in the short-term in the belief that the long-term spread will be somewhat stable. When ranking pairs for trading, we thus use the following three, equally weighted criteria:

1. **Long-term variance (LTV):** We want a pair which has lowest possible variability in the long-term log spread. This is measured by calculating the variance in the daily log spread of settlement values. Pairs with low long-term variability in log spread will presumably be more stable.

2. **Short-term variance (STV):** We want a pair with the highest possible short-term variability of the log spread, in order to exploit extreme movements in the short-term. This is measured by the variance in an Ornstein-Uhlenbeck (OU) process estimated over the short-term formation window, with a moving average mean. We do this because we want the largest possible variance *relative* to the mean.

3. **Mean Crossover Rate (MCR):** A higher number of mean crossovers indicates that more trades can be made (Vidyamurthy, 2004, p.112). A high number of trades with small average profits is better than some trades with high gains. The former will presumably have a lower Sharpe Ratio (due to lower variance) and also possibly a lower Calmar ratio (lower chance for large drawdowns). MCR is measured on intraday data as the number of times the log spread of a pair crosses its moving average.

Two formation period lengths are used: $L = 100$ days for the long-term spread and $S = 3N$ days for the short-term spread, where $N = \{1, 5, 20\}$ is the trading period length in days (illustrated in figure 4.4). In the long-term formation period, *relative* contracts (e.g. "ICE BRN M1" vs. "ICE BRN M2") are used. In the short-term formation period, we use the *absolute* contracts (e.g. "ICE BRN FEB-2017" vs. "ICE BRN MAR-2017"). The reason for this is that short-term intraday dynamics can be quite different across rolling, e.g. there can be considerable jumps in the spread.

An example ranking made at 1st of July 2015 using a long-term formation period of $L = 100$ days and a short-term formation period of $S = 3$ days is shown in table 4.1.

Long-term spread

For the long-term spread, we rank pairs from lowest to highest estimated volatility (σ_{LT}) in the long-term formation period of $L = 100$ days[3]. The calculation is shown in equation 4.1, of which μ_{LT} is the mean of the log spread in the formation period. The log spread is defined by $Y_t = log(P_{A,t}) - log(P_{B,t})$, using daily settlement prices for the long-term spread.

$$\sigma_{LT} = \frac{1}{L-1} \sum_{t=1}^{L} (Y_t - \mu_{LT})^2 \tag{4.1}$$

[3] Note that the volatility estimates of the spread do not need to be annualized, because we only use it for ranking the spreads in a formation period of given length.

	Relative contracts		Absolute contracts		Volatility est.			Ranking values (0-17)			
PairsID	*A*	*B*	*A*	*B*	*Long-term*	*Short-term*	*MCR*	*Vol. LT*	*Vol. ST*	*MCR*	*Total*
13	M4	M5	DEC-2015	JAN-2016	0.003166	0.102564	104	1	9	0	10
14	M4	M6	DEC-2015	FEB-2016	0.005929	0.114936	77	5	6	3	14
11	M3	M5	NOV-2015	JAN-2016	0.006643	0.116256	62	6	5	4	15
15	M5	M6	JAN-2016	FEB-2016	0.002776	0.082804	62	0	12	5	17
6	M2	M3	OCT-2015	NOV-2015	0.003461	0.079999	80	2	15	1	18
10	M3	M4	NOV-2015	DEC-2015	0.003486	0.082327	77	3	13	2	18
8	M2	M5	OCT-2015	JAN-2016	0.010058	0.121214	41	12	3	8	23
1	M1	M2	SEP-2015	OCT-2015	0.003554	0.080272	53	4	14	6	24
2	M1	M3	SEP-2015	NOV-2015	0.006875	0.107964	34	7	7	11	25
12	M3	M6	NOV-2015	FEB-2016	0.009397	0.120617	39	11	4	10	25
17	Y0	IH1	DEC-2015	JUN-2016	0.008639	0.106385	40	10	8	9	27
4	M1	M5	SEP-2015	JAN-2016	0.013347	0.14166	26	15	1	13	29
9	M2	M6	OCT-2015	FEB-2016	0.012808	0.129284	23	14	2	14	30
18	H1	Y1	JUN-2016	DEC-2016	0.007172	0.098171	27	9	10	12	31
7	M2	M4	OCT-2015	DEC-2015	0.006917	0.076078	45	8	17	7	32
5	M1	M6	SEP-2015	FEB-2016	0.016083	0.152448	16	17	0	17	34
16	Y0	Y1	DEC-2015	DEC-2016	0.015596	0.09441	18	16	11	16	43
3	M1	M4	SEP-2015	DEC-2015	0.01025	0.077553	21	13	16	15	44

Table 4.1: Example of ranking of pairs in formation period, at 1st of July 2015. Long-term formation period is $L = 100$ days, and short-term formation period is $S = 3$ days. All contracts are ranked from 0-17 in each of the three ranking metrics (LT vol, ST vol and MCR) which are summed up to form the total ranking.

This is in contrast to Liu et al. (2016), which model the long-term spread as an Ornstein-Uhlenbeck (OU) process and back out the volatility of the process. However, during our tests, we find that the correlation between our volatility estimates σ_{LT} and the OU process volatility estimates are very high. This indicates that the OU estimation of the long-term spread might not add much value to the ranking, at least in the case of calendar spreads in Brent Crude futures. We simply find it to be a complicating factor and thus use the simple volatility of the long-term spread for ranking.

Short-term spread

Because log spreads are non-stationary in the short-term (as shown in 3.2.2), reversion to a constant mean as tested by the Engle-Granger procedure cannot be assumed. But studying the data for Brent Crude futures, we find that the log spread does seem to oscillate around a time-varying mean in the short-term (similar to the dynamics shown in figure 4.2). To be able to trade on this very short-term variability, we model the short-term spread as an Ornstein-Uhlenbeck (OU) process with a time-varying mean, as shown in equation (4.2).

$$dY = \theta(\mu(t) - Y(t))dt + \sigma_{ST} dW \tag{4.2}$$

θ is the mean reversion rate, taken to be strictly positive ($\theta > 0$). σ_{ST} is the estimated volatility of the process. $\mu(t)$ is a time-varying mean based on the S-day equally weighted moving average of the log spread in the short-term formation period. Note that $\mu(t)$ is calculated before the OU parameters are estimated, and thus it is taken as inputs into the short-term spread model. If the movements in the short-term log spread are large enough, and the time-varying mean stays relatively constant in our trading period (N), a statistical arbitrage approach should be able to generate profits. The short-term process is estimated using 5-min data in a formation period of $S = 3N$, and exclude all time steps with incomplete data (i.e. at least one of the prices is blank). The OU process is estimated using linear regression on the discretized form of equation (4.2), in accordance with the methodology found in Clewlow and Strickland (2000). A short description of the estimation procedure is found in appendix H.

4.2.2 Trading signals

During backtesting, we monitor the following variables for each time step (i.e. 5 minutes): the log spread (Y_t), the time-varying mean of the log spread (μ_t), thresholds for the log spread (τ_t), the current state of our position (open/closed) and time remaining until trading period end. Based on these variables we continuously generate trading signals. A graphical illustration is provided in figure 4.2, and we urge the reader to actively review it along with details of the entry and exit signals.

Entry signals

The entry signal is triggered when the observed log spread (based on "Close" values from last time step) deviates outside a threshold. This threshold is measured relative to the time-varying mean of the spread, μ_t. Depending on which direction the log spread is moving, signals are given as:

$$\text{SHORT A, LONG B: if } Y_t > \mu_t + \tau_t \tag{4.3}$$

$$\text{LONG A, SHORT B: if } Y_t < \mu_t - \tau_t \tag{4.4}$$

The threshold τ_t is set to be the maximum value of: 1) the estimated trading costs plus a minimum required return (TC_{min}), and 2) the α percentile of absolute differences between the time-varying mean μ_t and observed log spread Y_t in the short-term formation period (Th_α). In other words, the idea is to trade when the log spread breaks the α percentile observed in the formation period, but not unless it will cover our estimated round-trip trading costs.

$$\tau_t = max\left\{TC_{min}, Th_\alpha\right\} \tag{4.5}$$

$$TC_{min} = 4 \cdot \left(\frac{(P^{Ask} - P^{Bid}) + C}{0.5(P_{A,t-1} + P_{B,t-1})}\right) + r \tag{4.6}$$

$$Th_\alpha = \text{Percentile}_\alpha\left(|\mu_t - Y_t|\right), t \in FP_{ST} \tag{4.7}$$

In equation (4.6), the bid-ask spread ($P^{Ask} - P^{Bid}$) is assumed to be equal for both legs and C is the commission average per contract, both in dollar terms. r is a required return for the trader to take a position, in percentage terms. Th_α is the α percentile of the absolute difference between the time-varying mean (μ_t) and log spread in the short-term formation period (FP_{ST}). Note that the transaction cost estimate is based on previously observed prices (in the conservative case) and thus represents a potential error. Also to be noted is our inclusion of a required return for the trader. We do this because otherwise,

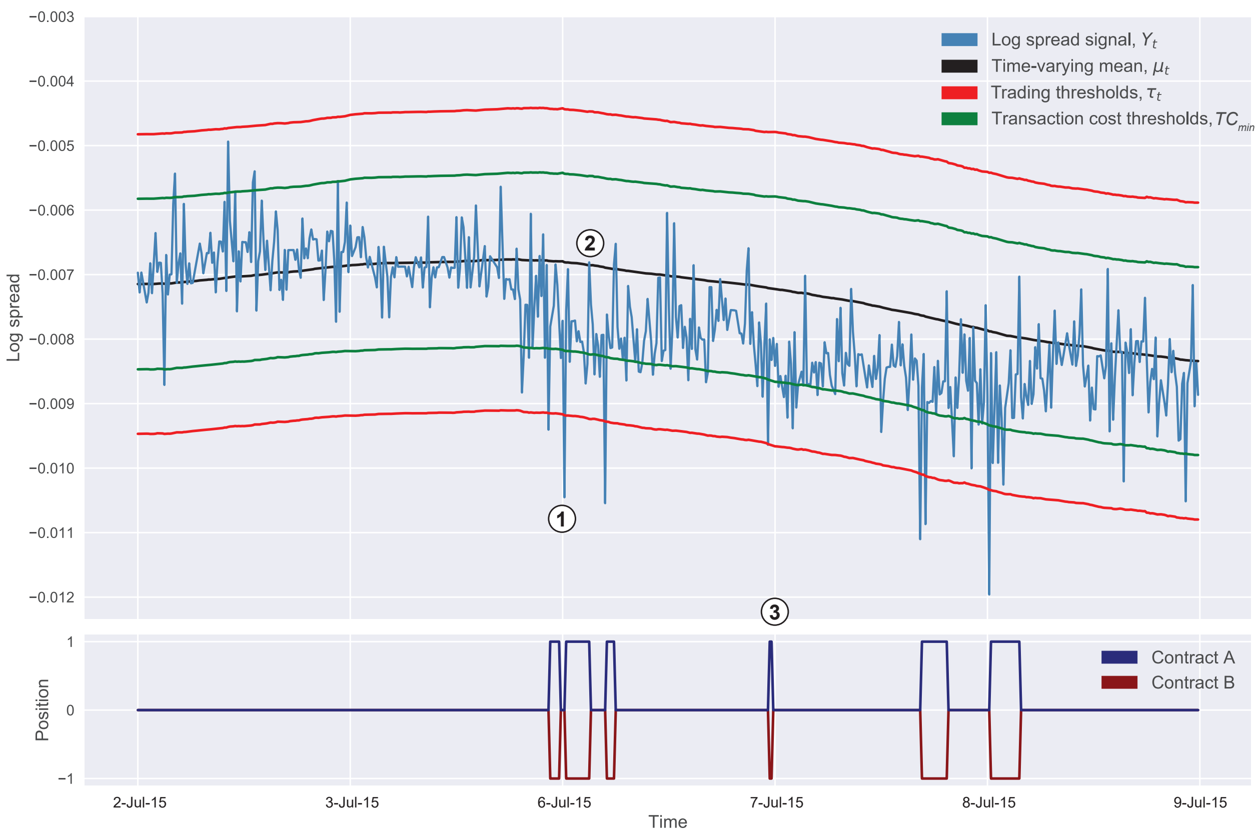


Figure 4.2: Example of how trading signals are generated. The strategy is based on a trading period of one day and a time-varying mean of 3 days length. (1): The log spread is outside the percentile threshold (τ_t), as defined in 4.5. In accordance with the entry criteria in equation 4.4, we LONG A and SHORT B. (2): The log spread has crossed back across the time-varying mean (μ_t) and our open position is closed, in accordance with the convergence criteria in equation 4.9. (3): First, an entry signal is triggered. But shortly after, we reach the end of our intraday trading period and the position is closed out.

trades that just cover the estimated trading costs will be entered into, leaving no room for profits. This illustrates an important principle of spread trading, in that profits are ”capped” once a position is entered into (assuming mean-reversion). Because of this, it is important for the trader to choose an appropriate required return for each trade.

Exit signals

We close the position if 1) the trade ”converge”, or if 2) we are at the end of the trading period specified. In our definition, the trade has converged when the log spread Y_t has crossed ”back” across the S-day moving average μ_t. I.e. for a later time t than the signal was entered:

$$\text{When SHORT A, LONG B: converged if } Y_t \leq \mu_t \tag{4.8}$$

$$\text{When LONG A, SHORT B: converged if } Y_t \geq \mu_t \tag{4.9}$$

The trading period end functions as a ”time stop-loss”, ensuring that non-converging positions are not held onto for too long (maximum $N = \{1, 5, 20\}$ days).

4.3 Backtesting and evaluating strategy performance

4.3.1 Backtesting design - formation periods and data snooping

Following Alexander (2008), the general idea of backtesting is to use an initial training period (formation period) to fit an econometric model and then apply this model to a new dataset (trading period) in order to evaluate performance. We split the time-series data of contract prices into formation periods and trading periods using the approach of ”walk forward validation”. This approach is based on moving through the time-series data from start to end, using a rolling window behind for formation and a window in front as the trading period. A graphical illustration is shown in figure 4.3. A motivation for splitting the data

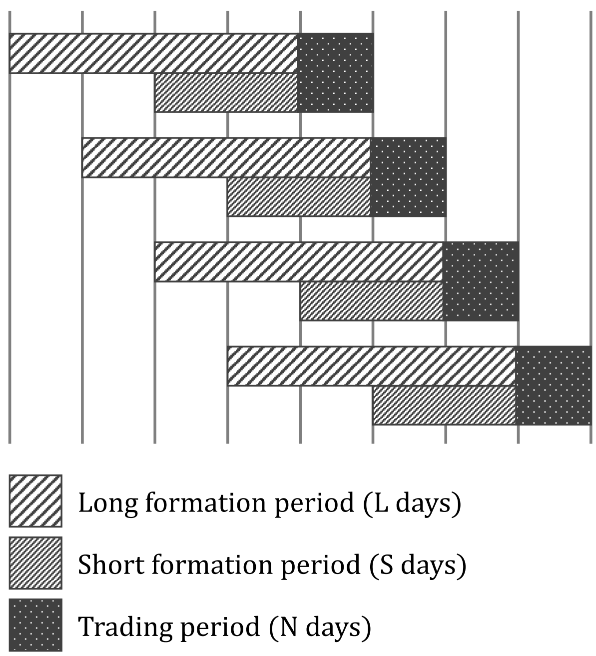


Figure 4.3: Graphical illustration of walk forward validation.

sample is the problem of data-snooping bias, in which we are ”using data that we are not supposed to know at the time when we estimate the model” Alexander (2008, Vol II, p. 363). This problem is of particular concern in the academic literature on trading strategies. It can also arise when testing a large number of parameter variations in-sample, of which one might find profitable trading rules by pure chance (similar to over-fitting a model). In the literature, it is common to apply White’s Reality Check (a statistical bootstrapping methodology) in order to test the robustness of the mean of returns. But this should only be necessary when using the same sample for both optimization and testing. By splitting the data into a formation period (in-sample) and a trading period (out-of-sample), we mitigate the risk of data-snooping and therefore do not apply White’s Reality Check on our results.

In our approach, the ranking of pairs and estimation of threshold parameters is performed in the formation period. Then the top K ranked pairs from the formation period are traded in the following trading period. This ensures that we are only using historical data when making decisions. The S-day moving average which is functioning as the mean of log spreads (μ_t) is re-calculated at every time step in the trading period. Performance of strategies is only measured during the trading period (out-of-sample).

We have not come across any formal rules in the literature on how long formation periods should be. We approach the choice of formation period length in the following manner (illustrated in figure 4.4):

- **The long-term formation period** is set to $L = 100$ days. Such a length would seem to capture long-term trends in log spread prices, based on looking at historical daily settlement prices before from 2000-2014, i.e. before our backtesting period.

- **The short-term formation period** is set to $S = 3N$, three (3) times the trading period length. This is based on rules-of-thumb in machine learning literature of using $\sim 80\%$ of the data for training and the rest for validation (e.g. the use of 5-fold Cross Validation). Thus, we implicitly use $3/4 = 75\%$ of the 5-minute data considered in a particular ranking for formation.

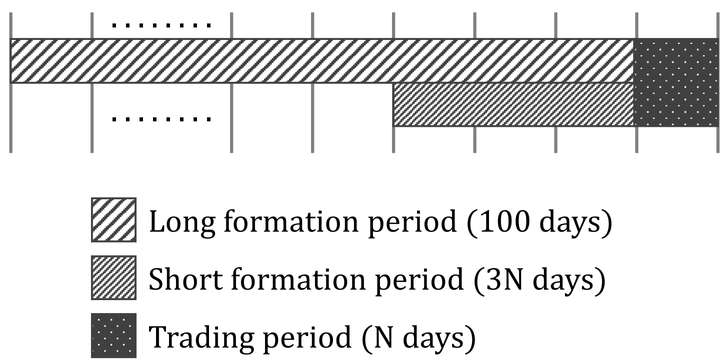


Figure 4.4: Example of the particular lengths of formation periods and trading period used. Note that the short formation period and the trading period is based on 5-min data, while the long formation period of 100 days is based on daily data.

4.3.2 Parameters of backtesting model

All parameters used in our backtesting model are listed in table 4.2, along with base case values. The six first parameters have several base case values, reflecting the fact that our base case consider three choices of trading period length, three cases of account numbers (top 5, top 10 and all pairs) for both an optimistic case and a conservative case.

4.3.3 The calculation of returns in a spread trading strategy

In backtesting strategies, we make use of a profit and loss (P&L) statement for each pair traded. Thus, each pair from the ranking function is given a dedicated account in a single trading period. We then specify an initial amount held in balance for each account and compute the changes in this account by

Parameter	Unit	Base case	Description
Trading period length	days	1/5/20	Length of trading period, before re-ranking
Trade at same time as signal	boolean	TRUE/FALSE	Optimistic vs. Conservative case
Which type of observations to use for execution	str	Close/VWAP	Optimistic vs. Conservative case
Length of short ranking period	days	3/15/60	Set equal to ma_multiple * trading_period_delta
Number of accounts	x	5/10/18	The number of top K pairs in ranking to be traded
Cash balance at start (total)	USDm	5/10/18	Set equal to 1 USDm per account
Threshold percentile	%	95 %	Percentile to use for threshold, based on daily changes in log spreads in ranking period
Bid/ask spread assumption	USD/contract	0,02	
Short margin	%	100 %	Margins needed for short position
Long margin	%	100 %	Margins needed for long position
Prices used for signals	str	Close	Close prices are always used for signal generation
Moving average multiple	x	3	
Length of long ranking period	days	100	Constant throughout the study
Rebalancing of account balances	boolean	FALSE	At the end of each trading period
Time resolution of trading	str	5min	Resolution of time series used for trading
Commission cost	USD/lot	1,34	Based on information from ICE cost sheets
Lot multiplier	x	1 000	Based on product specification from ICE
Required return in addition to trading cost	%	0,1%	Needed in order to not execute trades covering transaction costs only
Trading days per year	days	250	
Global start date (start of backtesting)	date	2015-06-01	
Global end date (end of backtesting)	date	2018-04-25	

Table 4.2: Overview of parameters in backtesting model. The six first parameters are varied in the base case, while others are held constant. Parameters 7-10 are varied in the sensitivity analysis part of results section. The parameter variable names in the Python program is found in appendix J.1.

"marking it to market" each time step. This approach mimics the exact dynamics of a trading account at the major brokerage houses and exchanges and allows us to implement restrictions, such as margin requirements when deciding on how many lots to buy.

We log data of all trades throughout the backtest and construct accumulated return series for all traded accounts. By aggregating the total returns across accounts, we also construct accumulated return series for the entire portfolio of accounts traded. Performance metrics (such as Sharpe Ratios, trades won/lost, mean returns, drawdowns, etc.) can be calculated using this data.

4.3.4 Performance metrics

An important tenet of modern portfolio theory and quantitative finance is to evaluate strategies on a *risk-adjusted* basis. A large number of performance metrics have been developed, and a short-list of both absolute and risk-adjusted metrics are found in table 4.3. First presented in Sharpe (1966), the Sharpe Ratio has become one of the industry standards for measuring risk-adjusted returns for the strategies of alternative investment funds. In this thesis, our main discussion of strategy performance will evolve around the Sharpe Ratio. For details on the other performance metrics used, we refer the reader to appendix I.

Because of the possibility of autocorrelation in returns, which can overstate the original Sharpe Ratio, we follow the methodology of Alexander (2008, Vol. I, p.259) and use the Adjusted Sharpe Ratio (ASR). The ASR can be calculated using daily mean returns ($\overline{R}$), standard deviation of daily returns (s), number of trading days (h) and an adjustment factor (k):

$$\mathrm{ASR} = \frac{h \cdot \overline{R}}{k \cdot s} \tag{4.10}$$

The daily mean returns are scaled with the number of trading days h (assumed to be 250) as usual, but the standard deviation needs an adjusted scaling factor in place of $\sqrt{h}$:

$$k = \sqrt{h + 2\frac{\rho}{(1-\rho)^2}\left[(h-1)(1-\rho) - \rho(1-\rho^{h-1})\right]} \tag{4.11}$$

where h is the number of trading days and ρ is the first order autocorrelation of the excess returns of the particular trading rule.

Absolute metrics	**Risk-return metrics**
Mean return	Sharpe ratio
Compound Annual Growth Rate (CAGR)	Sortino ratio
Volatility (SD)	Treynor ratio
Maximum Drawdown (MDD)	Sterling ratio
Length of Max. Drawdown	MAR/Calmar ratio
VaR/CVaR	K-ratio
Winning/losing trades	
Ulcer index	

Table 4.3: A short-list of absolute and risk-adjusted performance metrics. Metrics in grey are presented for the backtest results in this thesis.

4.3.5 Practical notes on the implementation of strategies

Signal vs. execution prices

In our implementation, we distinguish between signal and execution prices. By doing this we can test two cases: 1) An *optimistic* case, in which we are able to trade at the same prices we observe (observe Close and trade at same Close). 2) A *conservative* case, in which we observe the Close price in the *previous* interval and execute trades at the volume-weighted average price (VWAP) in the current interval[4]. In the absence of bid-ask order book data, we argue that the latter is a more sensible approximation in the backtest. In the results chapter, we show that the two approaches yield significantly different returns. We may also get an idea of the difference between the two approaches by looking at Figure 4.2, discussed earlier in this chapter. We note that upon signal generation, the spread tends to revert relatively quickly to within the transaction cost thresholds, making an eventual trade unprofitable when accounting for transaction costs.

[4] Daily settlement prices at ICE is calculated as the VWAP from 7:28:00 PM to 7:30:00 PM, BST. Thus, using VWAP at 5-minute intraday intervals is analogous to a settlement price for the 5-minute interval.

Handling of missing data

As described in the data chapter (3), we subset the data in such a way that there is a low percentage of missing values (i.e. timestamps without trades) in the 5-minute intervals we aggregate tick data into. In the backtest, we approach the issue of missing values in a practical manner. In the event that a blank occurs, the following happens in the backtest:

1. Trading signals are not updated and set equal to the previous trading signal.

2. No trades can be executed at intervals with missing values.

3. Market value of balances is not updated.

Other practical notes

- We let the proportion of lots in each leg of the trade be equal (energy-neutral). This is further motivated in the cointegration tests in section 3.2.2.

Chapter 5

Results

5.1 Performance of trading strategy

As described in the methodology chapter, we have divided our backtest into two cases: an optimistic case and a conservative case. A short recap of the cases is provided below.

1. **Optimistic:** We generate trading signals and execute trades at the same observed Close prices. Here, we assume *simultaneity* in signal and execution.

2. **Conservative:** We generate trading signals based on Close prices and execute trades at the VWAP for the following time interval. Here, we assume a *lag* between signal and execution.

Our main results are outlined in table 5.1. With support in the tabulated results, we will now state our main findings.

1. **Strategy performance is very sensitive to details of execution.** This is our most important finding. By comparing the performance of the optimistic and the conservative case, and by studying the performance sensitivity to changes in the bid-ask spread (further discussed in section 5.2), we conclude that even the smallest changes in execution price and timing may derail a strategy. This is a huge potential pitfall in any intradaily backtest. We try to illustrate the sensitivity to execution price and timing by varying the degree of simultaneity and the transaction costs. In the absence of historical order book data, this is the best we can do.

2. **Optimistic case performance is (very) good.** Spread trading in Brent Crude futures will yield good risk-adjusted returns if we assume simultaneity in signal and execution. Under our base case assumption of \$0.02 bid-ask spread per contract leg, implying a round-trip transaction cost of \$0.08 on a per-contract basis for every trade[1], we still achieve ASRs between 3.9 and 4.3 and unlevered annualized returns between 3.33% and 4.08% for 1-day trading periods. We see that these types of strategies are "ideal" for leverage (if they are sufficiently robust). Wit a modest leverage ratio of 4:1 (many brokerage accounts operate with sub-10% margin requirements for futures trading), the

[1] \$0.02 x 4 legs = \$0.08 per round-trip trade plus a negligible fee of \$1.34 per lot, or \$0.00134 per contract leg.

Strategy parameters			**Top 5 pairs**					**Top 10 pairs**					**All pairs**				
Optimistic case	*TP*	T_i	*Return*	*SD*	*ASR*	*MDD*	*Trades*	*Return*	*SD*	*ASR*	*MDD*	*Trades*	*Return*	*SD*	*ASR*	*MDD*	*Trades*
1	1d	0.95	3.93 %	1.0 %	4.0	0.1 %	1235	2.23 %	0.8 %	2.8	0.7 %	2797	0.27 %	0.8 %	0.4	2.6 %	5447
2	5d	0.95	3.05 %	0.8 %	3.6	0.1 %	821	1.37 %	0.7 %	1.9	0.9 %	1366	0.37 %	0.8 %	0.5	1.9 %	2115
3	20d	0.95	0.74 %	0.7 %	1.1	0.1 %	243	0.22 %	0.7 %	0.3	1.0 %	390	0.26 %	0.8 %	0.3	0.3 %	580
4	1d	0.9	4.08 %	1.1 %	3.9	0.1 %	1315	2.27 %	0.9 %	2.6	0.8 %	3050	0.08 %	0.8 %	0.1	3.0 %	6067
5	5d	0.9	3.26 %	1.0 %	3.4	0.1 %	916	1.45 %	0.8 %	1.8	1.0 %	1547	0.33 %	0.9 %	0.4	2.2 %	2438
6	20d	0.9	0.86 %	0.8 %	1.1	0.7 %	300	0.15 %	0.8 %	0.2	1.3 %	480	0.08 %	0.9 %	0.1	1.4 %	721
7	1d	0.99	3.33 %	0.8 %	4.3	0.1 %	976	2.02 %	0.7 %	3.1	0.5 %	2183	0.57 %	0.6 %	0.9	1.8 %	4238
8	5d	0.99	2.01 %	0.6 %	3.4	0.1 %	571	0.89 %	0.6 %	1.6	0.8 %	989	0.18 %	0.7 %	0.3	1.4 %	1553
9	20d	0.99	0.47 %	0.5 %	0.9	0.7 %	153	0.18 %	0.6 %	0.3	0.8 %	268	0.40 %	0.8 %	0.5	0.3 %	422
Conservative case	*TP*	T_i	*Return*	*SD*	*ASR*	*MDD*	*Trades*	*Return*	*SD*	*ASR*	*MDD*	*Trades*	*Return*	*SD*	*ASR*	*MDD*	*Trades*
10	1d	0.95	-6.36 %	1.0 %	-6.5	17 %	1220	-7.01 %	0.9 %	-8.1	18.5 %	2767	-7.49 %	0.8 %	-9.0	19.6 %	5393
11	5d	0.95	-4.25 %	0.7 %	-6.2	12 %	821	-3.59 %	0.6 %	-5.6	10.0 %	1364	-3.02 %	0.8 %	-3.9	8.5 %	2112
12	20d	0.95	-1.19 %	0.4 %	-2.9	3 %	243	-1.01 %	0.5 %	-1.9	3.2 %	390	-0.55 %	0.8 %	-0.7	2.6 %	580
13	1d	0.9	-6.78 %	1.1 %	-6.3	18 %	1300	-7.70 %	0.9 %	-8.1	20.1 %	3024	-8.43 %	0.9 %	-9.4	21.8 %	6022
14	5d	0.9	-4.79 %	0.8 %	-6.4	13 %	916	-4.12 %	0.7 %	-6.2	11.3 %	1546	-3.61 %	0.8 %	-4.4	10.0 %	2437
15	20d	0.9	-1.48 %	0.4 %	-3.4	4 %	300	-1.34 %	0.6 %	-2.3	4.1 %	480	-0.88 %	0.8 %	-1.0	3.5 %	721
16	1d	0.99	-5.02 %	0.7 %	-6.9	14 %	964	-5.43 %	0.7 %	-8.0	14.6 %	2161	-5.67 %	0.7 %	-8.1	15.3 %	4201
17	5d	0.99	-3.01 %	0.5 %	-5.9	8 %	569	-2.58 %	0.6 %	-4.7	7.3 %	987	-2.26 %	0.7 %	-3.4	6.4 %	1549
18	20d	0.99	-0.73 %	0.4 %	-2.0	2 %	153	-0.65 %	0.5 %	-1.4	2.0 %	268	-0.16 %	0.7 %	-0.2	1.6 %	422

Table 5.1: Performance metrics for base case scenario. Notation: T_i is threshold percentile for generating signals (see section XYZ for details). Return is mean annualized return. Trades is number of round-trip trades completed. Numbers to the left under Strategy Parameters is reference numbers for the specific parameter cases. Refer to appendix A for a complete list of acronyms.

top-performing 1-day strategy will achieve an annualized return of 16.8% at the cost of a 4.0% standard deviation and a maximum drawdown of 0.3%. This would be an extraordinarily good strategy.

3. **Conservative case performance is (very) poor.** Without simultaneity in signal and execution, spread trading profits diminish. None of our parameter choices yields positive returns, and performance is at its poorest for 1-day trading periods. We suspect that performance for 20-day trading periods is slightly less poor simply because we trade less frequently.

4. **Higher ranked pairs outperform lower-ranked pairs.** Ranking and selection of pairs for trading works, as we consistently achieve better results with higher ranked pairs. This can be seen in the results table by comparing performance for top 5-, top 10- and all pairs, respectively. We emphasize that this is not an after-the-fact comparison of the best to worst performing pairs, but rather a strategical concept in which we rank pairs based on formation period performance and select the top K performers for trading in the subsequent trading period. For daily trading periods, this ranking is thus performed every day. The concept of ranking is further discussed in sections 4.2.1 and 5.3.

5. **For winning strategies, shorter trading periods outperform longer trading periods.** In a similar fashion that negative returns are amplified by shorter trading periods for the conservative case, positive returns are amplified by shorter trading periods in the optimistic case. This should be fairly intuitive: If we take bets with positive expected value more often, we earn more.

5.2 The impact of bid-ask spread assumptions on backtest performance

As described in the Methodology chapter, we include the bid-ask spread when calculating transaction costs. We do this by assuming that we always cross the spread when executing a trade. In a real-life implementation of the described strategies, we would prefer to observe the actual order book and execute trades at the offered prices if they are above or below our thresholds, hence removing large parts of the uncertainty related to the simultaneity of signals and execution. As we do not have access to historical order book data we instead take the size of the bid-ask spread in as a parameter and assume that we always have to pay the bid-ask spread. The cost of crossing the bid-ask spread is the largest component of transaction costs; the fixed fee per lot is only \$1.34 (\$0.00134 per contract).

Trading strategy performance is highly sensitive to the size of the bid-ask spread. Transaction costs are driven by the size of the bid-ask spread, which in turn determines 1) what threshold we will base our trading signals on, and 2) the cost of taking a trade. In this sense, a higher bid-ask spread will result in 1) higher thresholds and 2) higher costs. With higher bid-ask spreads, we will trade less frequently for lower profits. The data in table 5.2 unambiguously confirm this. For the optimistic case, we see that increasing the bid-ask spread result in lower returns and fewer trades across all pairs selections and trading period (TP) lengths. Moreover, we note that profits are (heavily) impacted by small changes in the bid-ask spread. We use the optimistic case to illustrate: For the top 5 pairs with TPs of one day (top left panel), a spread of \$0.01 will yield annualized profits of 14.0% at an ASR of 6.9 - unlevered. Removing the bid-ask spread altogether, equivalent of assuming that we can trade at the exact prices we observe, the top 5 pairs yield an annualized return of 37.7% at an ASR of 12.5. This is incredibly high. However: just the slightest increase in bid-ask spreads will hurt profits significantly. At a \$0.03 bid-ask spread, returns are barely positive at 0.6% annualized; at a \$0.04 bid-ask spread, returns turn negative. This illustrates a key property of short-term pairs trading strategies: even the slightest alteration of parameters might break the strategy, as we rely on highly frequent trades with low expected profits per trade.

Moving along with the conservative case, another interesting feature is to be noted: all implementations apart from one yield negative returns. The delay between signal and execution hurts profits badly. The sole positive-return strategy, where we supposedly trade all pairs over 20-day trading periods at an assumed bid-ask spread of zero, yields 0.22% annualized at an ASR of 0.3. Short-term spread trading in Brent Crude futures under conservative assumptions is not very promising.

Another interesting conclusion can be drawn from the bid-ask sensitivity in the conservative case. If we exclude the bid-ask spread of zero, increasing the bid-ask spread will in some cases actually improve returns. This is especially apparent for 1-day TPs. This may seem counter-intuitive, but the explanation is straightforward: the conservative case yield negative returns for all but one strategy. As we already have pointed out, increasing the bid-ask spread will significantly reduce the number of trades we make due to higher threshold values. By increasing the bid-ask spread, we simply take fewer trades, and because of the negative expected value of trades, we end up losing less. This is the same, sobering fact that all losing gamblers will eventually have to face: the only way to improve long-term returns in a game with negative expected value is to play less.

Choosing an average bid-ask spread for all contracts is not necessarily correct. We hypothesize that

Strategy parameters			**Top 5 pairs**					**Top 10 pairs**					**All pairs**				
Optimistic case	*TP*	*BA spread*	*Return*	*SD*	*ASR*	*MDD*	*Trades*	*Return*	*SD*	*ASR*	*MDD*	*Trades*	*Return*	*SD*	*ASR*	*MDD*	*Trades*
19	1d	0.00	37.68 %	3.0 %	12.5	0.0 %	6111	28.45 %	2.3 %	12.3	0.0 %	9473	20.72 %	1.7 %	12.2	0.0 %	13044
20	1d	0.01	14.00 %	2.0 %	6.9	0.0 %	2910	10.27 %	1.5 %	7.0	0.0 %	5305	6.86 %	1.1 %	6.3	0.2 %	8539
1	1d	0.02	3.93 %	1.0 %	4.0	0.1 %	1235	2.23 %	0.8 %	2.8	0.7 %	2797	0.27 %	0.8 %	0.4	2.6 %	5447
21	1d	0.03	0.60 %	0.5 %	1.2	0.9 %	549	-0.68 %	0.6 %	-1.2	3.1 %	1562	-2.54 %	0.7 %	-3.8	7.4 %	3657
22	1d	0.04	-0.31 %	0.3 %	-0.9	1.3 %	254	-1.55 %	0.5 %	-3.3	4.6 %	904	-3.54 %	0.7 %	-5.4	9.8 %	2509
23	1d	0.05	-0.55 %	0.3 %	-2.1	1.8 %	142	-1.65 %	0.4 %	-3.9	4.7 %	561	-3.73 %	0.6 %	-5.8	10.3 %	1765
24	1d	0.10	-0.16 %	0.1 %	-1.5	0.5 %	10	-0.47 %	0.2 %	-2.4	1.4 %	52	-1.75 %	0.4 %	-4.2	5.0 %	314
25	5d	0.00	13.12 %	1.6 %	8.0	0.0 %	1869	8.32 %	1.1 %	7.5	0.1 %	2532	5.43 %	1.0 %	5.5	0.2 %	3305
26	5d	0.01	7.23 %	1.2 %	6.1	0.1 %	1348	4.30 %	0.9 %	5.0	0.1 %	1978	2.57 %	0.9 %	3.0	0.2 %	2749
2	5d	0.02	3.05 %	0.8 %	3.6	0.1 %	821	1.37 %	0.7 %	1.9	0.9 %	1366	0.37 %	0.8 %	0.5	1.9 %	2115
27	5d	0.03	0.72 %	0.6 %	1.1	0.9 %	482	-0.40 %	0.6 %	-0.6	2.5 %	937	-1.10 %	0.8 %	-1.4	3.8 %	1641
28	5d	0.04	-0.32 %	0.5 %	-0.6	1.8 %	309	-1.29 %	0.6 %	-2.2	3.8 %	699	-1.99 %	0.8 %	-2.5	5.7 %	1360
29	5d	0.05	-0.80 %	0.4 %	-2.0	2.5 %	193	-1.77 %	0.6 %	-3.1	5.1 %	529	-2.60 %	0.8 %	-3.3	7.3 %	1149
30	5d	0.10	-0.60 %	0.3 %	-2.4	1.8 %	29	-1.69 %	0.5 %	-3.6	4.8 %	164	-3.28 %	0.8 %	-4.2	9.1 %	549
31	20d	0.00	2.54 %	0.8 %	3.3	0.1 %	355	1.49 %	0.7 %	2.1	0.3 %	502	1.23 %	0.8 %	1.5	0.3 %	692
32	20d	0.01	1.56 %	0.7 %	2.2	0.1 %	305	0.81 %	0.7 %	1.2	0.3 %	452	0.71 %	0.8 %	0.9	0.3 %	642
3	20d	0.02	0.74 %	0.7 %	1.1	0.1 %	243	0.22 %	0.7 %	0.3	1.0 %	390	0.26 %	0.8 %	0.3	0.3 %	580
33	20d	0.03	0.14 %	0.7 %	0.2	0.8 %	203	-0.25 %	0.7 %	-0.4	1.6 %	347	-0.13 %	0.8 %	-0.2	1.7 %	536
34	20d	0.04	-0.29 %	0.7 %	-0.4	1.5 %	173	-0.62 %	0.7 %	-0.9	2.3 %	310	-0.47 %	0.8 %	-0.6	2.5 %	498
35	20d	0.05	-0.56 %	0.7 %	-0.8	1.9 %	141	-0.88 %	0.7 %	-1.3	3.0 %	273	-0.73 %	0.9 %	-0.9	3.2 %	457
36	20d	0.10	-0.82 %	0.4 %	-1.9	2.4 %	44	-1.40 %	0.6 %	-2.5	4.2 %	137	-1.55 %	0.9 %	-1.8	5.4 %	307
Conservative case	*TP*	*BA spread*	*Return*	*SD*	*ASR*	*MDD*	*Trades*	*Return*	*SD*	*ASR*	*MDD*	*Trades*	*Return*	*SD*	*ASR*	*MDD*	*Trades*
37	1d	0.00	-1.11 %	0.4 %	-2.8	3.4 %	6071	-0.79 %	0.4 %	-1.9	2.7 %	9405	-0.51 %	0.5 %	-1.0	1.8 %	12946
38	1d	0.01	-7.82 %	1.0 %	-8.2	20.4 %	2892	-6.94 %	0.8 %	-9.1	18.3 %	5256	-6.08 %	0.7 %	-8.7	16.3 %	8462
10	1d	0.02	-6.36 %	1.0 %	-6.5	16.9 %	1220	-7.01 %	0.9 %	-8.1	18.5 %	2767	-7.49 %	0.8 %	-9.0	19.6 %	5393
39	1d	0.03	-4.29 %	0.8 %	-5.4	11.8 %	547	-5.82 %	0.8 %	-6.9	15.6 %	1553	-7.39 %	0.9 %	-8.2	19.4 %	3629
40	1d	0.04	-2.55 %	0.5 %	-4.8	7.1 %	252	-4.40 %	0.7 %	-6.2	12.0 %	899	-6.58 %	0.9 %	-7.6	17.5 %	2481
41	1d	0.05	-1.82 %	0.5 %	-4.0	5.2 %	141	-3.37 %	0.6 %	-5.5	9.3 %	555	-5.72 %	0.8 %	-6.8	15.4 %	1746
42	1d	0.10	-0.27 %	0.2 %	-1.7	0.8 %	10	-0.62 %	0.2 %	-2.6	1.8 %	51	-2.02 %	0.5 %	-4.3	5.7 %	311
43	5d	0.00	-0.34 %	0.4 %	-0.8	1.6 %	1871	-0.34 %	0.5 %	-0.7	1.7 %	2533	-0.21 %	0.7 %	-0.3	1.9 %	3305
44	5d	0.01	-3.43 %	0.5 %	-6.4	9.5 %	1350	-2.64 %	0.6 %	-4.7	7.5 %	1979	-2.01 %	0.7 %	-2.8	5.7 %	2749
11	5d	0.02	-4.25 %	0.7 %	-6.2	11.7 %	821	-3.59 %	0.6 %	-5.6	10.0 %	1364	-3.02 %	0.8 %	-3.9	8.5 %	2112
45	5d	0.03	-3.85 %	0.7 %	-5.5	10.7 %	481	-3.71 %	0.7 %	-5.5	10.3 %	935	-3.54 %	0.8 %	-4.4	9.8 %	1638
46	5d	0.04	-3.27 %	0.6 %	-5.1	9.1 %	310	-3.63 %	0.7 %	-5.3	10.1 %	699	-3.84 %	0.8 %	-4.6	10.6 %	1358
47	5d	0.05	-2.56 %	0.5 %	-4.7	7.2 %	194	-3.39 %	0.7 %	-5.0	9.4 %	529	-4.00 %	0.8 %	-4.7	11.0 %	1148
48	5d	0.10	-0.77 %	0.3 %	-2.9	2.3 %	29	-2.08 %	0.6 %	-3.8	5.9 %	164	-3.76 %	0.8 %	-4.5	10.4 %	548
49	20d	0.00	-0.05 %	0.4 %	-0.1	0.8 %	356	-0.09 %	0.5 %	-0.2	1.1 %	503	0.22 %	0.8 %	0.3	0.3 %	693
50	20d	0.01	-0.76 %	0.4 %	-2.0	2.3 %	306	-0.62 %	0.5 %	-1.2	2.1 %	453	-0.20 %	0.8 %	-0.2	1.7 %	643
12	20d	0.02	-1.19 %	0.4 %	-2.9	3.5 %	243	-1.01 %	0.5 %	-1.9	3.2 %	390	-0.55 %	0.8 %	-0.7	2.6 %	580
51	20d	0.03	-1.51 %	0.4 %	-3.4	4.4 %	203	-1.34 %	0.6 %	-2.4	4.0 %	347	-0.86 %	0.8 %	-1.1	3.4 %	536
52	20d	0.04	-1.70 %	0.5 %	-3.4	4.8 %	173	-1.56 %	0.6 %	-2.7	4.6 %	310	-1.11 %	0.8 %	-1.3	4.1 %	498
53	20d	0.05	-1.79 %	0.5 %	-3.4	5.1 %	141	-1.75 %	0.6 %	-2.9	5.1 %	273	-1.34 %	0.9 %	-1.6	4.7 %	457
54	20d	0.10	-1.11 %	0.4 %	-2.9	3.2 %	44	-1.73 %	0.5 %	-3.2	5.0 %	137	-1.85 %	0.9 %	-2.1	6.2 %	307

Table 5.2: Sensitivity analysis with respect to assumptions of bid-ask spread in trading. Notation: TP is Trading Period length. BA spread is bid-ask spread assumed for all contracts when trading. Return is mean annualized return. Trades are the number of round-trip trades completed. Numbers to the left under Strategy Parameters are reference numbers for the specific parameter cases. Refer to appendix A for a complete list of acronyms.

bid-ask spreads tend to increase with decreasing contract liquidity. Thus, we are likely overestimating transaction costs for near-term contracts while possibly underestimating transaction costs for longer-term contracts. This poses a few problems. From the section on the ranking of pairs, we know that mid-term spreads yield the best spread trading profits. If these profits are simply an artefact of overly optimistic bid-ask spread assumptions, we would have to conclude that spread trading cannot be profitably applied in the markets studied.

5.3 A closer look on the ranking of pairs

As described in section 4.2.1, we rank and select pairs for trading based on three criteria: long-term variance (LTV), short-term variance (STV) and mean crossover rate (MCR). As we are looking to profit from short-term, mean-reversion based trading strategies, we prefer pairs that have low LTV, high STV and a high MCR. Ranking works well for the optimistic case: as observed in figure 5.1, top-ranked pairs clearly outperform lower ranked pairs in terms of Adjusted Sharpe Ratios. In this section, we seek to explain which contracts are selected for trading and why. Our metrics of choice for this analysis will be the selection frequency, which we define as the percentage of time a specific pair is selected as a top N-th pair in the ranking function, and the average rank achieved. We show selection frequencies and average ranks for all traded pairs in tables 5.3 - 5.5.

By examining tables 5.3 - 5.5, it will become apparent to the reader that not all calendar spreads are equally desirable for short-term trading. Some pairs are frequently selected among the top pairs, while others are seemingly always at a disadvantage. By simply looking at the colour coding of the three tables, it becomes apparent that certain pairs are superior or inferior regardless of trading period (TP) length. Among the top rankings, we see that the M5-M6 pair is selected far more often than any other pair: it is the most preferable pair for short-term trading 66.5%, 68.9% and 75.4% of the time for TPs of 1 day, 5 days and 20 days, respectively. For 20-day TPs M5-M6 has an average rank of 1.4, and it is never ranked below 4. Further this, we discover that the second best pair is also uncontested for its spot: M4-M5 is the top pair 20% of the time across all TPs and ranked top 3 more than 5 out of 6 times. The average rank of M4-M5 is 2.5 for all TPs.

Following the two highly superior pairs, M3-M4 and M4-M6 are next in line: top 3 pairs every second to third time across TPs, and more often than not in in the top 5 (65.7% at worst; above 90% for M3-M4 with a 20-day TPs). M3-M4 is occasionally the top pair for 1- and 5-day TPs: 6.7% and 8.5% of the time, respectively. M2-M3 follows, with an average rank of 6.7 for 1-day TPs.

We now examine the top 10 pairs. We observe that spreads with M2 as the first leg are increasingly coming into favour, with the shorter time-delta pairs being selected for top 10 much more often than not (M2-M3 is a top 10 pair 9 out of 10 times for 1-day TPs, M2-M4 about 7 in 10 times). M2 spreads achieve average ranks between 6.0 and 11.6 across TPs, with the average rank increasing with time-delta. The observant reader may have noticed that spreads with M1 as the first leg are still disfavoured. M1-M2, the most liquid spread, has an average rank of 12.0 for 1-day TPs. For 20-day TPs the average rank improves to 9.2, still in the lower half. The remaining spreads that include M1 rank poorly. The average rank of the 4 pairs all falls between 12.9 and 14.0.

The bottom end of selection frequencies help us confirm the notion created early on - that spreads with high time-deltas consistently rank lower than spreads with low time-deltas. Y0-Y1 is the lowest ranked pair. With average ranks of 14.7, 15.6 and 15.8 for 1-, 5- and 20-day TPs, it confirms our theory that high time-delta spreads are inferior.

The top ranking pairs share two distinct characteristics. All of the pairs have short time-deltas (one or two months), and all of the pairs are comprised of mid-term contracts (M3, M4, M5 and M6). This is an interesting finding. From figure 3.1 in chapter 3, we know that pairs with lower time-deltas are typically lower in absolute values than pairs with higher time-deltas, simply due to the legs' proximity to each other on the forward curve. This implies a smaller spread in dollar value, a smaller spread relative to tick-sizes, and we can confirm from figure 4.1 that the MCR of close spreads is much larger on average than that of wider spreads. Going back to the data section once more, we also showed in figure 3.1 that log spreads and roll yields vary heavily with time-deltas. Log spreads for neighbouring contracts on the forward curve might be as low as 1%, while at the same time well above 10% for wider spreads such as Y0-Y1. For roll yields the opposite is true - the roll yield of close spreads are much higher in absolute terms than that of wider spreads. The most important finding, however, is the difference in volatility between spreads. The long-term volatility of close spreads is <u>much</u> lower than that of wider spreads. This impacts the ranking contribution from both LTV and MCR and is in turn what makes the wide spreads inferior to the close spreads.

We are still left with one important question: why does the ranking function prefer mid-term contracts over near-term contracts? We propose two possible explanations. Firstly, i.e. M5 and M6 are (a lot) less liquid than M1 and M2. We hypothesize that arbitrage opportunities and deviations from the LOP are more common in less liquid contracts. In this context, the reason why M1-M2 is inferior to M5-M6 might just be that the volumes traded in front- and second front-month contracts are (much) higher than the volumes traded in more distant contracts. We observe that the STV of M1-M2 usually ranks lower than the STV of M5-M6, and we can argue that the difference in traded volumes is what causes the differences in volatility. The other possible explanation is that the observed short-term volatility in M5-M6 and other less liquid spreads is, in fact, artificial volatility. We fear that with thinner volumes comes larger bid-ask spreads and that the observed short-term volatility in contracts may simply be buyers and sellers crossing a sizeable bid-ask spread. Another potential cause of artificial volatility can be the 5-minute frequency at which data is sampled. The closing prices of each leg, together making up the spread, may, in theory, be traded several minutes apart, and may not represent simultaneously available trading prices. If prices are not simultaneously available, and if short-term volatility is primarily caused by the bid-ask bounce, we are prone to significantly over-estimating STV and hence the profit potential of our spread trading strategy.

Pair	ID	Top 1	Top 2	Top 3	Top 4	Top 5	Top 6	Top 7	Top 8	Top 9	Top 10	Top 11	Top 12	Top 13	Top 14	Top 15	Top 16	Top 17	Top 18	Avg. rank
M1-M2	1	0.0 %	0.8 %	1.5 %	3.4 %	7.4 %	12.0 %	18.0 %	25.1 %	32.4 %	38.6 %	44.8 %	53.3 %	59.8 %	65.0 %	70.6 %	79.7 %	90.7 %	100.0 %	12.0
M1-M3	2	0.0 %	0.1 %	0.4 %	1.4 %	2.6 %	5.4 %	7.7 %	10.6 %	14.8 %	18.8 %	23.8 %	30.5 %	37.8 %	45.2 %	57.6 %	69.4 %	83.7 %	100.0 %	13.9
M1-M4	3	0.0 %	0.1 %	0.3 %	0.7 %	1.9 %	4.1 %	7.8 %	13.3 %	17.6 %	24.3 %	30.4 %	37.1 %	45.1 %	53.8 %	66.3 %	78.2 %	89.0 %	100.0 %	13.3
M1-M5	4	0.0 %	0.0 %	0.7 %	1.5 %	2.1 %	3.4 %	6.7 %	10.2 %	15.7 %	22.1 %	29.9 %	36.8 %	50.0 %	63.9 %	77.5 %	86.7 %	93.4 %	100.0 %	13.0
M1-M6	5	0.0 %	0.0 %	0.0 %	0.1 %	0.5 %	1.4 %	2.3 %	4.7 %	8.0 %	14.4 %	23.1 %	36.5 %	50.1 %	66.3 %	75.8 %	84.9 %	91.8 %	100.0 %	13.4
M2-M3	6	2.2 %	7.0 %	13.5 %	25.8 %	38.3 %	50.8 %	62.6 %	74.5 %	84.2 %	88.6 %	92.7 %	95.3 %	97.0 %	97.9 %	99.6 %	100.0 %	100.0 %	100.0 %	6.7
M2-M4	7	0.3 %	1.0 %	4.8 %	8.2 %	14.8 %	26.8 %	36.7 %	48.6 %	57.4 %	66.3 %	75.7 %	83.5 %	88.0 %	94.4 %	95.7 %	97.5 %	99.6 %	100.0 %	9.0
M2-M5	8	0.0 %	0.0 %	0.5 %	1.6 %	7.0 %	13.6 %	22.9 %	32.1 %	43.5 %	55.9 %	64.1 %	72.8 %	79.8 %	85.3 %	88.7 %	92.2 %	95.6 %	100.0 %	10.4
M2-M6	9	0.0 %	0.0 %	0.1 %	0.7 %	2.6 %	5.5 %	10.4 %	19.2 %	27.9 %	43.5 %	53.6 %	64.3 %	72.7 %	79.9 %	85.6 %	89.3 %	95.2 %	100.0 %	11.5
M3-M4	10	6.7 %	21.2 %	37.6 %	62.0 %	76.6 %	88.3 %	93.7 %	97.0 %	98.4 %	98.9 %	99.3 %	99.5 %	99.6 %	99.7 %	100.0 %	100.0 %	100.0 %	100.0 %	4.2
M3-M5	11	1.0 %	5.5 %	17.2 %	28.8 %	43.1 %	53.6 %	63.5 %	72.1 %	80.5 %	85.2 %	90.0 %	92.9 %	95.3 %	96.8 %	98.2 %	99.0 %	99.6 %	100.0 %	6.8
M3-M6	12	0.0 %	0.3 %	1.4 %	6.7 %	14.8 %	24.3 %	36.0 %	44.6 %	55.5 %	64.6 %	73.9 %	80.2 %	87.6 %	91.9 %	94.9 %	98.5 %	99.6 %	100.0 %	9.3
M4-M5	13	21.4 %	60.7 %	83.4 %	93.1 %	97.7 %	99.3 %	99.5 %	99.6 %	99.9 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	2.5
M4-M6	14	1.4 %	16.1 %	37.4 %	51.9 %	65.7 %	75.5 %	82.7 %	88.3 %	91.3 %	93.0 %	95.7 %	97.1 %	98.5 %	99.3 %	99.9 %	99.9 %	100.0 %	100.0 %	5.1
M5-M6	15	66.5 %	82.0 %	90.7 %	95.9 %	97.5 %	98.5 %	98.6 %	99.0 %	99.6 %	99.7 %	99.9 %	99.9 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	1.7
Y0-Y1	16	0.0 %	0.0 %	0.0 %	0.0 %	0.5 %	1.2 %	2.9 %	5.1 %	8.0 %	12.0 %	15.8 %	20.6 %	26.8 %	34.8 %	47.4 %	64.7 %	85.7 %	100.0 %	14.7
Y0-H1	17	0.0 %	0.8 %	1.8 %	4.1 %	8.2 %	13.0 %	18.7 %	22.8 %	27.3 %	32.0 %	38.9 %	45.9 %	53.2 %	62.4 %	72.8 %	83.7 %	91.2 %	100.0 %	12.2
H1-Y1	18	0.5 %	4.4 %	8.8 %	13.9 %	18.4 %	23.2 %	29.3 %	33.1 %	38.0 %	42.0 %	48.5 %	53.8 %	58.8 %	63.3 %	69.4 %	76.5 %	85.0 %	100.0 %	11.3

Table 5.3: Cumulative selection frequency from ranking of pairs. Trading period equal to 1 day (TP = 1day). Color coding: white < 25%, light gray 25-75%, dark gray > 75%.

Pair	ID	Top 1	Top 2	Top 3	Top 4	Top 5	Top 6	Top 7	Top 8	Top 9	Top 10	Top 11	Top 12	Top 13	Top 14	Top 15	Top 16	Top 17	Top 18	Avg. rank
M1-M2	1	0.0 %	0.0 %	0.0 %	1.8 %	8.5 %	13.4 %	18.9 %	29.3 %	43.9 %	49.4 %	59.1 %	67.7 %	73.8 %	82.3 %	87.8 %	90.9 %	93.9 %	100.0 %	10.8
M1-M3	2	0.0 %	0.0 %	0.0 %	0.6 %	2.4 %	3.7 %	7.3 %	9.1 %	12.8 %	17.7 %	20.1 %	28.7 %	37.2 %	51.2 %	68.3 %	79.9 %	93.9 %	100.0 %	13.7
M1-M4	3	0.0 %	0.0 %	0.6 %	0.6 %	1.2 %	3.0 %	6.1 %	10.4 %	15.2 %	22.0 %	30.5 %	35.4 %	44.5 %	56.1 %	68.3 %	82.9 %	91.5 %	100.0 %	13.3
M1-M5	4	0.0 %	0.0 %	0.0 %	1.2 %	1.8 %	1.8 %	4.3 %	6.1 %	12.2 %	18.9 %	28.7 %	37.2 %	52.4 %	61.0 %	67.7 %	76.2 %	90.2 %	100.0 %	13.4
M1-M6	5	0.0 %	0.0 %	0.0 %	0.0 %	1.8 %	2.4 %	5.5 %	7.3 %	11.0 %	17.1 %	26.8 %	43.3 %	56.7 %	71.3 %	79.3 %	85.4 %	90.9 %	100.0 %	13.0
M2-M3	6	1.2 %	7.3 %	13.4 %	29.3 %	43.9 %	65.2 %	78.7 %	87.2 %	89.6 %	94.5 %	96.3 %	96.3 %	98.2 %	98.8 %	100.0 %	100.0 %	100.0 %	100.0 %	6.0
M2-M4	7	0.0 %	1.2 %	3.0 %	6.1 %	13.4 %	25.6 %	37.2 %	49.4 %	60.4 %	69.5 %	78.0 %	86.6 %	90.9 %	95.7 %	98.2 %	98.8 %	100.0 %	100.0 %	8.9
M2-M5	8	0.0 %	0.0 %	0.6 %	0.6 %	2.4 %	9.8 %	18.9 %	28.0 %	35.4 %	53.7 %	61.6 %	68.9 %	78.7 %	85.4 %	90.2 %	92.7 %	96.3 %	100.0 %	10.8
M2-M6	9	0.0 %	0.0 %	0.0 %	1.2 %	4.3 %	7.3 %	11.6 %	21.3 %	36.6 %	47.0 %	57.9 %	65.9 %	70.7 %	75.6 %	79.3 %	84.1 %	95.1 %	100.0 %	11.4
M3-M4	10	8.5 %	17.7 %	41.5 %	72.6 %	89.6 %	93.3 %	95.7 %	98.2 %	99.4 %	99.4 %	99.4 %	99.4 %	99.4 %	99.4 %	99.4 %	100.0 %	100.0 %	100.0 %	3.9
M3-M5	11	0.0 %	3.7 %	12.8 %	22.6 %	35.4 %	50.0 %	59.8 %	70.7 %	78.0 %	82.9 %	87.2 %	91.5 %	93.9 %	97.6 %	98.2 %	100.0 %	100.0 %	100.0 %	7.2
M3-M6	12	0.0 %	0.0 %	0.0 %	6.7 %	14.6 %	25.6 %	38.4 %	51.8 %	62.8 %	67.7 %	75.0 %	80.5 %	85.4 %	87.8 %	92.7 %	97.0 %	99.4 %	100.0 %	9.1
M4-M5	13	20.7 %	59.8 %	82.9 %	92.1 %	97.0 %	98.2 %	98.8 %	99.4 %	99.4 %	99.4 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	2.5
M4-M6	14	0.6 %	18.9 %	46.3 %	59.8 %	70.1 %	76.8 %	81.7 %	86.6 %	88.4 %	92.1 %	94.5 %	97.0 %	98.8 %	98.8 %	100.0 %	100.0 %	100.0 %	100.0 %	4.9
M5-M6	15	68.9 %	89.6 %	94.5 %	96.3 %	96.3 %	97.6 %	97.6 %	99.4 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	1.6
Y0-Y1	16	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	1.2 %	2.4 %	2.4 %	3.7 %	6.1 %	6.7 %	8.5 %	11.0 %	19.5 %	36.6 %	60.4 %	81.1 %	100.0 %	15.6
Y0-H1	17	0.0 %	0.0 %	0.6 %	1.2 %	4.3 %	7.3 %	15.2 %	18.9 %	23.2 %	29.3 %	36.0 %	43.9 %	53.7 %	59.8 %	67.7 %	77.4 %	88.4 %	100.0 %	12.7
H1-Y1	18	0.0 %	1.8 %	3.7 %	7.3 %	12.8 %	17.7 %	22.0 %	24.4 %	28.0 %	33.5 %	42.1 %	49.4 %	54.9 %	59.8 %	66.5 %	74.4 %	79.3 %	100.0 %	12.2

Table 5.4: TP = 5 days

Pair	ID	Top 1	Top 2	Top 3	Top 4	Top 5	Top 6	Top 7	Top 8	Top 9	Top 10	Top 11	Top 12	Top 13	Top 14	Top 15	Top 16	Top 17	Top 18	Avg. rank
M1-M2	1	0.0 %	0.0 %	1.6 %	1.6 %	4.9 %	18.0 %	31.1 %	41.0 %	57.4 %	75.4 %	86.9 %	90.2 %	90.2 %	93.4 %	95.1 %	95.1 %	98.4 %	100.0 %	9.2
M1-M3	2	0.0 %	0.0 %	0.0 %	0.0 %	1.6 %	3.3 %	8.2 %	11.5 %	13.1 %	23.0 %	24.6 %	36.1 %	52.5 %	72.1 %	77.0 %	90.2 %	95.1 %	100.0 %	12.9
M1-M4	3	0.0 %	0.0 %	1.6 %	1.6 %	1.6 %	1.6 %	3.3 %	4.9 %	6.6 %	9.8 %	14.8 %	31.1 %	41.0 %	47.5 %	62.3 %	78.7 %	91.8 %	100.0 %	14.0
M1-M5	4	0.0 %	0.0 %	1.6 %	1.6 %	1.6 %	1.6 %	6.6 %	8.2 %	13.1 %	18.0 %	31.1 %	41.0 %	47.5 %	60.7 %	67.2 %	78.7 %	80.3 %	100.0 %	13.4
M1-M6	5	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	1.6 %	9.8 %	14.8 %	23.0 %	26.2 %	39.3 %	57.4 %	70.5 %	77.0 %	77.0 %	82.0 %	100.0 %	13.2
M2-M3	6	0.0 %	1.6 %	6.6 %	14.8 %	45.9 %	62.3 %	80.3 %	88.5 %	91.8 %	95.1 %	96.7 %	96.7 %	98.4 %	98.4 %	98.4 %	100.0 %	100.0 %	100.0 %	6.2
M2-M4	7	0.0 %	1.6 %	3.3 %	4.9 %	8.2 %	18.0 %	24.6 %	41.0 %	47.5 %	57.4 %	75.4 %	78.7 %	86.9 %	91.8 %	93.4 %	96.7 %	96.7 %	100.0 %	9.7
M2-M5	8	0.0 %	0.0 %	0.0 %	0.0 %	1.6 %	4.9 %	13.1 %	24.6 %	39.3 %	50.8 %	62.3 %	68.9 %	75.4 %	78.7 %	83.6 %	90.2 %	100.0 %	100.0 %	11.1
M2-M6	9	0.0 %	0.0 %	0.0 %	0.0 %	3.3 %	9.8 %	14.8 %	26.2 %	34.4 %	42.6 %	52.5 %	62.3 %	63.9 %	72.1 %	75.4 %	82.0 %	96.7 %	100.0 %	11.6
M3-M4	10	1.6 %	13.1 %	34.4 %	68.9 %	90.2 %	93.4 %	95.1 %	96.7 %	98.4 %	98.4 %	98.4 %	98.4 %	98.4 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	4.1
M3-M5	11	3.3 %	3.3 %	9.8 %	23.0 %	32.8 %	44.3 %	54.1 %	63.9 %	78.7 %	82.0 %	85.2 %	86.9 %	91.8 %	93.4 %	98.4 %	98.4 %	100.0 %	100.0 %	7.5
M3-M6	12	0.0 %	0.0 %	1.6 %	11.5 %	18.0 %	29.5 %	50.8 %	55.7 %	65.6 %	75.4 %	82.0 %	83.6 %	83.6 %	90.2 %	93.4 %	96.7 %	100.0 %	100.0 %	8.6
M4-M5	13	19.7 %	72.1 %	88.5 %	91.8 %	93.4 %	96.7 %	96.7 %	96.7 %	98.4 %	98.4 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	2.5
M4-M6	14	0.0 %	16.4 %	50.8 %	67.2 %	75.4 %	86.9 %	88.5 %	88.5 %	88.5 %	91.8 %	95.1 %	98.4 %	98.4 %	98.4 %	100.0 %	100.0 %	100.0 %	100.0 %	4.6
M5-M6	15	75.4 %	91.8 %	95.1 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	1.4
Y0-Y1	16	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	1.6 %	3.3 %	3.3 %	4.9 %	6.6 %	9.8 %	16.4 %	36.1 %	59.0 %	80.3 %	100.0 %	15.8
Y0-H1	17	0.0 %	0.0 %	1.6 %	1.6 %	4.9 %	8.2 %	9.8 %	16.4 %	18.0 %	19.7 %	24.6 %	34.4 %	52.5 %	62.3 %	73.8 %	78.7 %	88.5 %	100.0 %	13.0
H1-Y1	18	0.0 %	0.0 %	4.9 %	11.5 %	16.4 %	21.3 %	21.3 %	24.6 %	31.1 %	36.1 %	39.3 %	47.5 %	52.5 %	54.1 %	68.9 %	78.7 %	90.2 %	100.0 %	12.0

Table 5.5: TP = 20 days

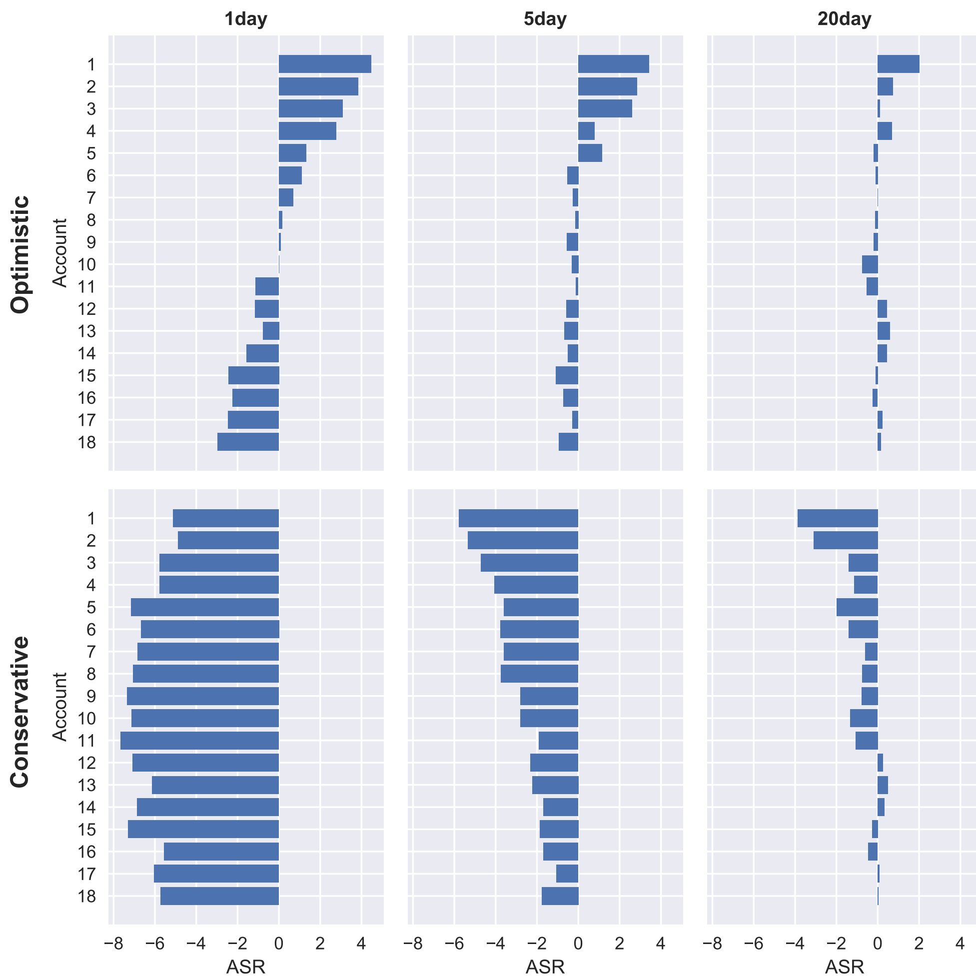


Figure 5.1: Adjusted Sharpe Ratio (ASR) for each account in base case (95% threshold, 0.02 USD bid-ask), for optimistic and conservative case.

Chapter 6

Conclusion

In this thesis, we propose an intradaily spread trading strategy based on a stochastic process model. We go on to examine whether this strategy can be profitably applied in Brent Crude oil futures markets. Under optimistic assumptions, our strategy achieves a maximum Sharpe ratio of 4.3, which is in line with the existing literature of stochastic approaches. For energy commodities and using daily data, Cummins and Bucca (2012) achieved Sharpe ratios of around 2 for the top 10 strategies in the 2003-2010 period. In the case of oil companies and using intradaily data in the 2013-2015 period, Liu et al. (2016) achieved Sharpe ratios of up to 7 using a "doubly mean-reverting" approach which has in part inspired this thesis.

However, under conservative assumptions, our Sharpe ratios are negative for all parameter choices. Although our strategy may be highly profitable under optimistic assumptions, we emphasize that results are very sensitive to small changes in bid-ask spreads and the timing of trade execution. In particular, we question the assumption of price simultaneity (both generating signals and trading on the same prices) used in e.g. Liu et al. (2016).

Top-ranked pairs are consistently the most profitable pairs in the backtest. This could indicate that the ranking function actually identifies pairs which are superior. However, we note that the top pairs are typically comprised of medium-term contracts, i.e. M3, M4, M5 and M6. In our empirical study, we have shown that contracts with longer time to maturity are less liquid than closer contracts. Because of this, we hypothesize that we may be underestimating the size of bid-ask spreads for our most profitable pairs, and hence might be overestimating profits. In our view, this should motivate a conservative approach to trading strategy assumptions.

Although profitable intraday spread trading strategies have been documented in e.g. Liu et al. (2016), we advise caution when backtesting such strategies. After studying 5-min data of Brent Crude futures, we admit that there are several factors affecting intradaily pricing that we are not able to explain well enough without order book data. This includes the size and variability of bid-ask spreads, order flow dynamics, timing of signal generation and trade execution, simultaneity of prices and so forth. We acknowledge that many of these factors may affect spread trading profits, and without a robust way to describe them, we argue that a cautious approach should be taken when implementing these parameters in a backtest. This is the reason why we let the conservative case dictate our final conclusion - namely, that intraday spread trading in ICE Brent Crude futures based on the stochastic process model put forward in this thesis is not profitable.

6.1 Further work

Even though statistical arbitrage strategies have been studied in academic papers for several decades, there is still room for innovative work. To the best of our knowledge, intraday approaches to spread trading are still only lightly covered in the academic literature. We propose four main branches for further work on intraday spread trading.

1. **Stochastic process models with non-Gaussian distributions:** Most of the current stochastic process approaches evolve around modelling the spread using an Ornstein-Uhlenbeck (OU) process, which assumes a Gaussian distribution. Due to the leptokurtic nature of log spreads, models based on other distributions might be more appropriate. In particular, conditional modelling approaches which take into account intraday patterns in distributions of returns would be of interest.

2. **Order book data for backtesting:** None of the papers we find in the literature currently incorporate order book data in testing strategies. When using daily data and trading liquid instruments, this is acceptable; but for intraday strategies (particularly those with short holding periods) order book data of Level-1 (or preferably Level-2) is necessary to achieve high confidence in the results.

3. **Order book data for spread signals:** In the literature, the last observed close price of both legs in the spread are commonly used to calculate the log spread for signal generation. With order book data, other approaches could be studied to identify whether they contain an additional informational value in generating trading signals. Some suggestions are:

   (a) The mid-price (average of bid and ask) in both legs to form log spreads.

   (b) Log spreads based on bid in one leg and ask in the other, e.g. $log(P_{A,ask}) - log(P_{B,bid})$. In this way, one could monitor spreads that are "immediately" executable.

   (c) Log spreads based on a volume weighted average of bid and ask prices in the order books of both legs (up/down to a certain level in the book).

4. **Extensions to more general intraday statistical arbitrage models:** We have limited our study to: i) Calendar spreads in Brent Crude, and ii) Only trading a single pair of securities in each account (spread trading). A possible extension would be to consider related products (e.g. WTI crude oil, heating oil, gasoil) and portfolios of securities.

References

Alexander, C. (2008). *Market risk analysis, volume i-iv.* John Wiley & Sons.

Andersen, T. G., & Bollerslev, T. (2014). Intraday periodicity and volatility persistence in financial markets. *Journal of Empirical Finance, Volume 4, Issues 2–3, June 1997, Pages 115-158.*

Avellaneda, M., & Lee, J.-H. (2010). Statistical arbitrage in the us equities market. *Quantitative Finance, Volume 10, Issue 7.*

Bertram, W. (2010). Analytic solutions for optimal statistical arbitrage trading. *Physica A: Statistical Mechanics and its Applications 398(11): 2234-2254.*

Brooks, C. (2014). *Introductory econometrics for finance.* Cambridge University Press.

Caldeira, J. F., & Moura, G. V. (2013). Selection of a portfolio of pairs based on cointegration: A statistical arbitrage strategy. *Brazilian Review of Finance no.11, p.49–80.*

Chan, E. P. (2013). *Algorithmic trading: Winning strategies and their rationale.* John Wiley & Sons.

Clewlow, L., & Strickland, C. (2000). *Energy derivatives: Pricing and risk management.* Lacima Publications.

Cummins, M., & Bucca, A. (2012). Quantitative spread trading on crude oil and refined products markets. *Quantitative Finance, Volume 12, Issue 12.*

Do, B., & Faff, R. (2009). Does naïve pairs trading still work? http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.547.8922&rep=rep1&type=pdf. (Online; Retrieved from CiteSeerX 9-May-2018.)

Do, B., Faff, R., & Hamza, K. (2006). A new approach to modeling and estimation for pairs trading. https://www.researchgate.net/publication/267721035_A_New_Approach_to_Modeling_and_Estimation_for_Pairs_Trading. (Online; Retrieved from ResearchGate 22-March-2018.)

Dunis, C., Rudy, J., Giorgioni, G., & Laws, J. (2010). Statistical arbitrage and high-frequency data with an application to eurostoxx 50 equities. *Working paper, Liverpool Business School.*

Elliott, R. J., Van Der Hoek, J., & Malcolm, W. P. (2005). Pairs trading. *Quantitative Finance, Volume 5, Issue 3.*

Gatev, E., Goetzmann, W. N., & Rouwenhorst, K. G. (2006). Pairs trading: Performance of a relative value arbitrage rule. *Yale ICF Working Paper No. 08-03.*

Krauss, C. (2017). Statistical arbitrage pairs trading strategies: Review and outlook. *Journal of Economic Surveys, Vol. 31, No. 2.*

Lewis, M. (2014). *Flash boys: A wall street revolt.* W. W. Norton Company.

Liu, B., Chang, L.-B., & Geman, H. (2016). Intraday pairs trading strategies on high frequency data: the case of oil companies. *Quantitative Finance*.

McDonald, R. L. (2014). *Derivatives markets*. Pearson Education.

Rad, H., Low, R. K. Y., & Faff, R. (2015). The profitability of pairs trading strategies: distance, cointegration, and copula methods. *Working paper, University of Queensland*.

Rogers, I. C. G., & Zane, O. (1998). Designing and estimating models of high-frequency data. *University of Bath and First National Bank of Chicago*.

Sharpe, W. F. (1966). Mutual fund performance. *The Journal of Business, Vol. 39, No. 1, Part 2: Supplement on Security Prices*.

StataCorp. (2017). *Stata 15 base reference manual*. Stata Press. https://www.stata.com/manuals/tswntestq.pdf.

Vidyamurthy, G. (2004). *Pairs trading: Quantitative methods and analysis*. John Wiley & Sons.

Appendix A

Acronyms

ACF Autocorrelation Function

ADF Augmented Dickey-Fuller (test)

API Application Programming Interface

ASR Adjusted Sharpe Ratio

BA spread Bid-Ask spread

BRN Brent

CAGR Compound Annual Growth Rate

CB Confidence Bound

CVaR Conditional Value at Risk

ICE Intercontinental Exchange

IS In-sample

LTV Long-Term Variance

MA Moving Average

MCR Mean Crossover Rate

MDD Maximum Drawdown

OS Out-of-sample

OLS Ordinary Least Squares (Regression)

P&L Profit & Loss statement (Income Statement)

Q-Q Quantile-Quantile (plot)

SD Standard Deviation

STV Short-Term Variance

TP Trading Period

VaR Value at Risk

VWAP Volume Weighted Average Price

Appendix B

ICE Brent Crude futures contracts studied

2015	2016	2017	2018	2019	2020	2021
ICE BRN MAR-2015	ICE BRN JAN-2016	ICE BRN JAN-2017	ICE BRN JAN-2018	ICE BRN JAN-2019	ICE BRN JAN-2020	ICE BRN JUN-2021
ICE BRN APR-2015	ICE BRN FEB-2016	ICE BRN FEB-2017	ICE BRN FEB-2018	ICE BRN FEB-2019	ICE BRN JUN-2020	ICE BRN DEC-2021
ICE BRN MAY-2015	ICE BRN MAR-2016	ICE BRN MAR-2017	ICE BRN MAR-2018	ICE BRN MAR-2019	ICE BRN DEC-2020	
ICE BRN JUN-2015	ICE BRN APR-2016	ICE BRN APR-2017	ICE BRN APR-2018	ICE BRN APR-2019		
ICE BRN JUL-2015	ICE BRN MAY-2016	ICE BRN MAY-2017	ICE BRN MAY-2018	ICE BRN MAY-2019		
ICE BRN AUG-2015	ICE BRN JUN-2016	ICE BRN JUN-2017	ICE BRN JUN-2018	ICE BRN JUN-2019		
ICE BRN SEP-2015	ICE BRN JUL-2016	ICE BRN JUL-2017	ICE BRN JUL-2018	ICE BRN JUL-2019		
ICE BRN OCT-2015	ICE BRN AUG-2016	ICE BRN AUG-2017	ICE BRN AUG-2018	ICE BRN AUG-2019		
ICE BRN NOV-2015	ICE BRN SEP-2016	ICE BRN SEP-2017	ICE BRN SEP-2018	ICE BRN SEP-2019		
ICE BRN DEC-2015	ICE BRN OCT-2016	ICE BRN OCT-2017	ICE BRN OCT-2018	ICE BRN OCT-2019		
	ICE BRN NOV-2016	ICE BRN NOV-2017	ICE BRN NOV-2018	ICE BRN NOV-2019		
	ICE BRN DEC-2016	ICE BRN DEC-2017	ICE BRN DEC-2018	ICE BRN DEC-2019		

Table B.1: An overview of all ICE BRN contracts studied in the Jan-2015 to Apr-2015 period.

Appendix C

Pair combinations

PairsID	Contract A	Contract B
1	ICE BRN M1	ICE BRN M2
2	ICE BRN M1	ICE BRN M3
3	ICE BRN M1	ICE BRN M4
4	ICE BRN M1	ICE BRN M5
5	ICE BRN M1	ICE BRN M6
6	ICE BRN M2	ICE BRN M3
7	ICE BRN M2	ICE BRN M4
8	ICE BRN M2	ICE BRN M5
9	ICE BRN M2	ICE BRN M6
10	ICE BRN M3	ICE BRN M4
11	ICE BRN M3	ICE BRN M5
12	ICE BRN M3	ICE BRN M6
13	ICE BRN M4	ICE BRN M5
14	ICE BRN M4	ICE BRN M6
15	ICE BRN M5	ICE BRN M6
16	ICE BRN Y0	ICE BRN Y1
17	ICE BRN Y0	ICE BRN H1
18	ICE BRN H1	ICE BRN Y1

Table C.1: Overview of all pair combinations which are considered for trading in each ranking. The contract names are given on a *relative* basis, but *absolute* contracts are used as underlying price series when a timestamp for trading is given. E.g. would Contract A for PairsID 1 at 2 January 2018 be the ICE BRN MAR-18 contract.

Appendix D

Correlation matrices

	M1	M2	M3	M4	M5	M6	Y0	H1	Y1
M1	1.000	0.980	0.965	0.945	0.922	0.899	0.951	0.868	0.861
M2	0.980	1.000	0.969	0.949	0.928	0.904	0.954	0.873	0.866
M3	0.965	0.969	1.000	0.951	0.931	0.910	0.949	0.876	0.868
M4	0.945	0.949	0.951	1.000	0.932	0.911	0.937	0.876	0.866
M5	0.922	0.928	0.931	0.932	1.000	0.912	0.919	0.872	0.859
M6	0.899	0.904	0.910	0.911	0.912	1.000	0.899	0.862	0.846
Y0	0.951	0.954	0.949	0.937	0.919	0.899	1.000	0.887	0.885
H1	0.868	0.873	0.876	0.876	0.872	0.862	0.887	1.000	0.889
Y1	0.861	0.866	0.868	0.866	0.859	0.846	0.885	0.889	1.000

Table D.1: Correlation matrix of log returns for the 5-minute intraday data.

	M1	M2	M3	M4	M5	M6	Y0	H1	Y1
M1	1.000	0.998	0.996	0.994	0.992	0.990	0.948	0.952	0.941
M2	0.998	1.000	0.999	0.998	0.997	0.995	0.951	0.958	0.949
M3	0.996	0.999	1.000	1.000	0.999	0.997	0.952	0.961	0.953
M4	0.994	0.998	1.000	1.000	1.000	0.999	0.955	0.965	0.958
M5	0.992	0.997	0.999	1.000	1.000	1.000	0.956	0.968	0.962
M6	0.990	0.995	0.997	0.999	1.000	1.000	0.957	0.971	0.966
Y0	0.948	0.951	0.952	0.955	0.956	0.957	1.000	0.992	0.978
H1	0.952	0.958	0.961	0.965	0.968	0.971	0.992	1.000	0.995
Y1	0.941	0.949	0.953	0.958	0.962	0.966	0.978	0.995	1.000

Table D.2: Correlation matrix of log returns for the daily data.

Appendix E

Engle-Granger routine for testing cointegration of time series

The concept of cointegration has already been described in section 2.2.2, and in this section, we focus on the Engle-Granger routine to testing for cointegration of two securities A and B. We will only briefly describe the routine and the reader is referred to Brooks (2014, p.361-363) for further technical details. In order to verify that the (log) price series are cointegrated, we follow a two-step algorithm:

1. Verify that the order of integration for each of the two time series is one, i.e. I(1)

2. Test for cointegration by testing the cointegrating residuals for stationarity

E.1 Testing for I(1) process in both time series

The Augmented Dickey-Fuller (ADF) test is used to test for a unit root in a time series y_t. The testing procedure is applied by estimating the regression model described in equation E.1, of which p is a chosen number of lags. In this thesis, we use the Akaike Information Criterion (AIC) to choose the number of lags.

$$\Delta y_t = \psi y_{t-1} + \sum_{i=1}^{p} \alpha_i \Delta y_{t-i} + u_t \tag{E.1}$$

The ADF statistic is now defined in equation E.2. Relevant critical values can be found in the statistical tables in appendix of Brooks (2014) or given in standard statistical software packages. The null hypothesis is a unit root is present in y_t. Thus, a rejection would result in concluding that the series is stationary (or trend-stationary depending on the exact test).

$$\text{ADF statistic} = \frac{\widehat{\psi}}{\widehat{SE(\psi)}} \tag{E.2}$$

If a unit roots are found in the series, this implies that it is not stationary. We then take the first difference of the series and run the test again. If the first differences are stationary, this would imply that the original series is integrated of order one, I(1).

E.2 Testing for stationarity in residuals of cointegrating regression

Two series are cointegrated if a linear combination of them are found to be a stationary time series. The Engle-Granger approach is to run an OLS regression (the cointegrating regression) of one time series onto the other, and then test to see if the residuals are stationary. If they are, this would imply that the series are cointegrated. In our case, let $X_{i,t} = log(P_{i,t})$. The resulting regression equation is found in (E.3).

$$X_{i,t} = \mu + \kappa X_{j,t} + \epsilon_t \tag{E.3}$$

In order to test for cointegration, we use the Augmented Dickey-Fuller (ADF) test on the residuals (ϵ_t). The test statistic is calculated as before, but it should be noted that the critical values are now changed because we are applying it to the residuals of a regression. The reason for this is that the test is now operating on the residuals of an estimated model rather than raw data (Brooks, 2014, p.377).

Because the Engle-Granger method is based on an OLS regression, it does not find all possible cointegrated relationships between the two time series. In fact, the Engle-Granger approach seeks out the stationary linear combination that has the minimum variance (Alexander, 2008, Vol II, p.235).

Appendix F

Additional plots

F.1 Spreads from cointegrating regression on daily data

For the long-term spread, we test for cointegration on daily settlement data for the whole Jan-2000 to Apr-2018 period. The results of the Engle-Granger routine was presented in section 3.2.2 and all pairs were shown to be cointegrated with high significance levels. Residuals for all pairs are shown in figure F.1, and confirm that the spreads are stationary in the long-term (with some exceptions of rapid changes in term structure in 2009 and 2014-2015).

F.2 Spreads from cointegrating regressions on M1-M2 pair for 5-minute data

For the short-term spread, we test for cointegration with 5-minute data on 40 one-month intervals in the Jan-2015 to Apr-2018 period. The residuals time series and histograms for the M1-M2 pair (PairsID 1) are shown in figure F.2-F.5. Some plots indicate mean-reversion in residuals, but most do not. We thus conclude that the M1-M2 pair cannot be deemed cointegrated on a short-term basis. Similar results are found for all the other pairs, and plots are not included in this thesis due to the high number of plots it would result in ($18 \cdot 40 \cdot 2 = 1440$). The results for all pairs are summarized by distributions of p-values in figure 3.13 in section 3.2.2.

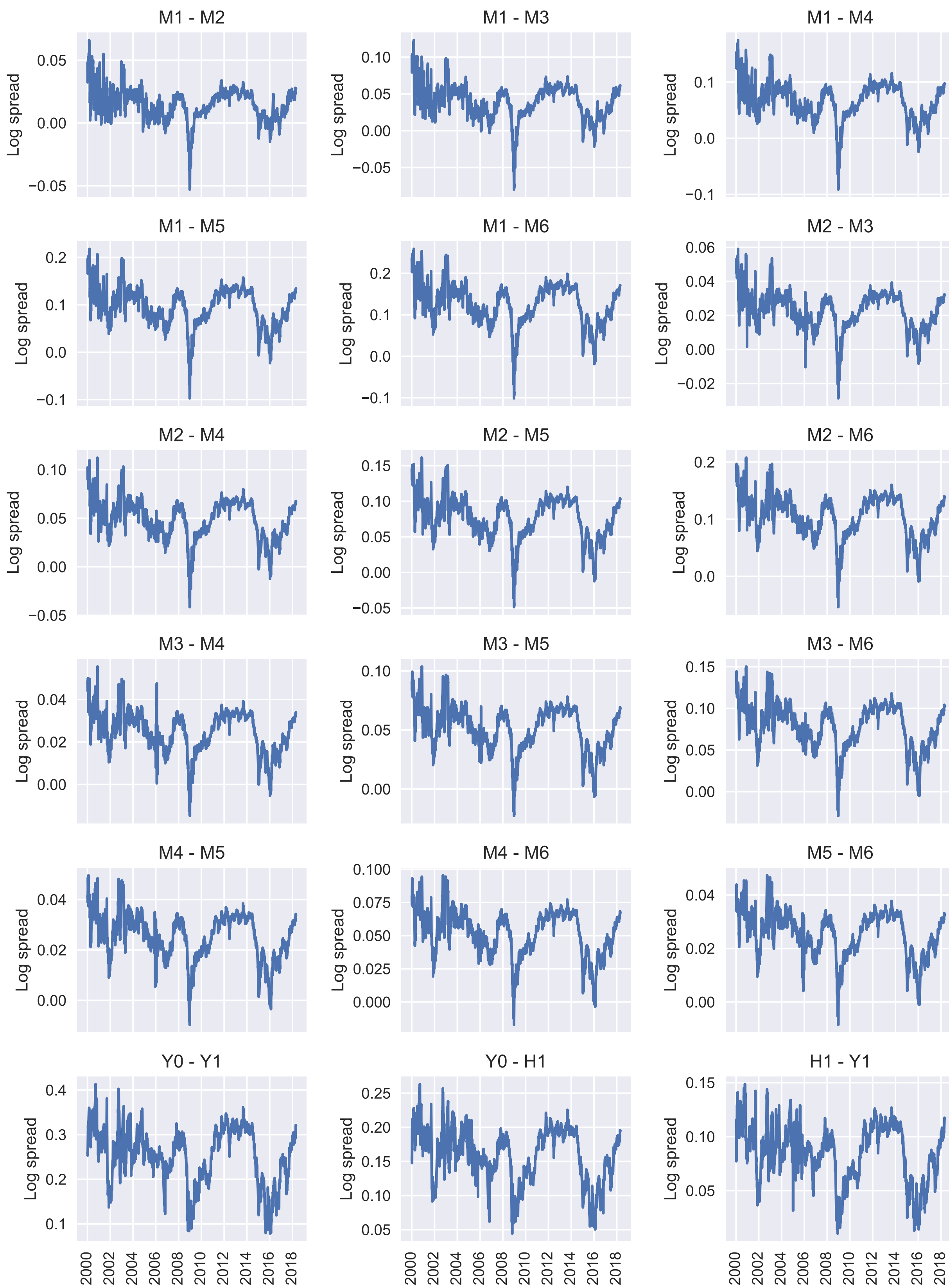


Figure F.1: Log spreads of all pairs from the cointegrating regression. The Engle-Granger results for daily data is found in table 3.5.

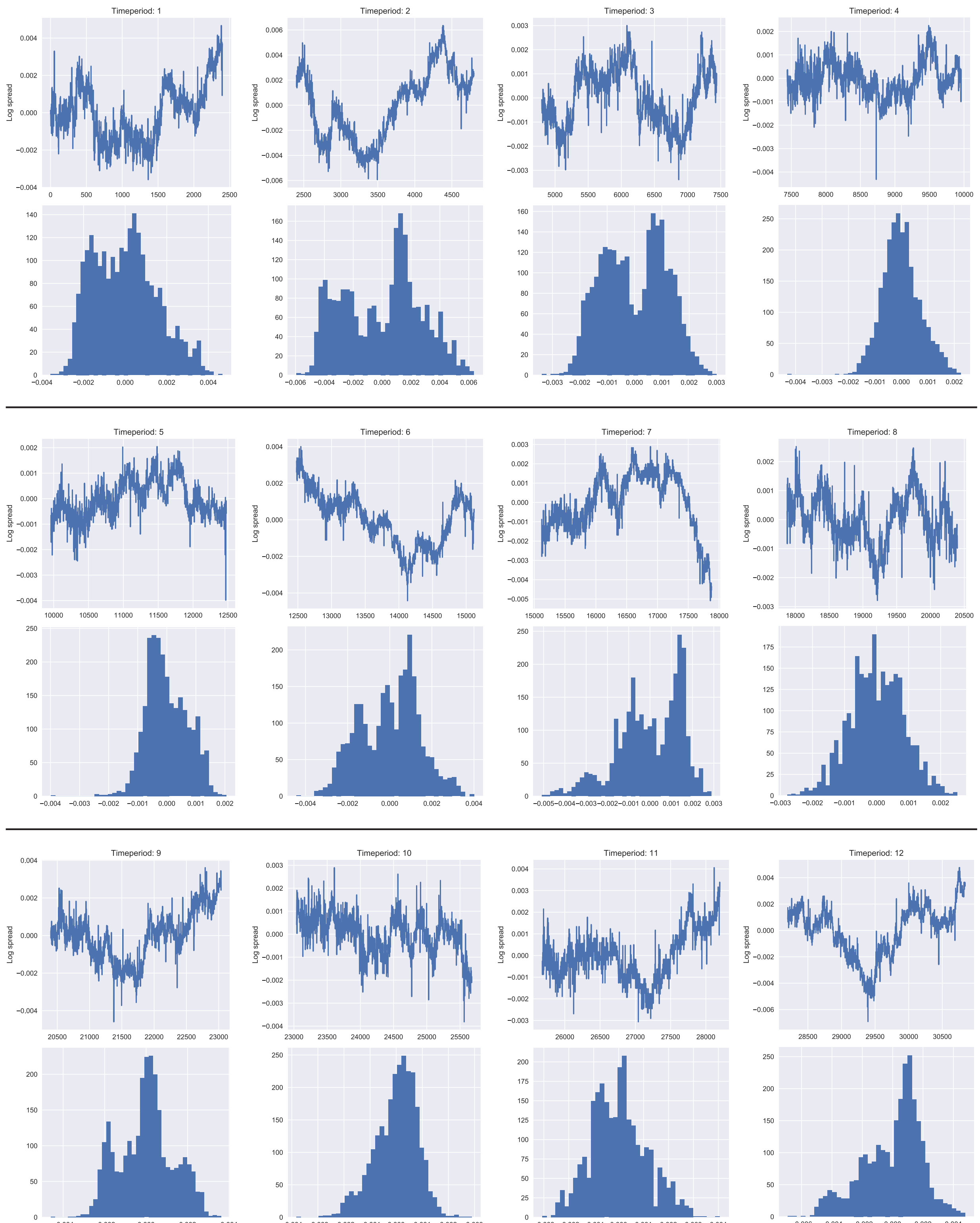


Figure F.2: Residuals of cointegrating regressions on 5-minute data for ICE BRN M1 in 12 separate one-month intervals.

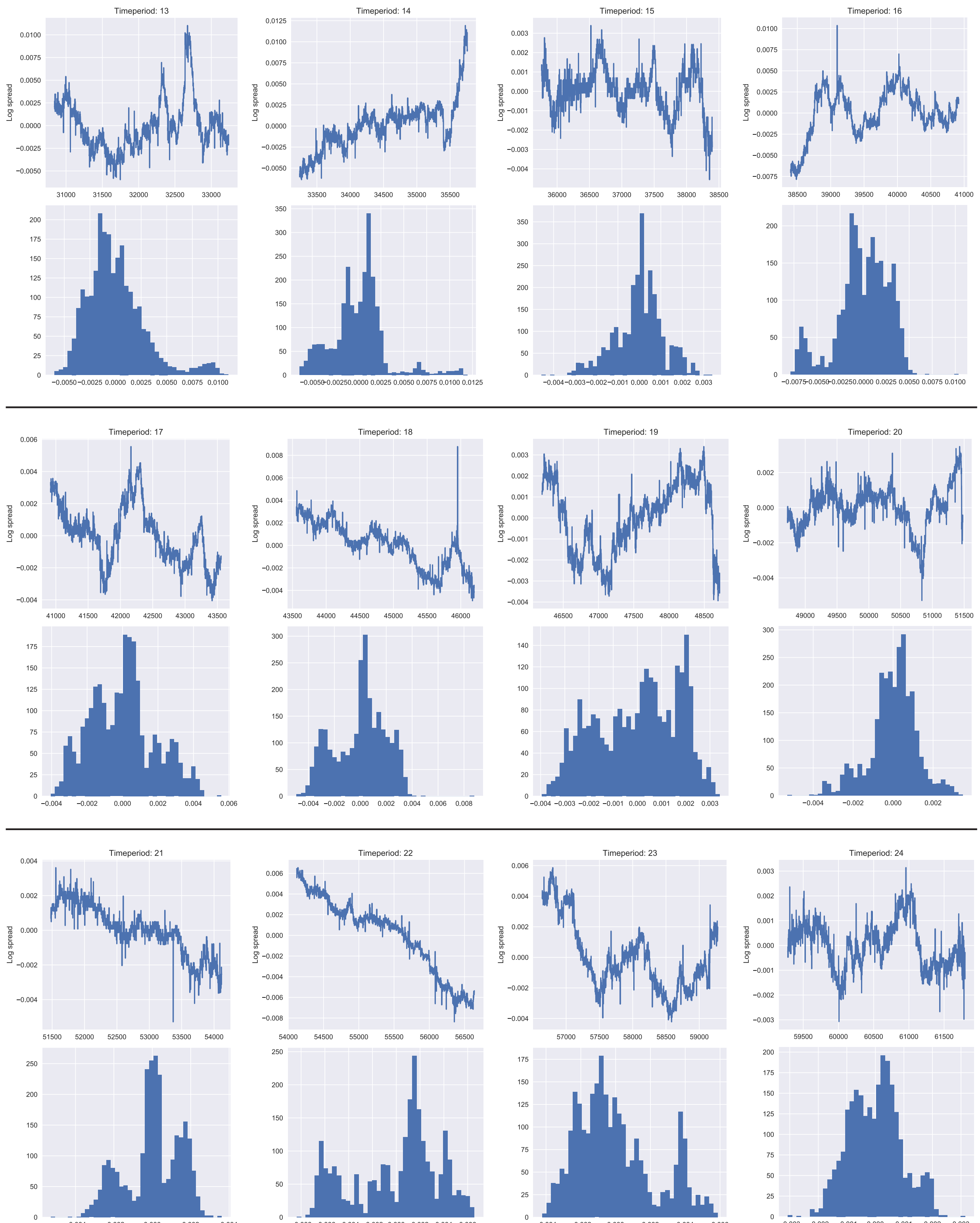


Figure F.3: Residuals of cointegrating regressions on 5-minute data for ICE BRN M1 in 12 separate one-month intervals. Continued from figure F.2.

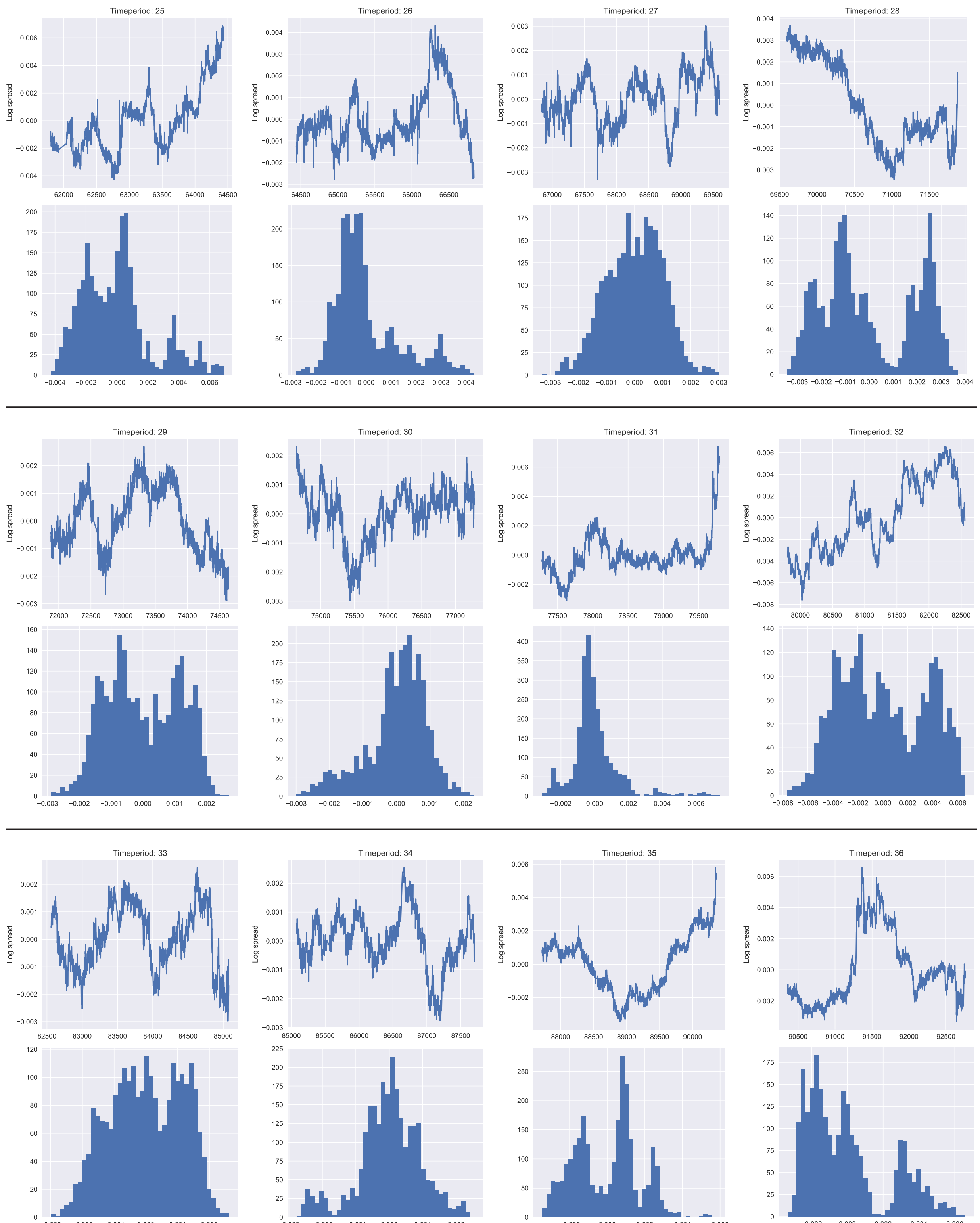


Figure F.4: Residuals of cointegrating regressions on 5-minute data for ICE BRN M1 in 12 separate one-month intervals. Continued from figure F.3.

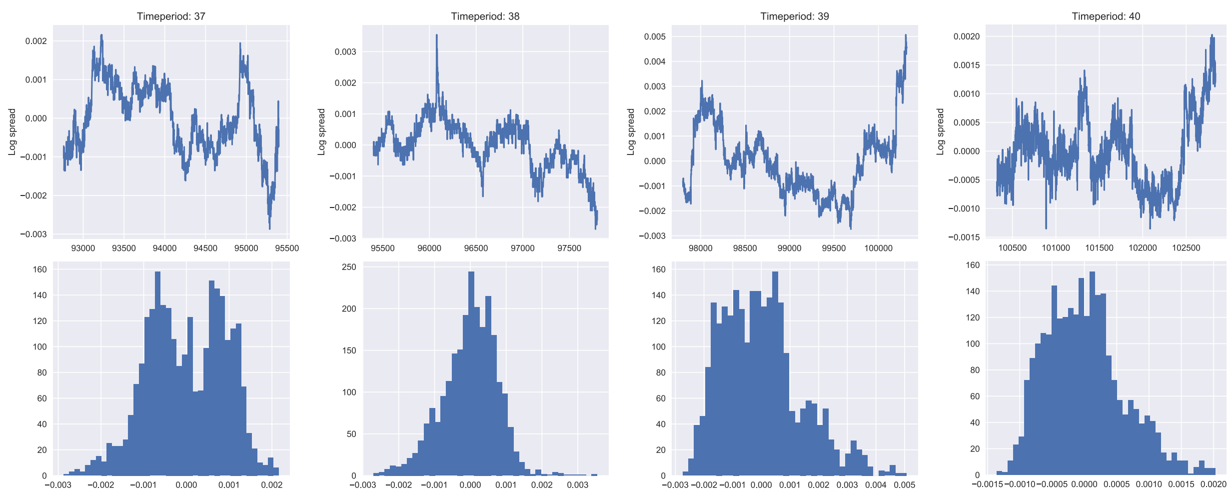


Figure F.5: Residuals of cointegrating regressions on 5-minute data for ICE BRN M1 in 4 separate one-month intervals. Continued from figure F.4.

Appendix G

Theoretical models for futures prices and calendar spreads

Futures contracts with different maturities often have different prices in the markets, and form the forward curve (or term structure) for the particular commodity (e.g. Brent Crude oil). The study of futures prices and forward curves is rightfully an own field of study, and we will only touch upon the theoretical concepts needed to understand the spreads considered in this thesis.

A simple theoretical pricing model for commodity futures shown in equation (G.1) is widely found in the literature (e.g. McDonald (2014) or Clewlow and Strickland (2000)). $S(t)$ is the spot price of the commodity, c is the continuously compounded cost of holding the spot asset (including both borrowing costs and carry costs), δ is the continuously compounded convenience yield of the asset, t is the current time and T is the time of maturity (both measured in years). γ is the roll-yield, i.e. the yield a long-investor in the future can expect to earn by holding the future to maturity if the underlying spot price of the asset does not change.

$$F(t,T) = S(t)e^{(c-\delta)(T-t)} = S(t)e^{-\gamma(T-t)} \tag{G.1}$$

Using this futures pricing model and knowledge about the underlying, namely Brent Crude oil, we can make some considerations regarding the calendar log spreads we focus on in this thesis. For two contracts A and B, the log spread $Y(t)$ is shown in equations (G.2)-(G.5).

$$\begin{aligned} Y(t) &= log\Big(F_A(t,T_A)\Big) - log\Big(F_B(t,T_B)\Big) && \text{(G.2)} \\ &= log(S_A) - \gamma_A(T_A - t) - log(S_B) + \gamma_B(T_B - t) && \text{(G.3)} \\ &= -\gamma_A(T_A - t) + \gamma_B(T_B - t) && \text{(G.4)} \\ &\approx -\gamma(T_A - t) + \gamma(T_B - t) = \gamma(T_B - T_B) && \text{(G.5)} \end{aligned}$$

In equation (G.4) the spot price falls out as we are considering calendar spreads ($S_A = S_B$), and in equation (G.4) we make an approximation by assuming that the roll yield for the two contracts A and B is equal.

The resulting $Y(t)$ show that the log spread is a function of the roll yield of the contracts (assumed to be equal) and the difference in maturity time. An important observation from equation (G.5) is that the log spread does not depend on the spot price of the underlying. Thus, if the roll return is mean reverting over time we are dealing with log spreads which should also mean-revert.

Changes in roll-yield over time come from the fact that the term structure is changing. The reasons for this is out-of-scope for this thesis, but some hypotheses can be drawn from the components of the roll yield: if the storage costs and borrowing costs stay roughly the same over time, it is the change in convenience yield which drives changes in roll-yields.

Appendix H

Estimation of parameters in Ornstein-Uhlenbeck (OU) processes

To estimate the mean reversion rate and the volatility of a process, we adopt the approach of Clewlow and Strickland (2000, p.28). The discretised version of the OU-process is described by:

$$\Delta x_t = x_t - x_{t-1} = \theta(\mu_{t-1} - x_{t-1})\Delta t + \sigma \Delta W_t \tag{H.1}$$

$$\Delta x_t = \beta_1 Z_{t-1} + \epsilon_t \tag{H.2}$$

in which $\beta_1 = \theta\Delta t$, $Z_{t-1} = \mu_{t-1} - x_{t-1}$.

Further, it can be shown that:

$$\Delta W_t \sim N(0, \Delta t) = \sqrt{\Delta t} N(0,1) \tag{H.3}$$

which implies that ϵ_t in equation (H.2) can be described as:

$$\epsilon_t \sim \sigma\sqrt{\Delta t} N(0,1) = N(0, \sigma^2 \Delta t) \tag{H.4}$$

Using all the relations described in equations (H.1)-(H.4), we can estimate the parameters of the OU process by OLS linear regression and the following equations:

$$\theta = \frac{\beta_1}{\Delta t} \tag{H.5}$$

$$\mathrm{Var}(\epsilon_t) = \sigma^2 \Delta t \Leftrightarrow \sigma = \sqrt{\frac{\mathrm{Var}(\epsilon_t)}{\Delta t}} \tag{H.6}$$

It could be noted that the σ in an OU process is quoted in absolute units, not as per cent which is the case in the case of Geometrical Brownian Motion.

Appendix I

Description of performance metrics

Mean
Mean of daily returns in the sampling period.

SD Standard Deviation of daily returns in the sampling period. Calculated using the equally weighted volatility model described in Alexander (2008, Vol II).

ASR Adjusted Sharpe Ratio. Defined in section 4.3.4 of the Methodology section.

CAGR
Compound Annual Growth Rate. Calculated using the accumulated return and sampling period length. Indicate the expected growth rate that compounds to total accumulated return.

Max DD.
Maximum Drawdown. Measures the maximum loss attained from a peak to a trough, before a new peak is attained. Used in calculating risk-adjusted metrics such as Calmar ratio.

VaR Value-at-Risk. Measure the magnitude of losses which might be incurred in a single day at a given significance level. Extensively covered in Alexander (2008, Vol IV).

CVaR
Conditional Value-at-Risk. Measure the *expected* tail loss (ETL) which might be incurred in a single day at a given significance level. Extensively covered in Alexander (2008, Vol IV).

 # trades
The number of trades incurred in the sampling period for a given trading rule.

Avg. Trade length
The average length of each trade.

Days per trade
The number of days per trade.

Appendix J

Backtesting program

J.1 Parameters used in model

Parameter	Unit	Variable name in model
Trading period length	days	trading_period_delta
Trade at same time as signal	boolean	trade_and_signal_simultaneity
Which type of observations to use for execution	str	execution_basis
Length of short ranking period	days	short_len
Number of accounts	x	account_K
Cash balance at start (total)	USDm	totalCashBalance
Threshold percentile	%	threshold_percentile
Bid/ask spread assumption	USD/contract	bid_ask_spread
Short margin	%	short_margin
Long margin	%	long_margin
Prices used for signals	str	signal_basis
Moving average multiple	x	ma_multiple
Length of long ranking period	days	long_len
Rebalancing of account balances	boolean	rebalance_at_tp_end
Time resolution of trading	str	time_resolution
Commission cost	USD/lot	commisions_pr_lot
Lot multiplier	x	lot_multiplier
Required return in addition to trading cost	%	log_req_return
Trading days per year	days	trading_days_in_year
Global start date (start of backtesting)	date	start_date_global
Global end date (end of backtesting)	date	end_date_global

Table J.1: Overview of parameters in backtesting model. The six first parameters are varied in the base case, while others are held constant. Parameters 7-10 are varied in the sensitivity analysis part of results section. The parameter variable names in the Python program is listed in the third column.

J.2 Python code

The Python code for all functions used in ranking and backtesting the pairs for all trading periods are shown on the next pages. Each function contains a small descriptional comment in the start. The program is highly specialized for computationally efficient backtesting in a specific server-setup and is not directly executable on a new computer/server without configuration of certain variables. It is mainly provided for transparency in our backtesting methodology.

```python
import pandas as pd
import numpy as np
import datetime
import pathlib
import os
from scipy.optimize import linprog
import patsy
import statsmodels.api as sm
from tqdm import tqdm


# Functions related to backtesting
def backtestPairsStrategy(account_K,totalCashBalance,trading_days_in_year,trading_period_delta,ma_multiple,short_len,\
                          long_len,threshold_percentile,rebalance_at_tp_end,trade_and_signal_simultaneity,signal_basis,\
                          execution_basis,time_resolution,time_res_int,commissions_pr_lot,bid_ask_spread,lot_multiplier,\
                          log_req_return,short_margin,long_margin,start_date_global,end_date_global,timediff_sod,\
                          timediff_eod):
    # FUNCTION DESCRIPTION: This function backtests the entire period based on the parameters inputted. This includes:
    #   - Splitting entire time period into trading periods,
    #   - Ranking all pairs at the outset of each trading period, based on formation periods set by parameters,
    #   - Backtest trading for top K pairs in all trading periods, based on the output from ranking function,
    #   - Save down equity curves and other performance metrics for all accounts,
    #   - +++

    # Set parameters
    ##Server-specific
    base_path = 'C:/Users/danie/Dropbox/Masteroppgave V18/'
    timeformat = "%Y-%m-%d %H:%M:%S"

    #Import rawAxis for time resolution and overview of all pairs
    pairs_df = pairs_df = pd.read_excel(base_path + 'Data/PairsOverview.xlsx') #Import list of pairs
    rawAxis_complete = pd.read_csv(base_path + 'Data/Timeaxis/f9t19/' + 'timeAxis-' + str(time_resolution) + '.csv')  ##import\
    #  complete time axis file from csv

```

```python
##Create a new timeaxis, with rows showing start and end date of all trading periods
tradingPeriods = createTradingPeriods(base_path, '1day', start_date_global, end_date_global,trading_period_delta)

# Initialize accountBalances and output files
accountBalances = initializeAccounts(totalCashBalance,account_K)  ##Set initial cashBalance for each of the K accounts
acc_tradingcost_accounts = initializeAccounts(0, account_K)
accountList = accountBalances.index.tolist()  # Get list of account names from accountBalances pd Series
newpath, backtest_id=initializeOutputFiles(accountList,base_path)  ##Create output files with headers, ready for appending\
#  - for all of the K accounts

# For all trading periods set
tp_N=tradingPeriods.shape[0]
neg_thetas = open((newpath+'/runinfo/negative_thetas.txt'),'w')

for i in tqdm(range(0, tp_N)):
    ##Set dates of current trading period
    start_date_period = datetime.datetime.strptime(tradingPeriods.loc[i, 'StartTime'], timeformat)
    end_date_period = datetime.datetime.strptime(tradingPeriods.loc[i, 'EndTime'], timeformat)
    if i==(tp_N-1):
        next_date_period=np.nan
    else:
        next_date_period = datetime.datetime.strptime(tradingPeriods.loc[i, 'NextStartTime'], timeformat)
    #endif

    ##Conduct ranking of securities for this trading period
    pairs_ranking, rankThreshold, lookback_start_date = rankPairs(start_date_period, base_path, pairs_df,\
                                                        time_resolution, long_len,short_len, trading_days_in_year,\
                                                        threshold_percentile,rawAxis_complete,signal_basis,\
                                                        trading_period_delta, ma_multiple,time_res_int,neg_thetas)

    ##Choose K top pairs and allocate to accounts
    k_top_pairs=pd.DataFrame(columns=['Account','PairsID','rankThreshold'])
    k_top_pairs['Account']=pd.Series(accountList)
    k_top_pairs['PairsID']=pairs_ranking.loc[0:(account_K-1)]
    k_top_pairs['rankThreshold']=rankThreshold.loc[0:(account_K-1)]
```

```
    ##For each of the K accounts, backtest the top K pairs in current period
    for account in accountList:
        ##Set pairs ID for account in this trading period
        account_PairsID=k_top_pairs[k_top_pairs['Account']==account]['PairsID'].iloc[0]

        ###Set absolute contract names of A-B pair
        daily_data = pd.read_csv(base_path + 'Data/BRN_data/Daily/1day-bfm_roll/log_spreads_for_UO_estimation/PairsID'\
                                 + str(account_PairsID) + '.csv')
        start_line = daily_data[daily_data['TradingTime'] == start_date_period.strftime(timeformat)].index[0]
        end_line = daily_data[daily_data['TradingTime'] == end_date_period.strftime(timeformat)].index[0]
        daily_data = daily_data.loc[start_line:end_line, :]
        absoluteSymbolKeys = [daily_data.loc[start_line,'ContractName_A'], daily_data.loc[start_line,'ContractName_B']]

        ###Set parameters for backtesting
        rankThreshold_val=k_top_pairs[k_top_pairs['Account']==account]['rankThreshold'].iloc[0]

        ###Backtest each absoulte contract period using the backtestContractPair function
        backtest_start_date=start_date_period+timediff_sod
        if pd.isnull(next_date_period):
            backtest_end_date = end_date_period + timediff_eod
        else:
            backtest_end_date = next_date_period + timediff_sod
        #endif

        backtest_outputData,uncommittedCash_new,acc_tradecost_new=backtestAbsolutePair(backtest_start_date,backtest_end_date,\
                                lookback_start_date, accountBalances.loc[account],absoluteSymbolKeys, time_resolution, base_path,\
                                timeformat, rankThreshold_val,rawAxis_complete, signal_basis, execution_basis,\
                                trade_and_signal_simultaneity,commissions_pr_lot, bid_ask_spread,lot_multiplier,\
                                log_req_return,short_margin,long_margin,acc_tradingcost_accounts.loc[account],\
                                trading_period_delta,ma_multiple,time_res_int)

        ###Append output data for current interval to relevant output file
        backtest_outputData.to_csv((newpath + "/" + account + '.csv'),mode='a',header=False)
        accountBalances.loc[account]=uncommittedCash_new
```

```python
            acc_tradingcost_accounts.loc[account]=acc_tradecost_new
        #endfor

        ##Redistribute resources equally across the K accounts
        if rebalance_at_tp_end:
            new_totalCashBalance = accountBalances.sum()
            accountBalances = initializeAccounts(new_totalCashBalance, account_K)
        #endif

    # endfor

    #Export parameters to txt file
    parameters_total=[start_date_global.strftime(timeformat),end_date_global.strftime(timeformat),account_K,trading_period_delta,\
                      threshold_percentile,bid_ask_spread,trade_and_signal_simultaneity,signal_basis,\
                      execution_basis,totalCashBalance,trading_days_in_year,ma_multiple,short_len,\
                      long_len,rebalance_at_tp_end,time_resolution,time_res_int,commissions_pr_lot,lot_multiplier,\
                      log_req_return,short_margin,long_margin]
    parameters_columns=['start_date_global','end_date_global','account_K','trading_period_delta','threshold_percentile',\
                        'bid_ask_spread','trade_and_signal_simultaneity','signal_basis',\
                        'execution_basis','totalCashBalance','trading_days_in_year','ma_multiple','short_len',\
                        'long_len','rebalance_at_tp_end','time_resolution','time_res_int','commissions_pr_lot',\
                        'lot_multiplier','log_req_return','short_margin','long_margin']

    text_file = open((newpath+'/runinfo/run_info.txt'),'w')

    print("Parameters used in current output:",file=text_file)
    print("-------------------------------------",file=text_file)

    for i in range(0,len(parameters_total)):
        print("{}: {}".format(parameters_columns[i], parameters_total[i]),file=text_file)
        if i==1 or i==4 or i==5 or i==8:
            print("-------------------------------------", file=text_file)
        #endif
    #endfor
    text_file.close()
```

```
    neg_thetas.close()

    return 0;


def backtestAbsolutePair(start_date, end_date, lookback_start_date, startBalance, symbolKeys, resolution, base_path,
                         timeformat,rankThreshold,rawAxis_complete, signal_basis, execution_basis,\
                         trade_and_signal_simultaneity,commissions_pr_lot,bid_ask_spread,lot_multiplier,log_req_return,\
                         short_margin,long_margin,accumulated_TradingCost,trading_period_delta,ma_multiple,time_res_int):
    # FUNCTION DESCRIPTION: This function takes in the absolute contract names of both contracts in a pair, and backtest
    # the pairs trading strategy on the particular pair, given parameters in function. This includes:
    #   - Running through all time steps
    #   - Checking for trading signals
    #   - Executing the order-handling (accounting, updating balances, including trading costs, etc)
    #   - Add all results to an array for export

    #Parameters relating to thresholds
    tradecost_pr_lot = bid_ask_spread*lot_multiplier + commissions_pr_lot
    roundtrip_tradecost=tradecost_pr_lot*2/lot_multiplier #The transaction cost in DOLLARS of a roundtrip trade,\
    #  on a per _contract_ basis (not lots)

    # Get the data for relevant contracts and relevant time period
    rawDataExecution = axisAggregation(rawdata_folder_path=(base_path+'Data/BRN_data/Intraday/f9t19/'),\
                                       rawAxis_complete=rawAxis_complete,\
                                       start_date=lookback_start_date, end_date=end_date,\
                                       symbolKeys=symbolKeys, variable=execution_basis, resolution=resolution)
    if signal_basis==execution_basis:
        rawDataSignal=rawDataExecution
    else:
        rawDataSignal = axisAggregation(rawdata_folder_path=(base_path+'Data/BRN_data/Intraday/f9t19/'),\
                                       rawAxis_complete=rawAxis_complete,\
                                       start_date=lookback_start_date, end_date=end_date,\
                                       symbolKeys=symbolKeys, variable=signal_basis, resolution=resolution)
    #endif

```

```python
##Create new Pandas DataFrame with same timeaxis as contracts (final timeaxis) and room for all variables we want to save
lookback_startRow = int(rawDataSignal[rawDataSignal['TradingTime'] == lookback_start_date.strftime(timeformat)].index[0])
startRow = int(rawDataExecution[rawDataExecution['TradingTime'] == start_date.strftime(timeformat)].index[0])  ## Set row number\
#  in rawData of the starting datetime
lastRow = rawDataExecution.index[-1]  ## Set the last number in rawData

outputData=pd.DataFrame()
outputData['TradingTime'] = rawDataExecution['TradingTime']
outputData = outputData.loc[startRow:lastRow, :]
output_list=[] ##Define list to be used in saving data for outputData dataframe

# Define variables to be used
uncommittedCash, committedCash, balance_A, balance_B, contractNameA, contractNameB, \
lastSignal_A, lastSignal_B = initializeVariables1(startBalance, symbolKeys)

previous_MV_A = 0
previous_MV_B = 0
previous_committedCash = 0
lastSignal_A = np.nan
lastSignal_B = np.nan
previous_tradeSignal = np.nan
tradeClosedInPrevIteration = False
openTrade=False

# Run prerequisite calculations to be used in execution of trading strategy
rawDataSignal['LogSpread']=np.log(rawDataSignal[symbolKeys[0]])-np.log(rawDataSignal[symbolKeys[1]])
ma_window=int(60/time_res_int)*10*trading_period_delta*ma_multiple
rawDataSignal['ma_series'] = rawDataSignal['LogSpread'].rolling(window=ma_window,min_periods=1).mean()
last_mean = rawDataSignal['ma_series'].loc[lookback_startRow:startRow].dropna().iloc[-1] #Set the last observed mean before\
#  trading period start
ma_list = rawDataSignal['ma_series'].loc[startRow:lastRow].tolist() #Convert to list for computational efficiency in for loop

new_signal_A = np.nan
new_signal_B = np.nan
acc_tradeCosts=accumulated_TradingCost
```

```
# Set DataFrames used to lists for increased computational efficiency in for loop
rawDataSignal_A_list = rawDataSignal.loc[startRow:lastRow, contractNameA].tolist()
rawDataSignal_B_list = rawDataSignal.loc[startRow:lastRow, contractNameB].tolist()
rawDataExecution_A_list = rawDataExecution.loc[startRow:lastRow, contractNameA].tolist()
rawDataExecution_B_list = rawDataExecution.loc[startRow:lastRow, contractNameB].tolist()

# Run through all time steps in current period, executing strategy
for t in range(startRow, lastRow):
    ##Getting the mean value for this time step
    last_mean = last_mean if np.isnan(ma_list[t-startRow]) else ma_list[t-startRow]

    # Setting the the prices and openTrade variable
    if trade_and_signal_simultaneity:
        new_signal_A = rawDataSignal_A_list[t-startRow]
        new_signal_B = rawDataSignal_B_list[t-startRow]
    elif t!=startRow:
        new_signal_A = rawDataSignal_A_list[t-1-startRow]
        new_signal_B = rawDataSignal_B_list[t-1-startRow]
    #endif
    lastSignal_A = lastSignal_A if np.isnan(new_signal_A) else new_signal_A
    lastSignal_B = lastSignal_B if np.isnan(new_signal_B) else new_signal_B

    executionPriceA = rawDataExecution_A_list[t-startRow]
    executionPriceB = rawDataExecution_B_list[t-startRow]
    openTrade = False if (balance_A == 0 and balance_B == 0) else True

    # Setting the trading signal. +1 = long A, short B, 0 = hold, -1 = short A, long B. 3 = close position.
    tradeSignal, log_threshold, log_tradecost = tradingSignalGenerator(t, lastRow, new_signal_A,new_signal_B,openTrade,\
                                previous_tradeSignal,last_mean,balance_A,balance_B,log_req_return,roundtrip_tradecost,\
                                rankThreshold, trade_and_signal_simultaneity)

    # Put data into output_list
    ##Update market value of committedCash, if we have a position
    if openTrade:
```

```
            # update market position of balance
            committedCash, previous_MV_A, previous_MV_B = cashSettlementUpdate(
                previous_committedCash=previous_committedCash, \
                previous_MV_A=previous_MV_A, \
                previous_MV_B=previous_MV_B, \
                price_A=lastSignal_A, \
                price_B=lastSignal_B, balance_A=balance_A, \
                balance_B=balance_B, lot_multiplier=lot_multiplier)
            previous_committedCash = committedCash  # Set for next iteration
        # endif

        ##Add all data to output_list
        log_spread_signal=np.nan if (np.isnan(new_signal_A) or np.isnan(new_signal_B)) else\
            (np.log(new_signal_A)-np.log(new_signal_B))
        log_spread_execution = np.nan if (np.isnan(executionPriceA) or np.isnan(executionPriceB)) else\
            (np.log(executionPriceA) - np.log(executionPriceB))
        output_list.append([uncommittedCash,committedCash,balance_A,balance_B,tradeSignal,acc_tradeCosts,\
                            log_spread_signal,log_spread_execution,log_threshold,log_tradecost,last_mean])

        # Check if execution is possible in current timestep - e.g. if prices are available
        if (np.isnan(executionPriceA) or np.isnan(executionPriceB)) and tradeSignal != 0:
            previous_tradeSignal = tradeSignal
            continue         # If any of prices are np.nan -> jump to next time step without changing tradeSignal
        # endif

        # Trading: executing trades and updating balances
        ##Close position
        if tradeSignal == 3:
            # Update position before exiting
            committedCash, previous_MV_A, previous_MV_B = cashSettlementUpdate(
                previous_committedCash=previous_committedCash, \
                previous_MV_A=previous_MV_A, \
                previous_MV_B=previous_MV_B, \
                price_A=executionPriceA, \
                price_B=executionPriceB, balance_A=balance_A, \
```

```
                balance_B=balance_B, lot_multiplier=lot_multiplier)

            # close position - only transfer cash back and reset balances.
            committedCash, uncommittedCash, balance_A, balance_B, transactionCost = \
                closePosition(committedCash, uncommittedCash, balance_A, balance_B, tradecost_pr_lot)

            ## SET NEW VERSIONS OF PREVIOUS MV OF A AND B
            previous_MV_A = 0
            previous_MV_B = 0

            #Adjustments
            acc_tradeCosts += transactionCost
            transactionCost=0 #Reset transactionCost

        ##Open position
        elif tradeSignal == 1 or tradeSignal == -1:
            # Set for errortesting
            errortest_totalcash_open = committedCash + uncommittedCash

            # Complete trade
            committedCash, uncommittedCash, balance_A, balance_B, transactionCost = openPosition(tradeSignal=tradeSignal,
                                                                                committedCash=committedCash, \
                                                                                uncommittedCash=uncommittedCash,
                                                                                balance_A=balance_A,
                                                                                balance_B=balance_B,
                                                                                signal_price_A=lastSignal_A, \
                                                                                signal_price_B=lastSignal_B,
                                                                                execution_price_A=executionPriceA,
                                                                                execution_price_B=executionPriceB, \
                                                                                short_margin=short_margin,
                                                                                long_margin=long_margin,
                                                                                tradeRatio_B=1,
                                                                                lot_multiplier=lot_multiplier, \
                                                                                tradecost_pr_lot=tradecost_pr_lot)
```

```
            ### SET PREVIOUS MV OF A AND B
            previous_MV_A = executionPriceA * balance_A * lot_multiplier
            previous_MV_B = executionPriceB * balance_B * lot_multiplier

            #Adjustments
            previous_committedCash = committedCash
            acc_tradeCosts += transactionCost
            transactionCost=0 #Reset transactionCost
        # endif

        previous_tradeSignal = tradeSignal
    # endfor

    ### Force out of position at existing market value if not sold out earlier
    t=lastRow
    if trade_and_signal_simultaneity:
        new_signal_A = rawDataSignal_A_list[t - startRow]
        new_signal_B = rawDataSignal_B_list[t - startRow]
    elif t != startRow:
        new_signal_A = rawDataSignal_A_list[t - 1 - startRow]
        new_signal_B = rawDataSignal_B_list[t - 1 - startRow]
    # endif
    executionPriceA = rawDataExecution_A_list[t - startRow]
    executionPriceB = rawDataExecution_B_list[t - startRow]
    openTrade = False if (balance_A == 0 and balance_B == 0) else True

    if openTrade:
        if not(np.isnan(executionPriceA)) and not(np.isnan(executionPriceB)):
            committedCash, previous_MV_A, previous_MV_B = cashSettlementUpdate(
                previous_committedCash=previous_committedCash, \
                previous_MV_A=previous_MV_A, \
                previous_MV_B=previous_MV_B, \
                price_A=executionPriceA, \
                price_B=executionPriceB, balance_A=balance_A, \
                balance_B=balance_B, lot_multiplier=lot_multiplier)
```

```python
        #endif
        committedCash, uncommittedCash, balance_A, balance_B, transactionCost = closePosition(committedCash, uncommittedCash,\
                                                                                               balance_A, balance_B, tradecost_pr_lot)
        acc_tradeCosts += transactionCost
    #endif
    log_spread_signal = np.nan if (np.isnan(new_signal_A) or np.isnan(new_signal_B)) else\
        (np.log(new_signal_A) - np.log(new_signal_B))
    log_spread_execution = np.nan if (np.isnan(executionPriceA) or np.isnan(executionPriceB)) else\
        (np.log(executionPriceA) - np.log(executionPriceB))
    output_list.append(
        [uncommittedCash, committedCash, balance_A, balance_B, 0, acc_tradeCosts, log_spread_signal,
         log_spread_execution, np.nan, np.nan, last_mean])

    #Add all data from output_list to outputData dataframe
    list_index=outputData.index
    output_cols=['uncommittedCash', 'committedCash', 'balance_A', 'balance_B', 'tradingSignal', 'accumulated_TradingCosts',\
                 'log_spread_signal', 'log_spread_execution' ,'log_threshold', 'log_tradecost', 'last_mean']
    outputData2=pd.DataFrame(output_list,columns=output_cols,index=list_index)
    outputData=pd.concat([outputData,outputData2],axis='columns')

    return outputData, uncommittedCash, acc_tradeCosts;  # return df to be appended to a final csv file for the A-B pair


def tradingSignalGenerator(t, lastRow, vwap_A, vwap_B, openTrade, previous_tradeSignal, last_mean,\
                           balance_A, balance_B,log_req_return,roundtrip_trade_cost,rankThreshold, \
                           trade_and_signal_simultaneity):
    # FUNCTION DESCRIPTION: This function generates trading signals based on currently observed prices, the current
    # position (open or closed) and thresholds. It also handles the case of blanks in the prices, giving no signal.

    tradeSignal = 0  # Default

    #Check if in last time-steps
    if t >= (lastRow - 1):
        tradeSignal = 3 if openTrade else 0
        return tradeSignal,np.nan,np.nan;
```

```python

    #Handle cases of np.nan in prices
    if (np.isnan(vwap_A) or np.isnan(vwap_B)):
        if np.isnan(previous_tradeSignal):  # This is only the case if first prices are nan
            return 0,np.nan,np.nan;
        elif trade_and_signal_simultaneity:
            return 0, np.nan,np.nan;
        else:
            return previous_tradeSignal,np.nan,np.nan;
    # endif

    # Generate tradeSignal based on strategy
    log_spread=np.log(vwap_A)-np.log(vwap_B)
    mean_price=0.5*(vwap_A+vwap_B)
    log_tradecost=2*roundtrip_trade_cost/mean_price
    log_minimumThreshold=log_tradecost+log_req_return
    log_threshold=np.max([rankThreshold,log_minimumThreshold])

    if openTrade: ##Convergence criterion - crossing the daily mean
        if (balance_A<0 and balance_B>0) and log_spread<last_mean:
            tradeSignal=3
        elif (balance_A>0 and balance_B<0) and log_spread>last_mean:
            tradeSignal=3
        #endif
    elif not (openTrade): #Entry criterion - crossing up/down through a threshold
        if log_spread>(last_mean+log_threshold):
            tradeSignal = -1
        elif log_spread<(last_mean-log_threshold):
            tradeSignal = 1
        #endif
    # endif
    return tradeSignal,log_threshold,log_tradecost;

def updateBalances(committedCash, uncommittedCash, balance_A, balance_B, committedCashInflow, uncommittedCashInflow,
                   inflow_A, inflow_B):
```

```python
    # FUNCTION DESCRIPTION: This function updates the cash and lots balances, based on the flows and previous values.

    committedCash2 = committedCash + committedCashInflow
    uncommittedCash2 = uncommittedCash + uncommittedCashInflow
    balance_A2 = balance_A + inflow_A
    balance_B2 = balance_B + inflow_B
    return committedCash2, uncommittedCash2, balance_A2, balance_B2;


def openPosition(tradeSignal, committedCash, uncommittedCash, balance_A, balance_B, signal_price_A, signal_price_B,
                 execution_price_A, execution_price_B, short_margin, long_margin, tradeRatio_B, lot_multiplier,
                 tradecost_pr_lot):
    # FUNCTION DESCRIPTION: This function opens a position based on trading signal, signal prices and execution
    # prices. This includes: * Handling margins, * Setting the optimal number of lots based on available cash.

    # Setting starting variables
    inflow_A = 0
    inflow_B = 0
    lots_A = 0
    lots_B = 0
    margin_A = short_margin
    margin_B = short_margin

    # Go long / short depending on signal
    if tradeSignal == 1:  # Order flows when trading signal is LONG A and SHORT B
        ## Set number of lots to order - based on signals
        lots_A, lots_B = numberOfLotsToOrder(price_A=signal_price_A, price_B=signal_price_B,
                                             init_cash=uncommittedCash, lot_mult=lot_multiplier,
                                             tradeRatio_B=tradeRatio_B,
                                             margin_A=long_margin, margin_B=short_margin,
                                             tradecost_pr_lot=tradecost_pr_lot)
        inflow_A = lots_A
        inflow_B = -lots_B
        margin_A = long_margin
        margin_B = short_margin
```

```
    elif tradeSignal == -1:  # Order flows when trading signal is SHORT A and LONG B
        ## Set number of lots to order - based on signals
        lots_A, lots_B = numberOfLotsToOrder(price_A=signal_price_A, price_B=signal_price_B,
                                             init_cash=uncommittedCash, lot_mult=lot_multiplier,
                                             tradeRatio_B=tradeRatio_B,
                                             margin_A=short_margin, margin_B=long_margin,
                                             tradecost_pr_lot=tradecost_pr_lot)
        inflow_A = -lots_A
        inflow_B = lots_B
        margin_A = short_margin
        margin_B = long_margin
    # endif

    ##Create transaction flows - execution
    transactionCosts = (lots_A + lots_B) * tradecost_pr_lot
    committedCashInflow = lots_A * lot_multiplier * execution_price_A * margin_A + lots_B * lot_multiplier * execution_price_B * margin_B
    if committedCashInflow > uncommittedCash:
        temp = 0

    uncommittedCashInflow = -committedCashInflow - transactionCosts
    # endif

    ##Update balances
    committedCash, uncommittedCash, balance_A, balance_B = updateBalances(committedCash, uncommittedCash, balance_A, \
                                                                          balance_B, committedCashInflow, \
                                                                          uncommittedCashInflow, inflow_A, inflow_B)
    return committedCash, uncommittedCash, balance_A, balance_B, transactionCosts;


def closePosition(committedCash, uncommittedCash, balance_A, balance_B, tradecost_pr_lot):
    #FUNCTION DESCRIPTION: This function closes any open position, and handles the updating of balances.

    # Resetting order flows
    committedCashInflow = 0
    uncommittedCashInflow = 0
```

```python
    inflow_A = 0
    inflow_B = 0

    # Close position
    ##Setting inflows
    inflow_A = -balance_A
    inflow_B = -balance_B
    transactionCosts = (np.abs(inflow_A) + np.abs(inflow_B)) * tradecost_pr_lot
    committedCashInflow = -committedCash
    uncommittedCashInflow = committedCash - transactionCosts

    ##Update balances
    committedCash, uncommittedCash, balance_A, balance_B = updateBalances(committedCash, uncommittedCash, balance_A, \
                                                                          balance_B, committedCashInflow, \
                                                                          uncommittedCashInflow, inflow_A, inflow_B)
    return committedCash, uncommittedCash, balance_A, balance_B, transactionCosts;


def cashSettlementUpdate(previous_committedCash, previous_MV_A, previous_MV_B, price_A, price_B, balance_A, balance_B,
                         lot_multiplier):
    # FUNCTION DESCRIPTION: Updates the value of the committedCash balance based on last observed prices.
    # This is also known as "marking to market" the position currently held, i.e. updating market values of position.

    ##Value of current position
    current_MV_A = price_A * balance_A * lot_multiplier
    current_MV_B = price_B * balance_B * lot_multiplier

    # Changes in positions
    increase_MV_A = current_MV_A - previous_MV_A
    increase_MV_B = current_MV_B - previous_MV_B

    # Update position
    new_committedCash = previous_committedCash + increase_MV_A + increase_MV_B
    return new_committedCash, current_MV_A, current_MV_B;
```

```python
def rankPairs(trading_start_date,base_path,pairs_df,resolution,formation_length_long,formation_length_short,\
              tradingDaysInYear,threshold_percentile,rawAxis_complete,signal_basis,trading_period_delta,ma_multiple,time_res_int,\
              neg_thetas):
    # FUNCTION DESCRIPTION: This function returns a ranked list with the PairsID of all pairs, in accordance with the ranking rules.
    # datetime objects should be based on dates, not including the time stamps! End date will be modified to include that day.

    #Parameters and import of pairs
    timeformat='%Y-%m-%d %H:%M:%S'
    pairs_list=pairs_df['PairsID'].tolist()

    #Set formation period dates based on number of days in formation periods
    trading_days_BRN=pd.read_excel(base_path+'Data/TradingDaysBRN.xlsx')
    trading_day_start_line=trading_days_BRN[trading_days_BRN['Date']==trading_start_date.strftime(timeformat)].index[0]
    start_date_long=trading_days_BRN.loc[(trading_day_start_line-formation_length_long),'Date']
    end_date_long=trading_days_BRN.loc[(trading_day_start_line-1),'Date']
    start_date_short=trading_days_BRN.loc[(trading_day_start_line-formation_length_short-trading_period_delta*ma_multiple),'Date']
    start_date_short=datetime.datetime(year=start_date_short.year,month=start_date_short.month,day=start_date_short.day,hour=9)
    end_date_short=datetime.datetime(year=trading_start_date.year,month=trading_start_date.month,day=trading_start_date.day,hour=9)

    ##Create start date for lookback period in backtesting script
    start_date_lookback_backtest = trading_days_BRN.loc[(trading_day_start_line-trading_period_delta*ma_multiple-1),'Date']
    start_date_lookback_backtest = datetime.datetime(year=start_date_lookback_backtest.year,month=start_date_lookback_backtest.month,\
                                                     day=start_date_lookback_backtest.day,hour=9)

    # Define the neccessary arrays to store data for pairs
    ranking_df=pd.DataFrame(columns=['PairsID','A','B','A_abs','B_abs','vol_L','vol_Y','MCR'])
    ranking_df['PairsID']=pairs_df['PairsID']
    ranking_df['rankThreshold']=np.nan

    for pairsID in pairs_list:
        relative_contract_A=pairs_df.loc[(pairsID-1),'A']
        relative_contract_B=pairs_df.loc[(pairsID-1),'B']
        ranking_df.loc[(pairsID-1),['A','B']]=[relative_contract_A,relative_contract_B]
```

```
#Long-term spread model
##Import the spread data array for daily data
daily_data=pd.read_csv(base_path+'Data/BRN_data/Daily/1day-bfm_roll/log_spreads_for_UO_estimation/PairsID'+str(pairsID)+'.csv')
start_line=daily_data[daily_data['TradingTime']==start_date_long.strftime(timeformat)].index[0]
end_line=daily_data[daily_data['TradingTime']==end_date_long.strftime(timeformat)].index[0]
daily_data['TradingTime']=pd.to_datetime(daily_data['TradingTime'])

##Getting names for absolute contracts in trading period
trading_start = datetime.datetime(year=trading_start_date.year, month=trading_start_date.month,day=trading_start_date.day)
trading_start_line = daily_data[daily_data['TradingTime'] == trading_start.strftime(timeformat)].index[0]
absolute_contract_A = daily_data.loc[trading_start_line, 'ContractName_A']
absolute_contract_B = daily_data.loc[trading_start_line, 'ContractName_B']
ranking_df.loc[(pairsID - 1), ['A_abs', 'B_abs']] = [absolute_contract_A, absolute_contract_B]

##Restrict daily_data DataFrame to formation period
daily_data = daily_data.loc[start_line:end_line,:]

##Calculate the volatility of the long-term spread in absolute terms
vol_spread=np.std(np.asanyarray(daily_data['SettlementSpread_t']))
ranking_df.loc[(pairsID-1),'vol_L']=vol_spread #Save parameters to an array for comparison across pairs

# -------------------------------------------------------
# Short-term spread model
##Create array with 5-min prices from absolute contract series for the relevant formation period
rawData = axisAggregation(rawdata_folder_path=(base_path + 'Data/BRN_data/Intraday/f9t19/'), \
                              rawAxis_complete=rawAxis_complete, \
                              start_date=start_date_short, end_date=end_date_short, \
                              symbolKeys=[absolute_contract_A,absolute_contract_B], variable=signal_basis,\
                              resolution=resolution)
intraday_data = pd.DataFrame()
intraday_data['TradingTime']=pd.to_datetime(rawData['TradingTime'])
ma_window=int(60/time_res_int)*10*trading_period_delta*ma_multiple
short_period_start_line = intraday_data['TradingTime'].index[0]+ma_window

##Create delta_x and x_(t-1) arrays for regression
```

```python
intraday_data['Spread_t']=np.log(rawData[absolute_contract_A])-np.log(rawData[absolute_contract_B])
intraday_data['DeltaSpread_t']=intraday_data['Spread_t'].diff(periods=1)

##Calculate MA used as mean
intraday_data['Spread_ma'] = intraday_data['Spread_t'].rolling(window=ma_window,min_periods=1).mean()

##Set Z for regression OU estimation
intraday_data['Z_t']=intraday_data['Spread_ma']-intraday_data['Spread_t']
intraday_data['Z_tm1']=intraday_data['Z_t'].shift(periods=1)

##Cut down dataset to appropriate size
intraday_data=intraday_data.loc[short_period_start_line:,:] #Cut down intraday_data to actual \
# formation period (to base short-term process on)
intraday_data = intraday_data.dropna(axis='index') #Remove lines with NaN's

##Set newDay column
newDay_list=[]
tradingtime_list=intraday_data['TradingTime'].tolist()

for i in range(0,len(tradingtime_list)):
    tempTime=tradingtime_list[i]
    if tempTime.hour==9 and tempTime.minute==0:
        newDay_list.append(True)
    else:
        newDay_list.append(False)
    #endif
#endfor
intraday_data['newDay']=pd.Series(newDay_list,index=intraday_data['TradingTime'].index)

# Estimate the number of mean-crossovers
mcr_estimate=0
Z_t=intraday_data['Z_t'].tolist() ## Converting to lists because of computational efficiency in for loops
Z_tm1=intraday_data['Z_tm1'].tolist()
newDay=intraday_data['newDay'].tolist()

```

```python
        for i in range(0,len(Z_t)):
            if not(newDay[i]):
                if (Z_t[i]>0 and Z_tm1[i]<0) or (Z_t[i]<0 and Z_tm1[i]>0) or (Z_t[i]==Z_tm1[i]):
                    mcr_estimate+=1
            #endif
        #endfor
        ranking_df.loc[(pairsID - 1), 'MCR'] = mcr_estimate  # Save parameters to an array for comparison across pairs

        ##Use OLS to estimate regression coefficients
        intraday_data = intraday_data[intraday_data['newDay']!=True] #Remove observations at start of days.\
        # Overnight deltas should not be included.
        y, X = patsy.dmatrices('DeltaSpread_t ~ Z_tm1', data=intraday_data, return_type='dataframe')
        mod_res = sm.OLS(y, X).fit() #Use OLS to fit regression estimating short-term OU process

        ##Calculate the parameters of the short-term OU process
        delta_t = np.float(1 / (tradingDaysInYear*int(60/time_res_int)*10))
        residuals = np.asarray(mod_res.resid)
        var_resid = np.var(residuals)
        theta = mod_res.params[1]/delta_t
        vol_est = np.sqrt(var_resid / delta_t)
        ranking_df.loc[(pairsID-1),'vol_Y']=vol_est #Save parameters to an array for comparison across pairs

        ##Calculate the rankThreshold to pass onto backtesting function based on input parameter threshold_percentile
        intraday_ma_deltas = intraday_data['Z_t'].abs().sort_values(ascending=True)
        ranking_df.loc[(pairsID-1),'rankThreshold']=intraday_ma_deltas.quantile(threshold_percentile,'higher')

        ##Dont allow negative values for theta - then it should not be traded. Set parameters such that it is forced to no trading.
        if theta<0:
            ranking_df.loc[(pairsID - 1), 'vol_L'] = np.power(10,9)
            ranking_df.loc[(pairsID - 1), 'MCR'] = 0
            ranking_df.loc[(pairsID - 1), 'vol_Y'] = 0
            ranking_df.loc[(pairsID - 1), 'rankThreshold'] = np.power(10,9) #Forcing no trading in period
            end_date_short_print = end_date_short.strftime("%Y-%m-%d %H:%M:%S")
            print("Date: {}, PairsID: {}, Theta: {}".format(end_date_short_print,pairsID,theta),file=neg_thetas)
        #endif
```

```python
    #endfor

    #Assign all pairs a ranking within each criteria
    pair_numbers = ranking_df['PairsID']
    long_vol_ranking=ranking_df.sort_values('vol_L')
    short_vol_ranking=ranking_df.sort_values('vol_Y',ascending=False)
    mcr_ranking=ranking_df.sort_values('MCR',ascending=False)

    long_vol_ranking['PairsID']=pair_numbers
    long_vol_ranking=long_vol_ranking.reset_index(drop=True).sort_values('PairsID')
    ranking_df['Ranking_volL']=pd.Series(long_vol_ranking.index.values)

    short_vol_ranking['PairsID']=pair_numbers
    short_vol_ranking=short_vol_ranking.reset_index(drop=True).sort_values('PairsID')
    ranking_df['Ranking_volY']=pd.Series(short_vol_ranking.index.values)

    mcr_ranking['PairsID']=pair_numbers
    mcr_ranking=mcr_ranking.reset_index(drop=True).sort_values('PairsID')
    ranking_df['Ranking_MCR']=pd.Series(mcr_ranking.index.values)

    ##Sum rankings and create new ranking for best pairs
    ranking_df['Total_ranking']=ranking_df['Ranking_volL']+ranking_df['Ranking_volY']+ranking_df['Ranking_MCR']
    ranking_df=ranking_df.sort_values('Total_ranking').reset_index(drop=True)

    ranking_df.to_csv('ranking_df_to_file.csv')

    return ranking_df['PairsID'],ranking_df['rankThreshold'],start_date_lookback_backtest;


def numberOfLotsToOrder(price_A, price_B, init_cash, lot_mult, tradeRatio_B, margin_A, margin_B, tradecost_pr_lot):
    # FUNCTION DESCRIPTION: This function returns the number of lots to order, always as positive ingtegers.

    # Take into account trading costs and reduce the initial cash balance accordingly
    trade_cost_estimate = np.floor(
        (init_cash / np.min([price_A, price_B])) * (1 / (lot_mult * np.min([margin_A, margin_B])))) * tradecost_pr_lot
```

```
    cash = init_cash - trade_cost_estimate

    def _committedCash():
        return price_A * lots_A * lot_mult * margin_A + price_B * lots_B * lot_mult * margin_B;

    # First check if it is possible to buy at least one lot of each
    lots_A = 1
    lots_B = 1
    test1 = _committedCash()
    if test1 > cash:
        return 0, 0;
    else:
        lots_A = 0
        lots_B = 0
    # endif

    # Initial solution through LP
    c = [-price_A * lot_mult * margin_A, -price_B * lot_mult * margin_B]
    A_ub = [[price_A * lot_mult * margin_A, price_B * lot_mult * margin_B]]
    b_ub = [cash]
    A_eq = [[-tradeRatio_B, 1]]
    b_eq = [0]

    x0_bnds = (0, None)
    x1_bnds = (0, None)

    res = linprog(c, A_ub=A_ub, b_ub=b_ub, A_eq=A_eq, b_eq=b_eq, bounds=(x0_bnds, x1_bnds))

    lots_A = res.x[0]
    lots_B = res.x[1]

    return np.floor(lots_A), np.floor(lots_B);


# Initialization functions
```

```python
def initializeAccounts(totalCashBalance, account_K):
    # FUNCTION DESCRIPTION: This function initialize the account balances array, used to track account balances.

    startingAccountBalances = totalCashBalance / account_K  # Equal distribution of cash on all account when starting out
    cashBalances = [startingAccountBalances]  # Initialize list of cashbalances
    accounts = ['Account1']  # Initialize list of account names

    for i in range(2, (account_K + 1)):  # Create lists
        cashBalances.append(startingAccountBalances)
        accountName = 'Account' + str(i)
        accounts.append(accountName)
    # endfor
    accountBalances = pd.Series(cashBalances, index=accounts)  # Create pandas Series with account names and balances
    return accountBalances;


def initializeOutputFiles(accountsList, base_path):
    # FUNCTION DESCRIPTION: This function initialize the output files for all accounts, used to track the equity curves
    # and other performance metrics in backtesting.

    nowTime = datetime.datetime.now()
    nowTime = nowTime.strftime("%Y-%m-%d_%H-%M-%S")

    newpath = base_path + 'Data/backtesting_output/' + nowTime
    runinfopath = newpath + '/runinfo/'
    os.makedirs(newpath)
    os.makedirs(runinfopath)

    # Create individual csv files ready for appending
    for account in accountsList:
        accountPath = newpath + "/" + account + '.csv'
        outputFile = pd.DataFrame(
            columns=['TradingTime','uncommittedCash', 'committedCash', 'balance_A', 'balance_B', 'tradingSignal',\
                     'accumulated_TradingCosts', 'log_spread_signal', 'log_spread_execution' ,'log_threshold',\
                     'log_tradecost', 'last_mean'])
```

```python
        outputFile.to_csv(accountPath)
    return newpath, nowTime;


def initializeVariables1(startBalance, symbolKeys):
    # FUNCTION DESCRIPTION: Initializes some variables in the backtestContractPair function
    uncommittedCash = startBalance  # Start out with full amount in uncommitted cash
    committedCash = 0  # Cash committed to the strategy. It is updated every trading day at settlement.
    balance_A = 0  # Initial amount of holdnings in A, in AMOUNT - not cash value
    balance_B = 0  # Initial amount of holdnings in A, in AMOUNT - not cash value
    contractNameA = symbolKeys[0]
    contractNameB = symbolKeys[1]
    lastSignal_A = np.nan
    lastSignal_B = np.nan
    return uncommittedCash, committedCash, balance_A, balance_B, contractNameA, contractNameB, lastSignal_A, lastSignal_B;

# Supporting functions
def axisAggregation(rawdata_folder_path,rawAxis_complete, start_date, end_date, symbolKeys, variable, resolution):
    # FUNCTION DESCRIPTION: Use the raw data files for a given timeaxis (e.g. 5min) to aggregate into a single df.

    ## start_date and end_date must be datetime objects, \
    ## E.g. start_date = datetime.datetime(year=2015, month=1, day=2, hour=2, minute=0, second=0)
    ## sybolKeys must be a list with strings. \
    ## E.g. symbolKeys = ['ICE BRN NOV-2015', 'ICE BRN NOV-2016'] or symbolKeys = ['ICE BRN M1', 'ICE BRN M2']

    # Set parameters
    time_format = '%Y-%m-%d %H:%M:%S'
    rawdata_path = rawdata_folder_path

    # create output array with timestamps
    rawAxis=rawAxis_complete
    start_row_timeaxis = rawAxis[rawAxis['TradingTime'] == start_date.strftime(time_format)].index[0]  ##find start row
    end_row_timeaxis = rawAxis[rawAxis['TradingTime'] == end_date.strftime(time_format)].index[0] - 1  ##find end row
    outputData = pd.DataFrame(rawAxis.loc[start_row_timeaxis:end_row_timeaxis, 'TradingTime'])  ##create new sub-df from original
    rawAxis = pd.DataFrame()  ##free rawAxis data
```

```python
    header_cols=['LineNr','TradingTime','Open','Close','VWAP','Volume','N Trades']

    # iterate through contracts
    for key in symbolKeys:
        if end_date.year==2018: ##Efficiency boost of program - dont import all rows before going towards end of series
            endrow=None
        else:
            endrow=end_row_timeaxis-start_row_timeaxis+200
        #endif
        rawAxis = pd.read_csv((rawdata_path + key + '-' + str(resolution) + '.csv'),skiprows=(start_row_timeaxis-5),\
                              nrows=endrow,names=header_cols)  ##import data file from csv
        start_row = rawAxis[rawAxis['TradingTime'] == start_date.strftime(time_format)].index[0]  ##find start row
        end_row = rawAxis[rawAxis['TradingTime'] == end_date.strftime(time_format)].index[0] - 1 ##find end row
        out_df=rawAxis.loc[start_row:end_row, variable]
        out_df.index=outputData.index
        outputData.loc[:, key] = out_df ##create new sub-df from original
    return outputData;

def createTradingPeriods(base_path, time_resolution, start_date, end_date, trading_period_delta):
    # FUNCTION DESCRIPTION: Creates a list of trading periods; based on start date, end date and length of trading
    # periods. It also makes sure that trading periods do not overlap rolling-dates, so that we achieve a new ranking
    # at the start of each new rolling-period.

    # Convert datetime objects to strings in ISO format
    timeformat = "%Y-%m-%d_%H-%M-%S"
    timeformat2 = "%Y-%m-%d %H:%M:%S"
    sd_ISO = start_date.strftime(timeformat)
    ed_ISO = end_date.strftime(timeformat)

    # Check if there exists a tradingPeriods df saved down to csv already
    csv_path = (base_path + 'Data/tradingPeriods_arrays/tp_' + sd_ISO + "-"\
                + ed_ISO + "_" + str(trading_period_delta) + "_" + str(time_resolution) + '.csv')
    my_file = pathlib.Path(csv_path)

    if my_file.is_file():
```

```python
        importData = pd.read_csv(csv_path)
        return importData;

    # If not: create new tradingPeriods df
    timeAxis_global = pd.read_csv((base_path + 'Data/Timeaxis/timeAxis-' + time_resolution + '.csv'))  ##import a total timeaxis\
    #  for relevant resolution
    start_line_timeAxis = timeAxis_global[timeAxis_global['TradingTime']==start_date.strftime(timeformat2)].index[0]
    end_line_timeAxis = timeAxis_global[timeAxis_global['TradingTime']==end_date.strftime(timeformat2)].index[0]
    timeAxis_global = timeAxis_global.loc[start_line_timeAxis:end_line_timeAxis,:]
    timeAxis_global = timeAxis_global.drop(columns=['Open','High','Low','Close','Settlement','Volume','OpenInterest',\
                                                'ContractName','Unnamed: 0'])

    tradingPeriods = pd.DataFrame(columns=['TradingPeriod', 'StartTime', 'EndTime','NextStartTime'])
    timeAxis_row = start_line_timeAxis

    ##Iterate through all tradingPeriods
    r=0
    remaining_timeAxis_rows=end_line_timeAxis-timeAxis_row+1
    while remaining_timeAxis_rows>trading_period_delta:
        tp = r + 1
        time_ax_period=timeAxis_global.loc[(timeAxis_row):(timeAxis_row + trading_period_delta - 1),:]
        newContract_index=time_ax_period[time_ax_period['NewContract']==True].index.values
        if newContract_index.size==0:
            startTime = timeAxis_global.loc[timeAxis_row, 'TradingTime']
            endTime = timeAxis_global.loc[(timeAxis_row + trading_period_delta - 1), 'TradingTime']
            nextTime = timeAxis_global.loc[(timeAxis_row + trading_period_delta), 'TradingTime']
            timeAxis_row += trading_period_delta
        elif newContract_index[0]==timeAxis_row:
            startTime = timeAxis_global.loc[timeAxis_row, 'TradingTime']
            endTime = timeAxis_global.loc[(timeAxis_row + trading_period_delta - 1), 'TradingTime']
            nextTime = timeAxis_global.loc[(timeAxis_row + trading_period_delta), 'TradingTime']
            timeAxis_row += trading_period_delta
        else:
            startTime = timeAxis_global.loc[timeAxis_row, 'TradingTime']
            endTime = timeAxis_global.loc[(newContract_index[0]-1), 'TradingTime']
```

```python
            nextTime = timeAxis_global.loc[newContract_index[0], 'TradingTime']
            timeAxis_row = newContract_index[0]
        #endif
        tradingPeriods.loc[r, ['TradingPeriod', 'StartTime', 'EndTime','NextStartTime']] = pd.Series([tp, startTime, endTime,nextTime],\
                                                                 index=['TradingPeriod','StartTime', 'EndTime','NextStartTime'])
        r+=1
        remaining_timeAxis_rows = end_line_timeAxis - timeAxis_row + 1
    # endwhile

    ##Set last element in tradingPeriods based on time that is left in timeAxis_global
    tp = r +1
    startTime = timeAxis_global.loc[timeAxis_row, 'TradingTime']
    endTime = timeAxis_global.loc[end_line_timeAxis, 'TradingTime']
    tradingPeriods.loc[r, ['TradingPeriod', 'StartTime', 'EndTime','NextStartTime']] = pd.Series([tp, startTime, endTime, np.nan],\
                                                                 index=['TradingPeriod','StartTime', 'EndTime','NextStartTime'])

    ##Save down for future usage
    tradingPeriods.to_csv(csv_path, index=False)
    return tradingPeriods;

# Risk metrics function
def ValueAtRisk(timeseries, alpha):
    # FUNCTION DESCRIPTION: Calculates VaR and CVaR given a time series and a threshold alpha.
    sorted = timeseries.sort_values(by=timeseries.columns.values[0])
    cutoff = int(np.floor(len(sorted) * alpha))

    cvarseries = sorted.iloc[0:(cutoff - 1), 0]

    var = sorted.iloc[cutoff, 0]
    cvar = cvarseries.mean()

    return var, cvar;
```