

Merton's portfolio problem, constant  
fraction investment strategy and  
frequency of portfolio rebalancing

by

JOACHIM HOLTH

**THESIS**

*for the degree of*

**MASTER OF SCIENCE**

*(Modelling and data analysis)*



*Faculty of Mathematics and Natural Sciences  
University of Oslo*

*November 2011*

*Det matematisk- naturvitenskapelige fakultet  
Universitetet i Oslo*



# Acknowledgements

I would like to thank my supervisor professor Fred Espen Benth at the department of mathematics at the University of Oslo, for giving me an interesting task to work with. I would also like to thank the students of study room B 802 in Abel's tower for their good company. Finally I would like to thank my friends and family for supporting me in stressful times. A special thanks to my sister Ida in that regard.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Background theory</b>	<b>5</b>
2.1	Stock price model . . . . .	5
2.1.1	Constant volatility . . . . .	5
2.1.2	Stochastic volatility . . . . .	6
2.2	The Sharpe ratio . . . . .	7
2.3	The Euler-Maruyama method . . . . .	8
<b>3</b>	<b>Merton's portfolio problem</b>	<b>9</b>
3.1	Introduction . . . . .	9
3.2	Solution to the problem . . . . .	9
3.3	Power utility . . . . .	11
<b>4</b>	<b>Estimation of parameters</b>	<b>13</b>
4.1	Estimation of the risky asset and riskfree asset parameters . . . . .	13
4.2	Estimation of risk aversion through VaR . . . . .	15
4.3	Calibration of the Heston model . . . . .	17
4.3.1	Introduction . . . . .	17
4.3.2	Estimation of $\nu_0$ , $\theta$ and $\kappa$ through linear regression . . . . .	18
4.3.3	Estimation of $\xi$ and $\rho$ . . . . .	20
4.3.4	Doing the calibration . . . . .	21
<b>5</b>	<b>Simulation</b>	<b>23</b>
5.1	Introduction . . . . .	23
5.2	Basic simulation model . . . . .	23
5.2.1	Introduction . . . . .	23
5.2.2	Simulation model I . . . . .	25
5.2.3	Loss of utility . . . . .	27
5.2.4	Simulation test run . . . . .	28
5.2.5	Mean loss of utility . . . . .	30
5.2.6	Portfolio return and Sharpe ratio . . . . .	36
5.3	Simulation with transaction costs . . . . .	41

5.3.1	Introduction . . . . .	41
5.3.2	Simulation model II . . . . .	41
5.3.3	Simulation model III . . . . .	43
5.3.4	Simulation test runs . . . . .	47
5.3.5	Mean loss of utility . . . . .	52
5.3.6	Portfolio return and Sharpe ratio . . . . .	58
5.4	Simulation with stochastic volatility . . . . .	67
5.4.1	Introduction . . . . .	67
5.4.2	Stochastic volatility . . . . .	67
5.4.3	Simulation model IV . . . . .	70
5.4.4	Implementation . . . . .	70
5.4.5	Simulation test run . . . . .	71
5.4.6	Mean loss of utility . . . . .	74
5.4.7	Portfolio return and Sharpe ratio . . . . .	79
<b>6</b>	<b>Conclusion</b>	<b>83</b>
	<b>Appendices</b>	<b>86</b>
<b>A</b>	<b>Additional plots</b>	<b>87</b>
A.1	Simulation model II and III . . . . .	87
A.2	Simulation model IV . . . . .	92
<b>B</b>	<b>R source code</b>	<b>95</b>
B.1	Support functions . . . . .	95
B.2	Initialization and estimation . . . . .	99
B.3	General simulation machinery . . . . .	104
B.4	Simulation model I . . . . .	107
B.4.1	Simulation machinery . . . . .	107
B.4.2	Execution . . . . .	111
B.5	Simulation model II and III . . . . .	119
B.5.1	Simulation machinery . . . . .	119
B.5.2	Execution . . . . .	126
B.6	Simulation model IV . . . . .	159
B.6.1	Simulation machinery . . . . .	159
B.6.2	Execution . . . . .	167

# Chapter 1

## Introduction

Banks, investment funds and insurance companies are examples of investors that invest money in the financial markets. Naturally, they want to make as much money as possible on their investments, but any serious investor also need to consider the risk involved. Normally, an investor is to a certain degree risk averse, that is, the investor is reluctant to invest in an asset with a potentially high upside if it means that the risk of loosing money is high as well. For example, because of their obligations towards their customers, a traditional bank or an insurance company, which invest funds on behalf of their customers in the financial market, cannot allow themselves to take too much risk. The aim of such investors is to maximize the expected returns on their investments while at same time limiting the risk involved. One way of modelling such behaviour is through the theory of stochastic control and the maximization of expected utility.

Potential objects of investment can basically be divided into two categories: risky assets, which are assets with an uncertain future return, and risk-free assets, which are assets with a beforehand known future return. Examples of risky assets are stocks, derivatives, real estate, raw materials et cetera. Examples of risk-free assets are bonds and t-bills. Depending on the degree of risk aversion, an investor may compose an investment portfolio as a mix of both risky and risk-free assets to match the level of risk the investor is comfortable with. For such a risk averse investor it is natural to ask: which allocation strategy or investment strategy will maximize the expected utility of the portfolio? This is the question that Nobel laureate in economics Robert C. Merton addressed and mathematically solved in a paper [15] in 1969 by using stochastic control. The problem is popularly known as "Merton's portfolio problem", which has become a well-studied problem in articles and literature.

The most basic version of the problem gives an investor the limited choice of investing her wealth in a risky asset and a risk-free asset. Given some additional

assumptions, Merton found that the optimal allocation strategy or trading strategy is to keep a constant fraction of the wealth in the risky asset (and hence, a constant fraction in the risk-free asset). This can be generalized to a situation with several risky assets and one risk-free asset and the conclusion is basically the same, that is to keep a constant fraction of the wealth in the risky assets. This strategy is indeed a frequently used strategy among investors. For example, the norwegian pension fund, with an approximate value of NOK 3,000 billion, uses this strategy to control risk.

From a realistic point of view, the conclusion of "Merton's portfolio problem" is based on rather stylized mathematics as well as stylized assumptions. For example, one such assumption is that the dynamics of the risky assets are assumed to follow geometric Brownian motions, implying normally distributed log returns. With real stock prices, this is usually not the case. Analysis of the distributions of real stock returns shows that the distributions have heavier or fatter "tails", which means there is a higher chance of large price changes than one would expect with the normal distribution [7].

Another problem is that the conclusion is based on a continuous mathematical framework. It is also a fact that in today's extremely liquid financial markets, stocks and other risky assets change value almost continuously in time. This means that to follow the optimal strategy an investor has to rebalance her portfolio at the same rate as the prices changes. This is obviously not very realistic seen from a practical point of view. Also, transaction costs would make such a behaviour extremely expensive.

In this thesis we will address this problem by discretization. Wikipedia defines discretization as the process of transferring continuous models and equations into discrete counterparts [5]. The discretization of the model allows for simulation. Through the simulations we want to simulate the portfolio of an investor making investment decisions according to the optimal investment strategy of constant fractions. The investor will only be allowed to rebalance her portfolio at certain discrete time points. These discrete time points will be chosen in such a way as to reflect different types of rebalancing strategies, such as daily rebalancings or monthly rebalancings.

The design of simulation models as well as the discussion of the resulting simulation runs of these models is the main focus of this thesis. Through the simulations we want to investigate how the optimal strategy performs in a more realistic setting. To compare the impact of discretization with the original continuous model, we will among other things measure the difference in utility or the loss of utility. The loss of utility will also be related to different rebalancing strategies. Regarding the different rebalancing strategies we will also calculate the Sharpe ratio for each strategy. The Sharpe ratio relates portfolio return with portfolio risk.



Basically, we will consider three different simulation models. The first model, which will serve as a basis for the other models, is a simple and rather unrealistic model, where the main purpose is to look at the impact of discretization itself. In the second model we will increase the complexity and hopefully the realism of the model by adding transaction costs. Finally, in the third simulation model, we will assume stochastic volatility. So the basic idea is to start out with a relatively simple simulation model and then gradually add more complexity, and with that, more realism.



# Chapter 2

## Background theory

### 2.1 Stock price model

We will in this thesis consider two stock price models for the modelling of risky asset prices. The basic structure of the models are similar. The difference between them lies in the assumptions about volatility. In the first model we will make the rather naive assumption of constant volatility. In the second model we will make the more realistic assumption of stochastic volatility.

#### 2.1.1 Constant volatility

A frequently used model for modelling risky asset prices is the geometric Brownian motion. If  $S_t$  denotes the price of a risky asset at time  $t$ , then  $S_t$  will follow a geometric Brownian motion if it satisfies the following stochastic differential equation (abbreviated SDE),

$$dS_t = \mu S_t dt + \sigma S_t dB_t, \quad (2.1)$$

where  $\mu$  is the drift and  $\sigma$  is the volatility of the risky asset, which we assume is constant.  $B_t$  is the stochastic process known as Brownian motion. Benth [1] defines Brownian motion as follows,

**Definition 2.1.1** Brownian motion  $B_t$  is a stochastic process starting at zero, i.e.  $B_0 = 0$ , and which satisfies the following three properties:

1. Independent increments: The random variable  $B_t - B_s$  is independent of the random variable  $B_u - B_v$  whenever  $t > s \geq u > v \geq 0$ .

2. Stationary increments: The distribution of  $B_t - B_s$  for  $t > s \geq 0$  is only a function of  $t - s$ , and not of  $t$  and  $s$  separately.
3. Normal increments: The distribution of  $B_t - B_s$  for  $t > s \geq 0$  is normal with expectation 0 and variance  $t - s$ .

The probability density function of a normally distributed variable  $X$  is

$$f_X(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \left(\frac{x - \mu}{\sigma}\right)^2\right).$$

Using Ito's formula the explicit solution of the SDE of the geometric Brownian motion can be shown to be

$$S_t = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma B_t\right). \quad (2.2)$$

### 2.1.2 Stochastic volatility

Assume instead that the volatility is non-constant and stochastic. A popular model for modelling stochastic volatility is the Heston model, proposed in 1993 by the American mathematician Steven Heston [9]. The Heston model can be stated as follows,

$$dS_t = \mu S_t dt + \sqrt{\nu_t} S_t dB_t^S, \quad (2.3)$$

$$d\nu_t = \kappa(\theta - \nu_t)dt + \xi \sqrt{\nu_t} dB_t^\nu \quad (2.4)$$

$$dB_t^S dB_t^\nu = \rho dt. \quad (2.5)$$

The SDE (2.4) is also known as the SDE of a CIR-process [3]. The CIR-process is mean-reverting, which means that in the long run, the process tends to drift towards its long-term mean  $\theta$ . The intensity of this mean-reverting tendency is scaled by the parameter  $\kappa$ . Similarly to the stochastic stock price dynamics of the constant volatility model, the stochastic behaviour of the stock price of the Heston model is driven by a Brownian motion  $B_t^S$ . Additionally, we have that the volatility process <sup>1</sup>  $\nu_t$  is driven by a Brownian motion  $B_t^\nu$ . The Brownian motion is scaled by the parameter  $\xi$ , which often is referred to as the volatility of the volatility. The last expression (2.5) tells us that these Brownian motions are assumed to be correlated with correlation coefficient  $\rho$ . This means that the

---

<sup>1</sup>Note that the process  $\nu_t$  is a variance process, not a volatility process per se. The volatility process itself is of course given as  $\sqrt{\nu_t}$ , but given the context, we will refer to (2.4) as an SDE modelling stochastic volatility.

joint distribution of the Brownian motions is described by a bivariate normal distribution with mean vector  $\boldsymbol{\mu}$  and covariance matrix  $\boldsymbol{\Sigma}$  given as

$$\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} dt.$$

## 2.2 The Sharpe ratio

The Sharpe ratio, which was introduced by Nobel laureate William F. Sharpe in 1966, is a measure of portfolio performance and as such a measure of the performance of an investor or portfolio manager. The original name of the Sharpe ratio is the reward-to-variability ratio and it measures the excess return per unit of risk of a portfolio [18]. According to Sharpe [17], there are two versions of the Sharpe ratio. We have the ex ante version, which is calculated through expected values by assuming that the future returns on the portfolio are distributed according to some known statistical distribution, and hence, is prospective, and we have the ex post version where the calculation of the ratio is based on historical portfolio returns, and hence, is retrospective. The following definition of the ex ante Sharpe ratio is based on the definition in Wikipedia [18], but with slightly altered notation to better fit into the notational scheme of this thesis.

**Definition 2.2.1** If  $X_t$  is the return on an investment portfolio and  $X_t^f$  is the return on a benchmark asset at time  $t$ , then the ex ante Sharpe ratio at time  $t$  can be defined as

$$SR_t^{\text{ea}} = \frac{E[X_t - X_t^f]}{\sqrt{\text{Var}[X_t - X_t^f]}}. \quad (2.6)$$

We observe that the nominator of the ratio is a measure of the excess return on the portfolio, whereas the denominator is a measure of the risk of the portfolio. A positive excess return means that we expect our investment portfolio to perform better than the benchmark asset and vice versa. As such, the ex ante Sharpe ratio may serve as a guide as to where we should invest our money. We also observe that an increase in the risk of the portfolio is associated with a decrease in ex ante Sharpe ratio. This is based on the common assumption that a high-risk investment should yield high profits compared to a low-risk investment. Note that if  $x_t^f = X_t^f$  is a deterministic quantity or a constant it follows that the

ex ante Sharpe ratio can be formulated as

$$SR_t^{\text{ea}} = \frac{E[X_t] - x_t^f}{\sqrt{\text{Var}[X_t]}}.$$

Sharpe [17] gives the following definition of the ex post Sharpe ratio (with slightly altered notation):

**Definition 2.2.2** Given a time series of historical returns on a portfolio  $\{x_t\}_{t=1,\dots,T}$  and a time series of historical returns on a benchmark portfolio or asset  $\{x_t^f\}_{t=1,\dots,T}$ , the ex post Sharpe ratio is defined as

$$SR_T = \frac{\bar{x} - \bar{x}^f}{\hat{\sigma}_x},$$

where  $\bar{x} = \sum_{t=1}^T x_t$  is the sample mean of the portfolio returns,  $\bar{x}^f = \sum_{t=1}^T x_t^f$  is the sample mean of the returns of the benchmark portfolio or asset and  $\hat{\sigma}_x = (T-1)^{-1/2}(\sum_{t=1}^T (x_t - \bar{x})^2)^{1/2}$  is the sample standard deviation of the portfolio returns.

## 2.3 The Euler-Maruyama method

The following presentation of the Euler-Maruyama method is based on the presentation of Kloeden and Platen [12]. Consider an Ito process

$$dX_t = a(t, X_t) dt + b(t, X_t) dB_t,$$

defined on a time interval  $[0, T]$  with initial value  $x_0$ .  $B_t$  is Brownian motion at time  $t$ . An approximate solution to this Ito process can be found through a so-called Euler approximation, also known as an Euler-Maruyama approximation. The approximation method requires the time interval to be divided into smaller subintervals, that is we need to construct a time discretization of the time interval:

$$0 = t_0 < t_1 < \dots < t_n = T.$$

According to Kloeden and Platen, the Euler approximation is a continuous time stochastic process  $\{Y_t\}_{t \in [0, T]}$ . However, the process is only calculated at the discrete time points given by the time discretization. The Euler approximation of  $X_{k+1}$  ( $X_k = X_{t_k}$ ) is defined recursively as

$$Y_{k+1} = Y_k + a(Y_k) \Delta t_k + b(Y_k) \Delta B_k,$$

with  $Y_0 = x_0$  and where  $\Delta t_k = t_{k+1} - t_k$  and  $\Delta B_k = B_{k+1} - B_k$ . We see that the Euler approximation describes a simple, iterative approximation scheme.

# Chapter 3

## Merton's portfolio problem

### 3.1 Introduction

Consider a scenario where an investor has the limited choice of investing his wealth in only two different assets: a risky asset (for example a stock) and a risk-free asset (for example a bank account). Given a limited time horizon, the goal of the investor, who is avert to risk, is to maximize the expected utility of his wealth at the end of the time horizon. How should the investor allocate and reallocate his wealth at each time point to achieve this goal? Stated a bit differently, what is the optimal investment strategy at each time point that will maximize the expected utility of the wealth at some terminal time?

### 3.2 Solution to the problem

Let the price of the risky asset at time  $t$  be denoted by  $S_t$ . The dynamics of the risky asset price is given by (2.1), which is the stochastic differential equation also known as geometric Brownian motion. The parameters  $\mu$  and  $\sigma$  represent respectively the drift and the volatility of the risky asset.  $B_t$  is the stochastic process known as Brownian motion. The price of the risk-free asset at time  $t$  is denoted by  $R_t$  and satisfies the following deterministic differential equation:

$$dR_t = rR_t dt. \tag{3.1}$$

The parameter  $r$  represents the risk-free continuously compounding interest rate. It is natural to assume that  $E[S_t] > E[R_t]$  which means that we assume  $\mu > r$ .

Let the wealth of the investor at time  $t$  be denoted by  $V_t$ . At each time point  $t$  the investor must invest a fraction  $u_t$  of his wealth in the risky asset. The remaining

wealth  $1 - u_t$  is invested in the risk-free asset. This means that the value of the risky investment at time  $t$  is  $u_t V_t$  and that the value of the risk-free investment is  $(1 - u_t)V_t$ . The stochastic differential equation of the wealth or portfolio value is then simply

$$\begin{aligned} dV_t &= du_t V_t + d(1 - u_t)V_t = \mu u_t V_t dt + \sigma u_t V_t dB_t + r(1 - u_t)V_t dt \\ &= (\mu u_t + r(1 - u_t))V_t dt + \sigma u_t V_t dB_t. \end{aligned} \quad (3.2)$$

The object now is to find the optimal allocation strategy  $u_t$  at each time point  $t$ , which gives the best possible outcome at some future terminal time  $T$  for the investor. Assume that no borrowing or short selling is allowed, which means that we require that  $0 \leq u_t \leq 1$ . As already stated, the investor is risk averse. One way of modelling risk aversion is through expected utility theory. Introduce an increasing and concave utility function  $U(x)$ . Instead of maximizing the expected portfolio value itself, the investor wants to maximize the expected utility of the wealth at terminal time  $T$ . Assume a time horizon restricted by an initial time  $t_0$  and a terminal time  $T$ , i.e.  $t_0 < t < T$ , and assume an initial portfolio value  $V_{t_0}$ . The maximization problem can be stated as

$$I(t, x) = \max_{u_t} \mathbb{E}[U(V_T) | t_0 = t, V_{t_0} = x].$$

This constitutes an optimal control problem<sup>1</sup>, where the allocation strategy  $u_t$  is the actual control function. Define

$$\begin{aligned} \phi(t, x) &= \frac{\partial I(t, x)}{\partial t} + (\mu u_t + r(1 - u_t)) \frac{\partial I(t, x)}{\partial x} + \frac{1}{2} \sigma^2 u_t^2 x^2 \frac{\partial^2 I(t, x)}{\partial x^2} \\ &= \frac{\partial I(t, x)}{\partial t} + (r + (\mu - r)u_t) \frac{\partial I(t, x)}{\partial x} + \frac{1}{2} \sigma^2 u_t^2 x^2 \frac{\partial^2 I(t, x)}{\partial x^2}. \end{aligned} \quad (3.3)$$

The optimal solution must satisfy [15]

$$\max_{u_t} [\phi(t, x)] = 0, \quad t \in [t_0, T] \quad (3.4)$$

and  $I(T, V_T) = U(V_T)$ . (3.4) is a continuous-time version of the Bellman-Dreyfus fundamental equation of optimality. This requirement also gives the optimal solution to the problem. To find a solution that is compatible with the utility function  $U(x)$  (increasing and concave), we require that  $I_x = \partial I(t, x)/\partial x > 0$  and  $I_{xx} = \partial^2 I(t, x)/\partial x^2 < 0$ . Also, a first-order condition for finding a maximum is [15]

$$(\mu - r)I_x + \sigma^2 u_t x I_{xx} = 0,$$

---

<sup>1</sup>In this slightly simplified version of the problem, we do not consider the possibility that the portfolio value could reach zero.



which is equivalent to

$$u_t = -\frac{(\mu - r)I_x}{\sigma^2 x I_{xx}}. \quad (3.5)$$

Substituting this expression into (3.3) yields

$$\begin{aligned} & \begin{cases} \left\{ \begin{aligned} & \left[ I_t + x \left( r + (\mu - r) \left( -\frac{(\mu - r)I_x}{\sigma^2 x I_{xx}} \right) \right) \right] I_x, & t < T \\ & + \frac{1}{2} \sigma^2 \left( -\frac{(\mu - r)I_x}{\sigma^2 x I_{xx}} \right)^2 x^2 I_{xx} = 0 \\ & I(t, x) = U(x), & t = T \end{aligned} \right. \\ \Leftrightarrow & \begin{cases} I_t + rxI_x - \frac{(\mu - r)^2 I_x^2}{\sigma^2 I_{xx}} + \frac{1}{2} \frac{(\mu - r)^2 I_x^2}{\sigma^2 I_{xx}} = 0, & t < T \\ I(t, x) = U(x), & t = T \end{cases} \\ \Leftrightarrow & \begin{cases} I_t + rxI_x - \frac{(\mu - r)^2 I_x^2}{2\sigma^2 I_{xx}} = 0, & t < T \\ I(t, x) = U(x), & t = T \end{cases} \end{aligned} \quad (3.6) \end{aligned}$$

with  $I_t = \partial I(t, x) / \partial t$ .

### 3.3 Power utility

In this thesis we will model the utility of wealth  $x$  by the power function

$$U(x) = x^\gamma, \quad 0 < \gamma < 1. \quad (3.7)$$

This choice of utility function is compatible with the assumptions of the previous section, that is increasing and concave utility. This choice also allow us to find a closed form solution of the optimal control function. We will refer to  $\gamma$  as the risk aversion parameter. We see that a low value of the risk aversion parameter is associated with high aversion to risk and vice versa. To find a solution, we need to guess a solution, so we try

$$I(t, x) = f(t)x^\gamma. \quad (3.8)$$

Substituting this expression into (3.6) yields

$$\begin{aligned} & \begin{cases} f'(t)x^\gamma + rx f(t)\gamma x^{\gamma-1} - \frac{(\mu - r)^2 f^2(t)\gamma^2 x^{2(\gamma-1)}}{2\sigma^2 f(t)\gamma(\gamma - 1)x^{\gamma-2}} = 0, & t < T \\ f(t)x^\gamma = x^\gamma, & t = T \end{cases} \\ \Leftrightarrow & \begin{cases} -\frac{f'(t)}{f(t)} = r\gamma + \frac{(\mu - r)^2 \gamma}{2\sigma^2(1 - \gamma)}, & t < T \\ f(t) = 1, & t = T. \end{cases} \end{aligned}$$

Solving these equations with respect to  $f(t)$  yields

$$f(t) = \exp \left( \left( r\gamma + \frac{(\mu - r)^2 \gamma}{2\sigma^2(1 - \gamma)} \right) (T - t) \right).$$

Substituting this solution into (3.8) gives

$$I(t, x) = \exp \left( \left( r\gamma + \frac{(\mu - r)^2 \gamma}{2\sigma^2(1 - \gamma)} \right) (T - t) \right) x^\gamma. \quad (3.9)$$

Finally, we find the optimal control  $u_t^*$  by solving (3.5) with respect to (3.9),

$$u_t^* = - \frac{(\mu - r) \exp \left( \left( r\gamma + \frac{(\mu - r)^2 \gamma}{2\sigma^2(1 - \gamma)} \right) (T - t) \right) \gamma x^{\gamma-1}}{\sigma^2 x \exp \left( \left( r\gamma + \frac{(\mu - r)^2 \gamma}{2\sigma^2(1 - \gamma)} \right) (T - t) \right) \gamma(\gamma - 1) x^{\gamma-2}} = \frac{\mu - r}{\sigma^2(1 - \gamma)}, \quad (3.10)$$

which is in fact a constant independent of time. We can conclude that the optimal allocation strategy is to hold a constant fraction  $u^*$  of the wealth in the risky asset, and hence, a constant fraction  $1 - u^*$  in the risk-free asset.

The ratio (3.10) is also known as the Merton ratio. The numerator of the ratio is the difference between the risky asset drift and the risk-free rate of return. Under the assumption that no short selling is allowed, it is clear that if  $\mu - r \leq 0$  an investor will invest all of her money in the risk-free asset. For a rational and risk-averse investor, this is the obvious allocation strategy since it means the highest expected return combined with no risk at all. If  $\mu - r > 0$  the picture becomes more complex. A positive difference implies that the investor will invest at least a fraction of her wealth in the risky asset. This fraction is in part determined by the size of the difference between the risky asset drift and the risk-free rate of return, but it is also scaled by the parameter values of the denominator. The denominator is the product between the square of the volatility of the risky asset and one minus the risk aversion. Keeping all other parameters of the Merton ratio constant, we see that an increase in volatility leads to a decrease of the Merton ratio itself, and vice versa. This property of the Merton ratio is quite logical considering the fact that a risk-averse investor would be more reluctant to invest in the risky asset if the volatility increases. One minus the risk aversion can be interpreted as a scaling parameter that scales the impact of the volatility on the Merton ratio. We see that a low value of the risk aversion parameter  $\gamma$ , in relative terms, scales the impact of the volatility up, and vice versa. This is also a quite logical property since a low risk aversion parameter value is associated with high risk aversion.

# Chapter 4

## Estimation of parameters

### 4.1 Estimation of the risky asset and riskfree asset parameters

The SDE describing the dynamics of the risky asset has two parameters or constants, the drift  $\mu$  and the volatility  $\sigma$ . The differential equation describing the risk-free asset has only one parameter, the continuously compounding interest rate  $r$ . To estimate the risky asset parameters, we will use a time series consisting of daily closing index prices of the norwegian stock market index OBX to act as a proxy for stock investments. The plot of figure 4.1 shows the development

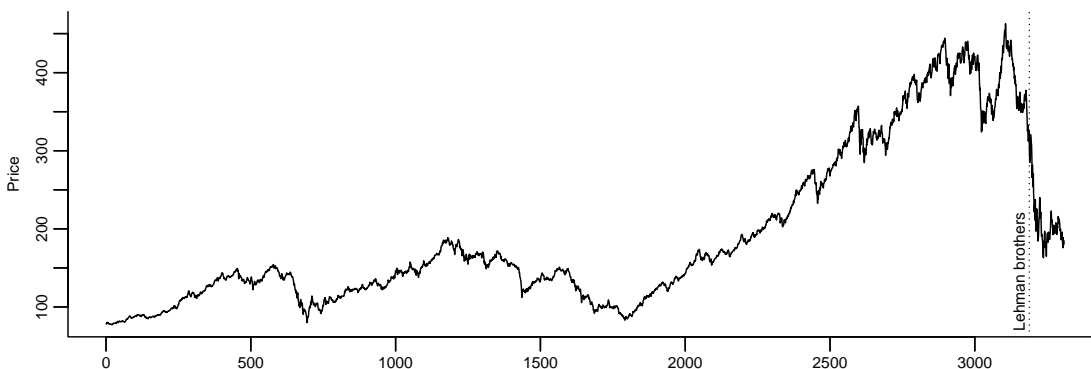


Figure 4.1: OBX index price, 3rd January 1996 - 9th March 2009.

of the OBX index price. The Lehman Brothers bankruptcy of 15th September 2008, which many count as the start of the financial crisis, is indicated by the dotted vertical line.

Due to the fact that the wealth process (5.2) describing the solution of the SDE (3.2) is a lognormal process it is natural to consider the log returns of the price

data [1] when we want to estimate  $\mu$  and  $\sigma$ . Given a time series of  $n$  daily prices  $\{s_k\}_{k=1,\dots,n}$ , the log return of the time interval  $[t_k, t_k + 1)$  is defined as

$$x_k = \log\left(\frac{s_{k+1}}{s_k}\right), \quad k = 1, \dots, n-1,$$

where log is interpreted as the natural logarithm. Using the estimation method of maximum likelihood, we can, according to Benth [1], estimate the drift  $\mu$  and the volatility  $\sigma$  by using

$$\hat{\mu} = \frac{1}{N\Delta t} \sum_{k=1}^{N-1} x_k \quad (4.1)$$

$$\hat{\sigma} = \sqrt{\frac{1}{(N-1)\Delta t} \sum_{k=1}^{N-1} (x_k - \hat{\mu})^2}. \quad (4.2)$$

This means that the risk of the risky asset is measured as the variability of the OBX log returns. Using the convention of 252 trading days in one year, to estimate the annual drift and volatility we must choose  $\Delta t = 1/252$  since the log returns are sampled on a daily basis.

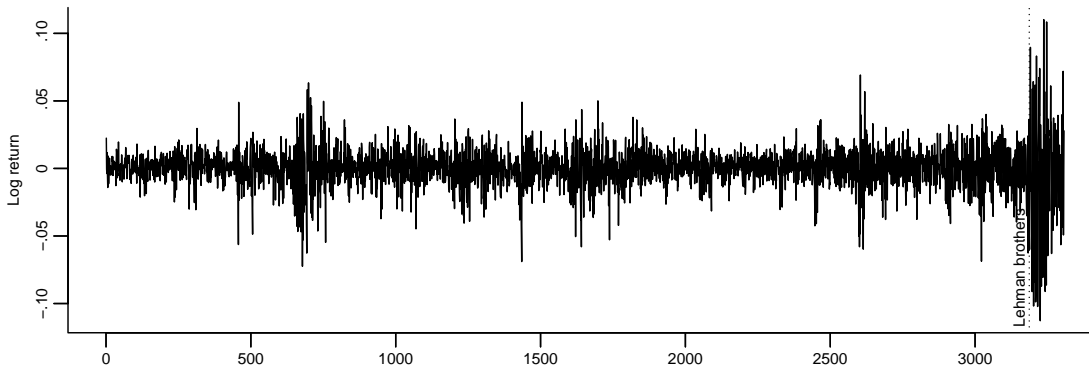
To estimate the continuously compounding interest rate we will use historical data of the effective annual interest rate of norwegian twelve month treasury bills. More specifically, the treasury bill time series consists of daily recordings of the syntetic annual interest rate. For easier comparison with the OBX log returns, given a time series of  $M$  annual treasury bill interest rates  $\{b_k\}_{k=1,\dots,M}$  and  $\Delta t = 1/252$ , the daily log returns can be calculated by the transformation

$$y_k = \Delta t \log(1 + b_k), \quad k = 1, \dots, M.$$

Analogously to the estimation of the risky asset drift, the continuously compounding interest rate  $r$  can then be estimated by using

$$\hat{r} = \frac{1}{M\Delta t} \sum_{k=1}^M y_k.$$

Initially, the OBX log return and treasury bill time series intended used for parameter estimation were time series covering the period from the start of 1996 until the end of 2010. However, by including OBX and treasury bill log return data for 2010 and most of 2009 the estimated difference between the risky asset drift and the continuously compounding interest rate becomes so large that (3.10) tells me to invest all of the wealth into the risky asset, i.e.  $u^* = 1$ , even for  $\gamma > 1$ . For the sake of an interesting simulation scenario and discussion,  $u^* = 1$  is not desirable. It turns out that estimates based on 3308 OBX log returns and 3117



**Figure 4.2:** OBX log returns, 3rd January 1996 - 9th March 2009.

treasury bill interest rates in the time period from 3rd January 1996 until 9th March 2009 do not give undesirable estimates. The estimates are summarized in table 4.1. The plot of figure 4.2 shows the development of the log returns of the OBX index. The size of the variations of the log returns reflects the amount of uncertainty in a market. We see how the uncertain economic times of the financial crisis has an impact on the variations of the log returns of the OBX index.

## 4.2 Estimation of risk aversion through VaR

The utility function (3.7) measures the investor's relative satisfaction with a given wealth  $x$ . The parameter  $\gamma$  is still to be interpreted as a risk aversion parameter. The utility function is usually assumed to be increasing and concave [14], which implicates that  $0 < \gamma < 1$ . This means that the investor becomes relatively less satisfied with increasingly bigger wealth, i.e. the investor is risk averse. For example, a low risk aversion parameter value would indicate a high aversion to risk.

To estimate the risk aversion parameter we will in this thesis employ the method of value at risk, abbreviated VaR. VaR gives us a simple way to measure the risk of losing money [8]. Jorion [11] gives the following definition: Value at risk is the worst loss over a target horizon such that there is a low, prespecified probability that the actual loss will be larger. In mathematical terms, by combining the definitions of Jorion and Benth, value at risk can be defined as follows:

**Definition 4.2.1** Define  $L$  as the loss, measured as a positive number, and  $\text{VaR}_{1-\alpha}$  as the value at risk at confidence level  $1 - \alpha$ . Then, value at risk is

defined as the loss, in absolute value, such that

$$P(L > \text{VaR}_{1-\alpha}) = \alpha.$$

There are different ways to measure the loss of the portfolio, for instance by looking at the actual portfolio value itself. But to achieve simplicity in the calculations we will choose the portfolio's log returns as our measure of loss. The log returns are defined as

$$X_k = \log \left( \frac{V_{k+1}}{V_k} \right). \quad (4.3)$$

Let  $x_{1-\alpha}^*$  denote the value at risk at confidence level  $1 - \alpha$ , then by definition

$$P(-X_k > x_{1-\alpha}^*) = \alpha. \quad (4.4)$$

If the dynamics of the wealth follows the SDE (3.2), it can be shown that the log returns are normally distributed with expectation  $(\mu u^* + r(1 - u^*) - .5\sigma^2 u^{*2})\delta$  and standard deviation  $\sigma u^* \sqrt{\delta}$ . With the probability distribution of the log returns known it is possible to solve (4.4) with respect to  $\gamma$ . The solution, which involves a quadratic equation, is

$$\gamma = 1 + \frac{(\mu - r) \left( \mu - r + \frac{q_\alpha \sigma}{\sqrt{\delta}} \pm \sqrt{\left( \mu - r + \frac{q_\alpha \sigma}{\sqrt{\delta}} \right)^2 + 2\sigma^2 \left( \frac{x_{1-\alpha}^*}{\delta} + r \right)} \right)}{2\sigma^2 \left( \frac{x_{1-\alpha}^*}{\delta} + r \right)}. \quad (4.5)$$

With values given for  $\mu$ ,  $\sigma$ ,  $r$ ,  $\delta$  and  $x_{1-\alpha}^*$  and with  $q_\alpha$  defined as the  $\alpha$ -quantile of the standard normal distribution, (4.5) gives us a way to estimate  $\gamma$ .

To be able to estimate  $\gamma$  we will also need to estimate the VaR. There are several different methods for estimating the VaR, but here we will use historical data as my method of estimation. Specifically, the historical data used for estimation of the VaR are the same historical log returns as were used for the estimation of the risky asset drift and volatility and the historical treasury bill rents as were used for the estimation of the risk-free rent. Given a confidence level  $1 - \alpha$ , an estimate for the VaR is simply the  $\alpha$ -quantile of the historical data. To take into account that the portfolio consists of investments both in a risky and a risk-free asset we will estimate the VaR by a weighted sum of the OBX and the treasury bill  $\alpha$ -quantiles. Choosing a conventional confidence level of .99, a time horizon of one day and multiplying the OBX and the treasury bill log return  $\alpha$ -quantiles with equal weights, that is weights equal to .5, we estimate that  $x_{.99}^* = .0252$ . The insertion of this estimate along with the other parameter estimates into (4.5) yields two solutions. Naturally, we choose to keep the solution,  $\hat{\gamma} = .5255$ ,

which is compatible with the assumption of an increasing and concave utility function. The complete set of parameter estimates required for the calculation of the optimal investment strategy  $u^*$  is summarized in table 4.1.

Parameter	Estimate
$\mu$	.0657
$\sigma$	.2537
$r$	.0449
$\gamma$	.5255

**Table 4.1:** The parameter estimates.

## 4.3 Calibration of the Heston model

### 4.3.1 Introduction

In this section we will estimate the parameters of the Heston stochastic volatility model, or in other words, calibrate the model. The parameters that need to be estimated are given in the set  $\Omega^H = \{\nu_0, \kappa, \theta, \xi, \rho\}$ . The calibration of the Heston model is not as straightforward as the calibration of the risky asset model (2.1). In fact, the calibration of stochastic volatility models can, according to some, be notoriously difficult. There are many different methods of calibration available, each with its own advantages and disadvantages. The different methods can be divided into two categories based on the underlying set of data used for the calibration. According to Javaheri [10] there are two possible sets of data that we can use for calibration: option prices or historical stock prices.

Using option prices, the goal is to find the set of parameter estimates that most accurately reproduces the volatilities that are implied by the real market prices of vanilla options. As such, the calibration problem that this approach entails, constitutes an inverse problem. According to Moodley [16] the most popular way of solving this inverse problem is to minimise the squared differences between the option prices implied by the model and the market prices over the parameter space. This method is also known as least squares estimation. For example, given a set of  $n$  call option market prices  $\{C_j(K_j, T_j)\}_{j=1, \dots, n}$  with strike  $K_j$  and maturity  $T_j$  and  $n$  model estimated call option prices  $\{\hat{C}_j(K_j, T_j)\}_{j=1, \dots, n}$  with stochastic volatility based on the Heston model, the least squares scheme could be formulated as

$$\min_{\Omega^H} \sum_{j=1}^n \left( \hat{C}_j(K_j, T_j) - C_j(K_j, T_j) \right)^2.$$

Alternatively, in conjunction with model calibration based on stock prices, there exists different estimation methods based on maximum likelihood. The basic idea with maximum likelihood estimation is to maximize the likelihood function (which is defined as a conditional joint probability function) over the model parameter set. Stated a bit differently, the goal is to find the most likely model parameter set given the stock price data.

What are the advantages and disadvantages of the two different approaches? According to Javaheri [10], the advantage of using calibration methods based on option prices is that it guarantees that the modelled option prices will match the option market prices within a certain tolerance. The disadvantage is the limited availability of option price data. With stock prices, the situation is opposite: we have no guarantee that the estimated option prices based on the model will match option market prices, but the availability of stock price data is usually plentiful. We will however not use any of these methods in this thesis.

### 4.3.2 Estimation of $\nu_0$ , $\theta$ and $\kappa$ through linear regression

For the calibration of the Heston model we will apply a simpler and more hands-on approach. As stated in subsection 2.1.2, the volatility process  $\nu_t$  is a CIR-process. The CIR-process is a popular model for modelling stochastic short term interest rates. To calibrate the CIR model, Wikipedia suggests discretizing the SDE and then to fit the discretized model to a set of short term interest rate data by using linear regression. To calibrate the Heston model, we will use a similar approach. The Euler approximation of the SDE of the volatility process of the Heston model can be expressed as

$$\nu_{k+1} = \nu_k + \kappa(\theta - \nu_k)\Delta t_k + \xi\sqrt{\nu_k}\Delta B_k^\nu. \quad (4.6)$$

This is equivalent to

$$\frac{\nu_{k+1} - \nu_k}{\sqrt{\nu_k}} = \kappa\theta\Delta t_k \frac{1}{\sqrt{\nu_k}} - \kappa\Delta t_k\sqrt{\nu_k} + \xi\epsilon_k^\nu, \quad (4.7)$$

where  $\epsilon_k \sim N(0, \Delta t_k)$ . We recognize this expression as a linear model suitable for linear regression.

Assume equidistant time increments, that is  $\Delta t_k = \delta$ . The linear model (4.7) can be reformulated as

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_i,$$

with

$$y_i = \frac{\nu_{i+1} - \nu_i}{\sqrt{\nu_i}}, \quad \beta_1 = \kappa\theta\delta, \quad x_{i1} = \frac{1}{\sqrt{\nu_i}}, \quad \beta_2 = -\kappa\delta, \quad x_{i2} = \sqrt{\nu_i}, \quad \epsilon_i = \xi\epsilon_i^\nu.$$



We can now apply the ordinary least squares estimators to find estimates for the  $\beta$ 's. From the above equations it is clear that

$$\hat{\theta} = -\frac{\hat{\beta}_1}{\hat{\beta}_2}, \quad \hat{\kappa} = -\frac{\hat{\beta}_2}{\delta}. \quad (4.8)$$

There is however a problem with this approach: we will require a data set of historical short term variances. Initially we do not have such a set of data, but given a set of historical log returns, we can construct a set of short term variances by calculating the variances over short subsections of the log return data. The basic idea is to let a narrow "window" move discretely from the beginning to the end of the log return data and to construct a variance data point each time the window moves up one notch. Given a time series of  $n$  log return data  $\{x_k\}_{k=1,\dots,n}$  and assuming a moving window of length  $l$ , a time series of short term variances can be constructed in the following fashion:

$$\begin{aligned} \nu_1 &= \frac{1}{(l-1)\Delta t} \sum_{j=1}^l (x_j - \bar{x}_1)^2, & \bar{x}_1 &= \frac{x_1 + \dots + x_l}{l} \\ \nu_2 &= \frac{1}{(l-1)\Delta t} \sum_{j=2}^{l+1} (x_j - \bar{x}_2)^2, & \bar{x}_2 &= \frac{x_2 + \dots + x_{l+1}}{l} \\ & \vdots & & \\ \nu_{n-l+1} &= \frac{1}{(l-1)\Delta t} \sum_{j=n-l+1}^n (x_j - \bar{x}_{n-l+1})^2, & \bar{x}_{n-l+1} &= \frac{x_{n-l+1} + \dots + x_n}{l}. \end{aligned}$$

We see that the moving window estimation method results in a new time series of  $n - l + 1$  short term variances. This way of constructing a new time series of short-term variances is quite simple and straightforward. However, it is not clear what the optimal choice of the window length  $l$  is. Different choices of  $l$  will yield somewhat different variance time series and as a consequence, different parameter estimates. We will get back to this problem when we start the actual parameter estimation.

In addition we need to estimate the initial volatility data point  $\nu_0$ , which is required in connection with simulation of the volatility process of the Heston model. There are at least two possible solutions to this problem. One solution is to use the estimated variance of the first window of the moving window estimation process. A problem with this approach is that the estimate we obtain, could turn out to be quite a long distance from the estimate of the long term mean  $\theta$ . Since the volatility process of the Heston model is a mean reverting process, this could lead to undesirable initial behaviour of a discretized simulation of the volatility process. A better solution is based on the fact that a CIR-process has a stationary

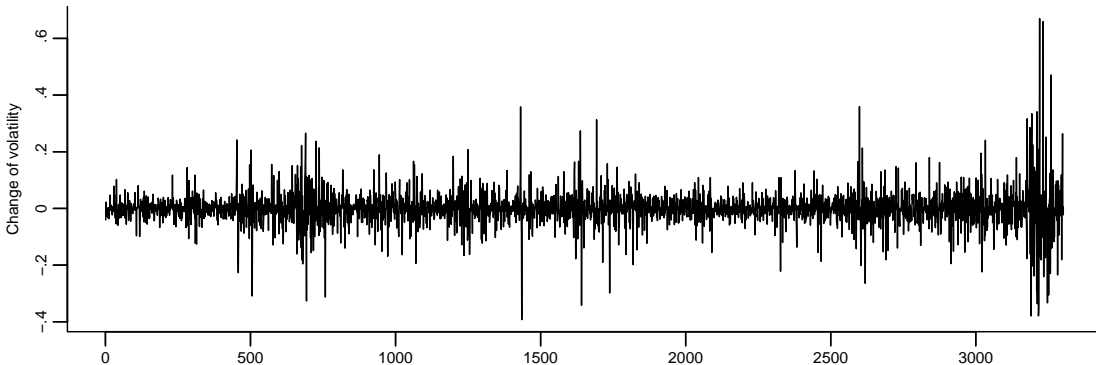
distribution. The stationary distribution of the volatility process can be shown to be a gamma distribution with shape parameter  $2\kappa\theta/\xi^2$  and scale parameter  $\xi^2/2\kappa$  [2]. This implies an expected value of  $\theta$ , which is the long-term mean of the variance process, as could be expected. As stated in subsection 2.1.2, because of the way a CIR-process is constructed, it always has a tendency to drift towards its long-term mean. As such, an estimate of the long-term mean  $\theta$  is also a neutral estimate of the initial volatility  $\nu_0$ .

### 4.3.3 Estimation of $\xi$ and $\rho$

The parameter  $\xi$  is the so-called volatility of the volatility. Given a time series of short term volatilities, a natural estimate of  $\xi$  is simply the sample standard deviation or the volatility of this time series.

The parameter  $\rho$  determines the correlation between the Brownian motion of the risky asset and the Brownian motion of the stochastic variance. As such,  $\rho$  represents the relationship between the price change of the risky asset and the change of volatility, or in other words, the relationship between the derivatives (in the discrete sense). For parameter estimation, we will use the index price data of the OBX index. A measure of the index price changes of the OBX index are the log returns, and a measure of the changes of the variance time series are the first order differences of the series. An estimate of  $\rho$  will be given as the correlation between the log returns and the first order differences.

Regarding the correlation between risky asset price change and volatility change, what can we expect? The plot of figure 4.3 shows the 1st order differences of the 5-



**Figure 4.3:** 1st order differences of annualized 5-day volatilities.

day volatilities of the OBX log returns. If we compare this plot with the OBX log returns of figure 4.2, it becomes clear that there is a positive correlation between the absolute sizes of change. If there is a correlation between the directions of

change, is however not clear. Research suggests that in most of the industrialized countries, the relationship between stock price returns and volatility is weak [13].

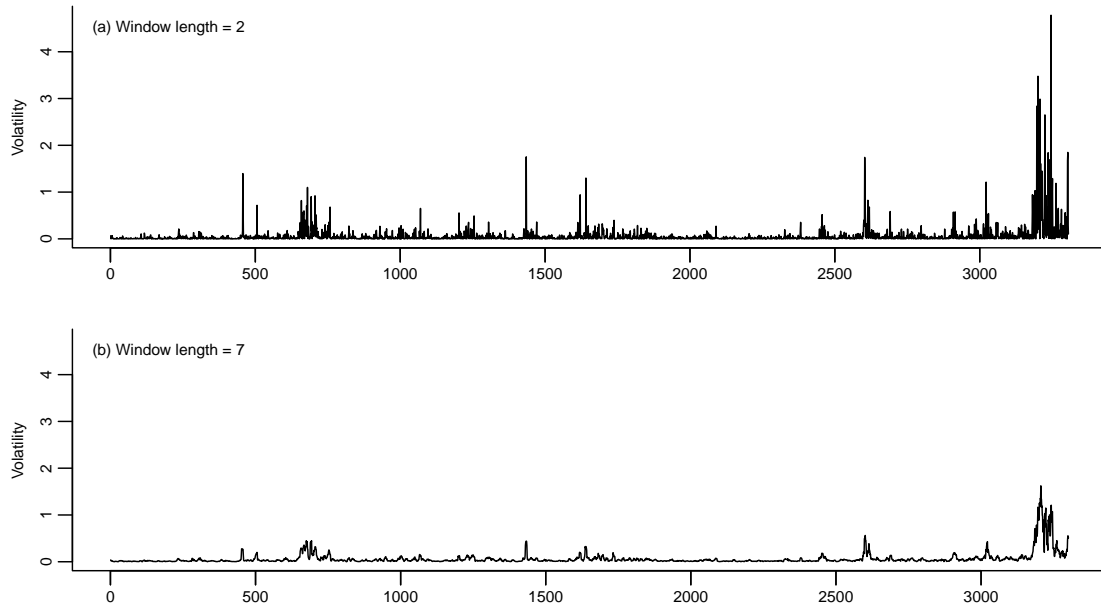
#### 4.3.4 Doing the calibration

The Euler approximation 4.6 of the volatility process is also the model that we will use for simulating the stochastic volatility of simulation model IV in the next chapter. What is the right choice of window length? The author of this thesis did unfortunately not succeed in finding any articles or other sources that address this problem. As a consequence we need to make an uneducated a priori choice of window length and five seems like a conservative choice. Other choices of window length are however available. A small range of window lengths along with the corresponding parameter estimates are given in table 4.2.

Window length	Parameter estimate				
	$\nu_0$	$\kappa$	$\theta$	$\xi$	$\rho$
2	$6.3212 \times 10^{-2}$	1377.4886	$6.3212 \times 10^{-2}$	.2292	$3.7391 \times 10^{-2}$
3	$6.4767 \times 10^{-2}$	844.6233	$6.4767 \times 10^{-2}$	.1165	$1.9363 \times 10^{-2}$
4	$6.6105 \times 10^{-2}$	599.2981	$6.6105 \times 10^{-2}$	.0775	$12.9317 \times 10^{-2}$
5	$6.7456 \times 10^{-2}$	320.1192	$6.7456 \times 10^{-2}$	.0590	$2.6706 \times 10^{-2}$
6	$6.8752 \times 10^{-2}$	214.3306	$6.8752 \times 10^{-2}$	.0511	$4.2394 \times 10^{-2}$
7	$6.9074 \times 10^{-2}$	170.0703	$6.9074 \times 10^{-2}$	.0409	$7.7981 \times 10^{-2}$

**Table 4.2:** Results of the calibration of the Heston model.

Table 4.2 summarizes the results of the calibration of the Heston model. We observe that the estimates of the parameters  $\nu_0$ ,  $\theta$  and  $\rho$  are not very sensitive to the choice of window length. The estimates for  $\kappa$  and  $\xi$  are on the other hand, very sensitive. In other words, there is a clear relation between choice of window length and the intensity of the mean reversion tendency and the volatility of the volatility. Short window lengths are associated with high estimates of  $\kappa$  and  $\xi$ . As a direct consequence of the way that the SDE (2.4) of the volatility of the Heston model is defined, higher estimates of  $\kappa$  will result in a more volatile behaviour of the volatility process  $\nu_t$  itself, since the tendency to revert towards the mean  $\theta$  will be stronger. As for the volatility of the volatility  $\xi$ , higher estimates of this parameter will obviously result in a more volatile process. These facts along with the plots of figure 4.4 explain why there is a negative correlation between window length and the estimates of  $\kappa$  and  $\xi$ . The plots of figure 4.4 show the estimated short term volatilities as a result of (a) window length equal to one, and (b) window length equal to seven. It is clear that the short term volatilities that results from a choice of window length equal to two are more spiked and volatile,



**Figure 4.4:** Short-term volatilities as a result of (a) window length equal to two, and (b) window length equal to seven.

whereas the short term volatilities that results from a choice of window length equal to seven are more smoothed out and less volatile. We observe how these features of the choices of window length are reflected in the parameter estimates of table 4.2.

Note that in relation with simulation model IV in the next section, we will simulate the stochastic volatility process using the same Euler approximation (4.6) of the SDE of the volatility as was used to create the linear regression model (4.7) of this section. The Euler approximation (4.6) is dependent on the size of the time increment  $\Delta t_k = \delta$ , which in turn implies that the linear regression model and the estimator of  $\kappa$  (4.8) also are time dependent. The estimate of  $\kappa$  needs to be scaled according to the size of the time increment. As stated earlier, we measure time in years. In the simulations, the variables will be updated hourly. Assuming 252 trading days in one year, hourly updates imply  $\delta = 1/6048$ . So, the estimates of  $\kappa$  of table 4.2 need to be interpreted in light of the size of the equidistant time increment.

# Chapter 5

## Simulation

### 5.1 Introduction

As mentioned in the introduction chapter (chapter 1), the goal of this thesis is to simulate the development of the value of a portfolio with two investment options, namely a risky asset and a risk-free asset. As already stated, the optimal strategy is for the portfolio manager or the investor to keep a constant fraction of her wealth in the risky asset and consequently a constant fraction in the risk-free asset. In Merton's portfolio problem, the investor is allowed to rebalance the portfolio continuously in time. The question is, how will this strategy perform in a more realistic, discrete time scenario?

### 5.2 Basic simulation model

#### 5.2.1 Introduction

In this section we will consider the most basic portfolio model, that is a portfolio model with constant parameters and no transaction costs. This means that we assume that the dynamics of the value of the risk-free asset follows the deterministic differential equation (3.1) and that the dynamics of the value of the risky asset follows the SDE (2.1). As shown in chapter 3, by assuming these dynamics for the risky and risk-free asset, we obtain an SDE for the portfolio value given by equation (3.2), where  $u_t$  is the control function at time  $t$ . The control function is the actual trading strategy or allocation strategy, that is, at time  $t$ , the investor must allocate a fraction  $u_t$  of the total wealth  $V_t$  in the risky asset and  $1 - u_t$  in the risk-free asset. The optimal strategy, which we will use, is to hold a constant

fraction  $u^*$  of the wealth in the risky asset, that is we assume that  $u_t = u^*$ . The dynamics of the value of the optimal portfolio is then given by

$$dV_t = (\mu u^* + r(1 - u^*))V_t dt + \sigma u^* V_t dB_t. \quad (5.1)$$

It can be shown that the solution of this SDE is

$$V_t = V_0 \exp \left( \left( \mu u^* + r(1 - u^*) - \frac{1}{2} \sigma^2 u^{*2} \right) t + \sigma u^* B_t \right). \quad (5.2)$$

This is the exact solution of the portfolio value and we will refer to  $V_t$  as the theoretical portfolio value at time  $t$ . The theoretical portfolio value will serve as a baseline for comparison.

The time domain in which we want to simulate the development of the portfolio value, is constrained by an initial time  $t_0 = 0$  and a terminal time  $t_n = T$ . Let

$$0 = t_0 < t_1 < t_2 < \dots < t_n = T \quad (5.3)$$

be the time discretization of this time domain and let  $\mathcal{T} = \{t_0, t_1, \dots, t_n\}$  denote the complete set of time points within the time interval. The time increments are defined as  $\Delta t_k = t_{k+1} - t_k$ . We will assume equidistant discretization times, i.e.  $\Delta t_k = \delta$ . The Euler-Maruyama approximation of the SDE (5.1) is defined as

$$V_{k+1} = V_k + (\mu u^* + r(1 - u^*))V_k \delta + \sigma u^* V_k \Delta B_k.$$

Observing that  $V_k = u^* V_k + (1 - u^*) V_k$ , the approximation can be rewritten as

$$V_{k+1} = \underbrace{u^* V_k}_{(i)} \underbrace{(1 + \mu \delta + \sigma \Delta B_k)}_{(ii)} + \underbrace{(1 - u^*) V_k}_{(iii)} \underbrace{(1 + r \delta)}_{(iv)}.$$

We recognize  $(i)$  as the value of the risky asset investment at time  $t_k$  and  $(ii)$  as one plus the return on the risky asset between time  $t_k$  and  $t_{k+1}$ . Likewise, we recognize  $(iii)$  as the value of the risk-free asset investment at time  $t_k$  and  $(iv)$  as one plus the return on the risk-free asset. This approximation will serve as a template for the simulation models. The approximation describes a recursive method of simulation. It is the correct method for simulating the portfolio value at discrete time points, because the portfolio value at each time point is the wealth at the preceding time point plus the return from the amount invested in the risk-free asset plus the return from the amount invested in the risky asset.

The amount invested in the risk-free and the risky asset will follow the optimal trading strategy, but the rebalancings of the portfolio will not necessarily happen at each and every time point. In the simulations one important task is to compare different rebalancing strategies, such as daily rebalancings, monthly rebalancings et cetera. Given a time interval and a set of time points according to a discretization of the time interval, we will achieve this by rebalancing the portfolio at time

points according to a subset of the time points. Because of this the simulated portfolio value will be calculated by using a somewhat modified Euler-Maruyama approximation scheme, which will be formulated in the next section.

To make a notational distinction between theoretical quantities and simulated quantities where it is necessary, simulated quantities will be indicated with a tilde. For example, the simulated portfolio value at time  $t_k$  will be given as  $\tilde{V}_k$ . The set of rebalancing time points is given by  $\mathcal{T}^{\text{reb}} = \{t_0, t_\epsilon, t_{2\epsilon}, \dots, t_n\}$  which constitutes a subset of the complete set of time points, i.e.  $\mathcal{T}^{\text{reb}} \subseteq \mathcal{T}$ . The positive integer  $\epsilon$  denotes the distance between rebalancing time indices and for simplicity we will assume that  $\epsilon$  is a divisor of  $n$ . Assume also that the last rebalancing time point relative to the time point in which we want to simulate the wealth is given by  $t_{k^*}$ . The total portfolio value can be seen as a sum consisting of two values: the value of the investment in the risky asset and the value of the investment in the risk free asset. The value of the risky asset investment at time  $t_k$  is denoted by  $\tilde{V}_k^S$ , the value of the risk free asset investment is denoted by  $\tilde{V}_k^R$  and the total portfolio value is denoted by  $\tilde{V}_k$ . In addition,  $Q_k$  denotes the amount that needs to be subtracted from the risky asset investment and added to the risk free asset investment, that is the transaction quantity, at each rebalancing time point to rebalance the portfolio in accordance with the optimal strategy. This implies that  $Q_k$  also can be negative. A negative transaction just means that the risk-free investment needs to be reduced and the risky investment increased, to put the portfolio in a state of balance according to the optimal strategy.

### 5.2.2 Simulation model I

We will refer to the basic and initial simulation model as simulation model I. The model is defined by the following set of equations:

**Simulation model I**

Transaction costs: none

Volatility: constant

$$\begin{aligned}\tilde{V}_k'^S &= u^* \tilde{V}_{k^*} \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) \\ \tilde{V}_k'^R &= (1 - u^*) \tilde{V}_{k^*} (1 + r\delta)^{k-k^*} \\ Q_k &= (1 - u^*) \tilde{V}_k'^S - u^* \tilde{V}_k'^R \\ \tilde{V}_k^S &= \begin{cases} \tilde{V}_k'^S - Q_k, & t_k \in \mathcal{T}^{\text{reb}} \\ \tilde{V}_k'^S, & \text{otherwise} \end{cases} \\ \tilde{V}_k^R &= \begin{cases} \tilde{V}_k'^R + Q_k, & t_k \in \mathcal{T}^{\text{reb}} \\ \tilde{V}_k'^R, & \text{otherwise} \end{cases} \\ \tilde{V}_k &= \tilde{V}_k^S + \tilde{V}_k^R.\end{aligned}$$

$\tilde{V}_k'^S$  represents the value of the risky asset investment at time  $t_k$ . At rebalancing time points,  $\tilde{V}_k'^S$  will represent the value of the risky asset investment before the portfolio is rebalanced. It is defined as the value of the risky asset investment at the preceding rebalancing time point  $t_{k^*}$  times the product of one plus the return on the risky asset of each time interval since the preceding rebalancing time point, that is the value after compounding. The value of the risk-free investment  $\tilde{V}_k'^R$  at time  $t_k$  is calculated using the same rationale. What about  $Q_k$ ? Assume that  $t_k$  is a rebalancing time point. For the portfolio to become rebalanced according to  $u^*$ , it is required that  $\tilde{V}_k^S = u^* \tilde{V}_k = u^* (\tilde{V}_k'^S + \tilde{V}_k'^R)$ . From this it is clear that

$$\begin{aligned}Q_k &= \tilde{V}_k'^S - u^* (\tilde{V}_k'^S + \tilde{V}_k'^R) = (1 - u^*) \tilde{V}_k'^S - u^* \tilde{V}_k'^R \\ &= u^* (1 - u^*) \tilde{V}_{k^*} \left( \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) - (1 + r\delta)^{k-k^*} \right) \\ &= u^* (1 - u^*) \tilde{V}_{k^*} \left( \left( \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) - 1 \right) - ((1 + r\delta)^{k-k^*} - 1) \right).\end{aligned}\tag{5.4}$$

Notice that the sign of  $Q_k$  is only determined by the difference between the returns on each asset investment since the last rebalancing time point  $t_{k^*}$ , which reflects the fact that the balance of the portfolio is preserved as long as the returns are equal. Hence, a difference in returns at a rebalancing time point requires the portfolio to be rebalanced.



Since  $Q_k$  is both added and subtracted at the same time at each rebalancing time point, it doesn't affect the total value of the portfolio. For the sake of the simulation of the portfolio value it is not even necessary to calculate  $Q_k$  because we know that  $\tilde{V}_k^S = u^* \tilde{V}_k$  and that  $\tilde{V}_k^R = (1 - u^*) \tilde{V}_k$ . What this means is that the simulation model can be stated in a more compact way:

$$\tilde{V}_k = u^* \tilde{V}_{k^*} \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) + (1 - u^*) \tilde{V}_{k^*} (1 + r\delta)^{k-k^*}. \quad (5.5)$$

This compact restatement of the simulation model is more ideal as a basis for implementation of fast simulation routines in R.

To illustrate how the simulation model works we will look at an example.

**Example 5.2.1** Assume that the portfolio is rebalanced at every 3rd time point, which implies  $\epsilon = 3$ . The subset of rebalancing time points is as a result given as  $\mathcal{T}^{\text{reb}} = \{t_0, t_3, t_6, \dots, t_n\}$ . Also assume that  $\tilde{V}_0 = V_0$  and that  $y_k = \mu\delta + \sigma\Delta B_k$  which is the return on the amount invested in the risky asset between time points  $t_k$  and  $t_{k+1}$ . Then according to (5.5) we have that

$$\begin{aligned} \tilde{V}_1 &= u^* V_0 (1 + y_0) + (1 - u^*) V_0 (1 + r\delta) \\ \tilde{V}_2 &= u^* V_0 (1 + y_0) (1 + y_1) + (1 - u^*) V_0 (1 + r\delta)^2 \\ \begin{cases} Q_3 &= (1 - u^*) u^* V_0 (1 + y_0) (1 + y_1) (1 + y_2) - u^* (1 - u^*) V_0 (1 + r\delta)^3 \\ \tilde{V}_3 &= u^* V_0 (1 + y_0) (1 + y_1) (1 + y_2) + (1 - u^*) V_0 (1 + r\delta)^3 \end{cases} \\ \tilde{V}_4 &= u^* \tilde{V}_3 (1 + y_3) + (1 - u^*) \tilde{V}_3 (1 + r\delta) \\ &\vdots \\ \begin{cases} Q_n &= (1 - u^*) u^* V_{n-3} (1 + y_{n-3}) (1 + y_{n-3}) (1 + y_{n-3}) - u^* (1 - u^*) V_{n-3} (1 + r\delta)^3 \\ \tilde{V}_n &= u^* \tilde{V}_{n-3} (1 + y_{n-3}) (1 + y_{n-3}) (1 + y_{n-3}) + (1 - u^*) \tilde{V}_{n-3} (1 + r\delta)^3. \end{cases} \end{aligned}$$

### 5.2.3 Loss of utility

The portfolio manager's utility of the wealth is given by a utility function (3.7), which is a utility function from the family of power functions. To measure the loss of utility at terminal time  $T$ , we simply calculate the difference between the utility of the theoretical wealth  $U(V_T)$  with the utility of the simulated wealth  $U(\tilde{V}_T)$ , that is, the measure of the loss of utility will be given by

$$U(V_T) - U(\tilde{V}_T). \quad (5.6)$$

### 5.2.4 Simulation test run

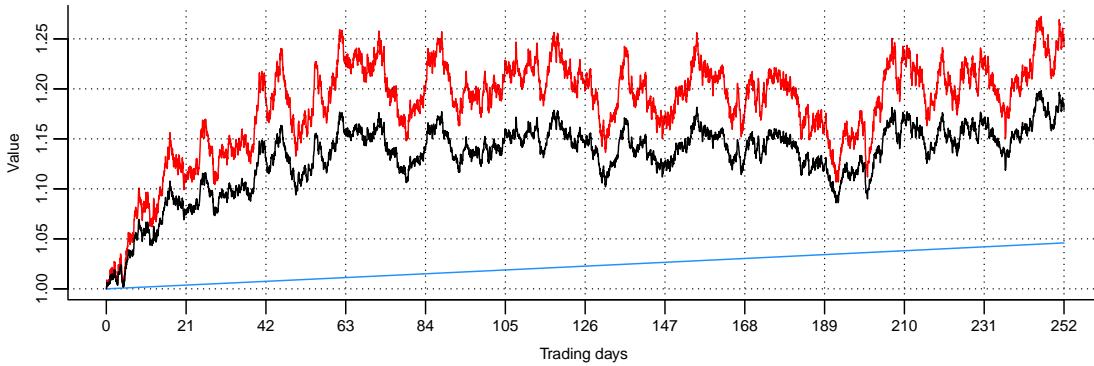
Parameter	Value	Parameter	Value
$V_0$	1	$c_B$	24
$\mu$	.0657	$c_P$	12/252
$\sigma$	.2537	$n$	6048
$r$	.0449	$\delta$	1/6048
$\gamma$	.5255		

**Table 5.1:** Example of a complete set of simulation input parameter values.

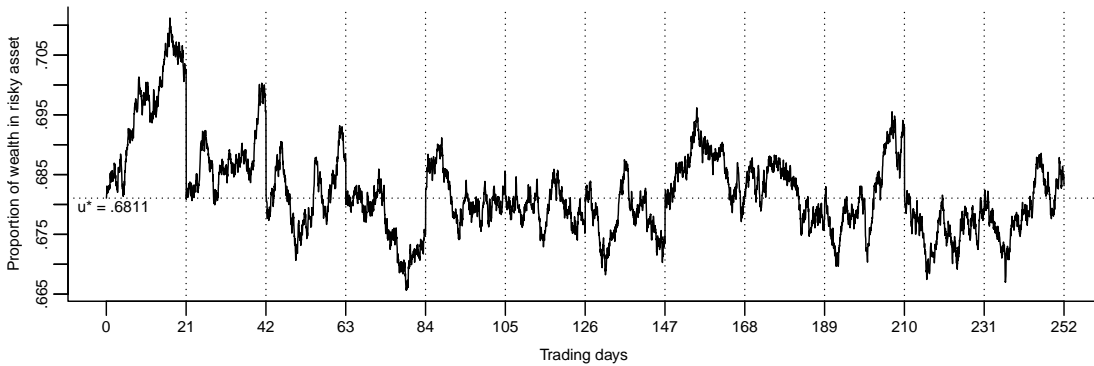
All of the simulations in this thesis were implemented and executed in the statistical language R. Initially, to get a feel for the behaviour of the simulations, we will implement a single simulation test run. The simulation function has several different input parameters: The parameters in "Merton's portfolio problem", that is, the initial wealth  $V_0$ , the continuously compounding interest rate  $r$  of the risk-free asset, the drift  $\mu$  and volatility  $\sigma$  of the risky asset, the risk aversion parameter  $\gamma$  and the optimal investment strategy  $u^*$ . As for the specific choices of these parameter values, these are of course the parameter estimates calculated in chapter 4. These estimates yield  $u^* = .6811$ .

For the simulations we also need to define additional parameters. These parameters are  $n$ , which denotes the total number of time points in one year,  $\delta$ , which denotes the size of the equidistant time increments,  $c_B$ , which denotes the number of daily changes of the risky asset, that is, the number of daily increments of the simulated Brownian motion underlying the stochastic dynamics of the risky asset, and  $c_P$ , which denotes the number of daily portfolio rebalancings the portfolio manager may do. This implies  $\epsilon = c_B/c_P$ . These are basically simulation specific parameters. Concerning the choices of these parameter values, the time will be measured in years and we will follow the conventional assumption of 252 trading days in one year. This means that one year will be discretized into  $n = 252c_B$  time points and that  $t_n = T = 1$ , which implies  $\delta = 1/n$ . Assume that  $c_B = 24$ , which means that the risky asset will change value at an hourly basis. A consequence of this choice is  $n = 6048$  and  $\delta = 1/6048$ . We will in the test run assume monthly rebalancings, that is a total 12 rebalancings in one year, which implies  $c_P = 12/252$ . The complete set of parameter values required for a simulation run, are given in table 5.1.

Figure 5.1 shows the results of a single simulation run with parameter values according to table 5.1. The vertical dotted lines indicate the rebalancing time points related to the number of trading days. Monthly rebalancings imply that the portfolio is rebalanced every 21st day. The plot of subfigure (a) shows the development of the risky asset value (red), the risk-free asset value (blue) and



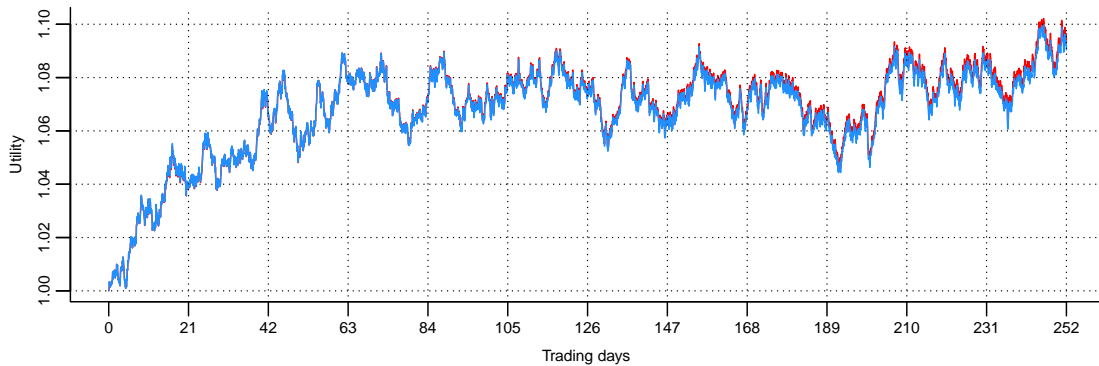
(a) Development of risky asset value (red), risk-free asset value (blue) and portfolio value (black).



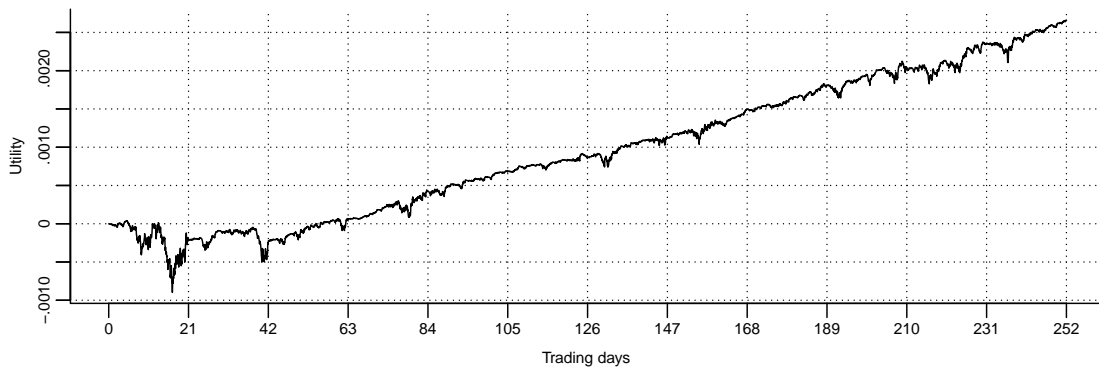
(b) Proportion of the wealth invested in the risky asset.

**Figure 5.1:** Results of the test run.

the simulated portfolio value (black). It is clear that for this particular simulation run, the development of the risky asset is far superior compared to the development of the risk-free asset. This is reflected in the plot of subfigure (b), which shows the size of the proportion of the wealth invested in the risky asset. We see that just before the first rebalancing of the portfolio at trading day 21, the strong development of the risky asset causes the proportion of the risky asset investment to deviate considerably from the optimal proportion  $u^*$ . We also see how the portfolio is adjusted at each rebalancing time point, to match the optimal allocation proportion. Figure 5.2 shows plots concerning the utility of the wealth of the investor. In subfigure (a) the utility of the simulated wealth (5.5) is plotted in blue on top of the utility of the theoretical wealth (5.2), which is plotted in red. As the plots of subfigure (a) shows, the value of the simulated wealth follows the theoretical wealth very closely, but clearly, there are small differences. These differences are magnified in the in subfigure (b), which shows the difference in utility at each time point. We observe that for this specific simulation run, the difference is relatively small but that it is increasing with time. In



(a) The utility of the theoretical (red) and the simulated wealth (blue).



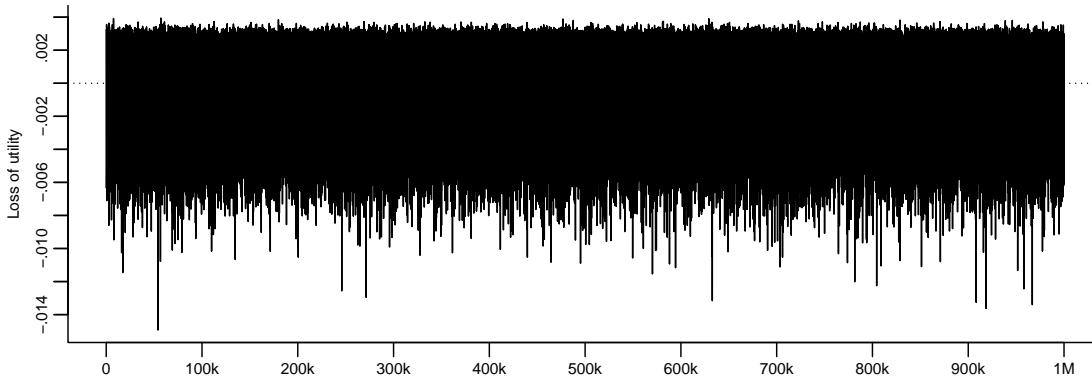
(b) The difference in utility.

**Figure 5.2:** One single simulation run over 252 trading days.

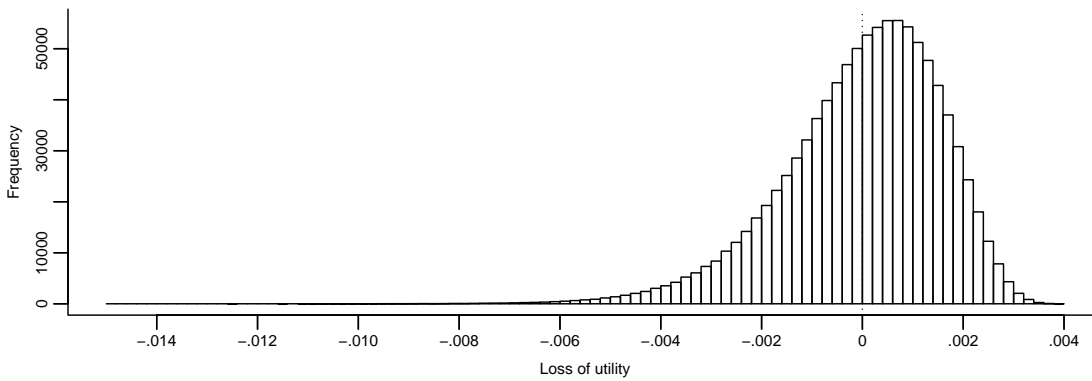
some time intervals, there also seems to be a correlation not only with time, but also with the utility of both the theoretical and the simulated wealth. However, one simulation is of course not sufficient to draw any serious conclusions about the loss of utility. To be able to do that, it is a good idea to consider the sample mean of the simulated loss of utilities, which is exactly what we will do in the next section.

## 5.2.5 Mean loss of utility

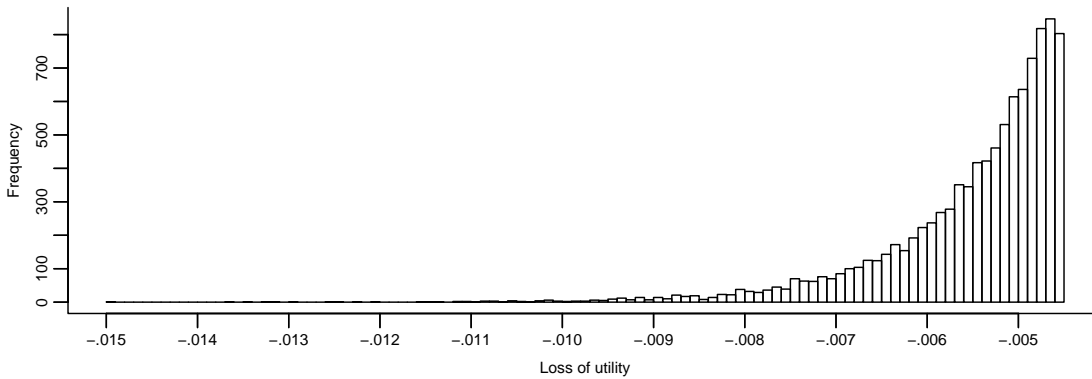
Figure 5.3 shows the results after calculating the terminal losses of utility (5.6) of one million simulation runs with parameter values according to table 5.1. The plot of figure 5.3 suggests that the mean loss of utility might be slightly less than zero due to the fact that many of the negative losses are larger in absolute value compared to the positive losses. However, the histogram of figure 5.4 (a) shows that the distribution of losses of utilities is skewed to the left with a global maximum in the positive region and with the sample mean close to zero. Also, the left



**Figure 5.3:** One million simulation runs.



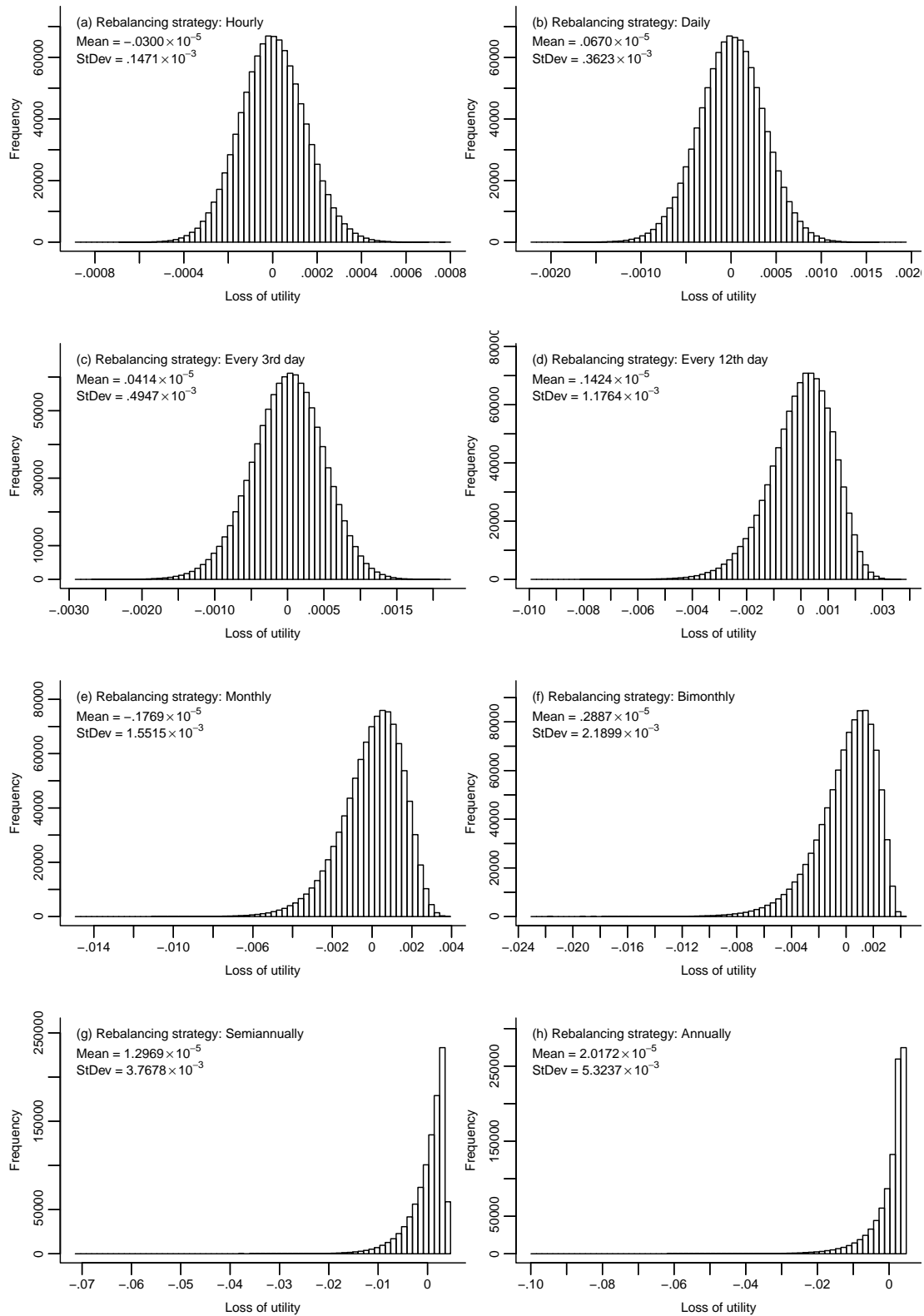
**(a)** The distribution of the losses of utility.



**(b)** The distribution of the lower one percent of the losses of utility.

**Figure 5.4:** Distributions of one million simulation runs.

tail is extremely long and narrow. This left tail behaviour is magnified in figure 5.4 (b) and shows that in relative magnitude, some of the negative losses are very large compared to the main bulk of losses, but that they are extremely rare. The histograms of figure 5.5 show how this left tail behaviour is related to the choice



**Figure 5.5:** The distributions of the losses of utility of the different rebalancing strategies.

of rebalancing strategy. The distribution of the losses of utility of the hourly-rebalancing strategy is similar to a normal distribution, whereas the distribution of the losses of utility of the annual-rebalancing strategy is extremely skewed to the left. As for the distributions of the intermediate rebalancing strategies, they describe an evolution from gaussian symmetry towards negative skewness. Remember that the loss of utility is the utility of the theoretical portfolio value minus the utility of the simulated portfolio value. This means that a negative loss of utility is equivalent to a gain of utility. We observe that on rare occasions, the gain of utility for the annual-rebalancing strategy can be quite large. .1 is a large gain of utility considering that the initial utility is equal to one. On the other, the maximum loss of utility is also larger for the annual-strategy:  $4.7914 \times 10^{-3}$  versus  $1.9437 \times 10^{-3}$  for the daily-strategy. On average, the daily-strategy seems to be a little bit better.

The plots of figure 5.6 show how the simulated losses of utility sample means develop as the number of simulations increases. The outer grey lines mark the lower and upper limits of a 95% confidence interval of the estimated mean, calculated under the assumption of a normally distributed mean in accordance with the central limit theorem. We observe that for all the different rebalancing strategies, the mean loss of utility seems to converge towards a value very close to zero. Considering that the strategy of hourly rebalancings is in fact the direct Euler-approximation of the portfolio value, it is not surprising that the mean loss of utility for this specific strategy is close to zero. The mean loss of utility is however very small for all the rebalancing strategies and for all practical purposes approximately equal to zero. With a significance level of 5%, the mean losses of utility for the semiannual and the annual strategy are significantly different from zero, but they are still extremely small. This result suggests that by the law of large numbers, the sample mean utilities of the simulated portfolio values will converge toward the true expected utility of the theoretical portfolio value, that is,

$$E[U(V_t)] = V_0^\gamma \exp\left(\left(\mu u^* + r(1 - u^*) - \frac{1}{2}\sigma^2 u^{*2}\right)t\right)^\gamma c$$

with

$$c = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(\sigma u^* \sqrt{t}x)^\gamma \exp\left(-\frac{1}{2}x^2\right) dx.$$

The plot of figure 5.7 and table 5.2 confirms the picture we have seen so far. The plot also provides more detail into the relationship between rebalancing strategy and the measuring uncertainty of the loss of utility. The plot shows the mean losses of utility as the curve in the middle with accompanying confidence limits of 95% confidence intervals. We observe that frequent rebalancings are associated with narrow confidence intervals and that strategies with just a few

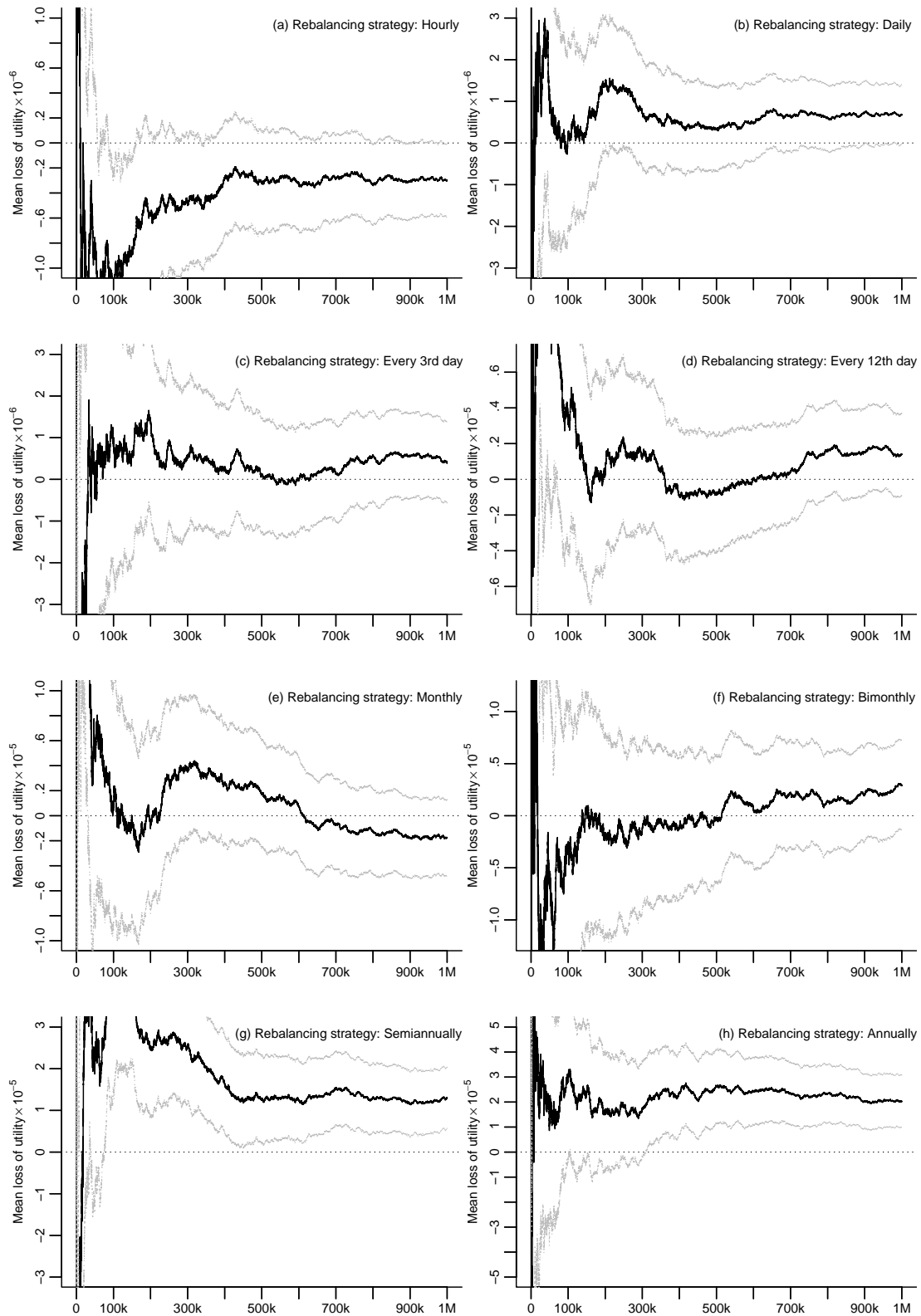
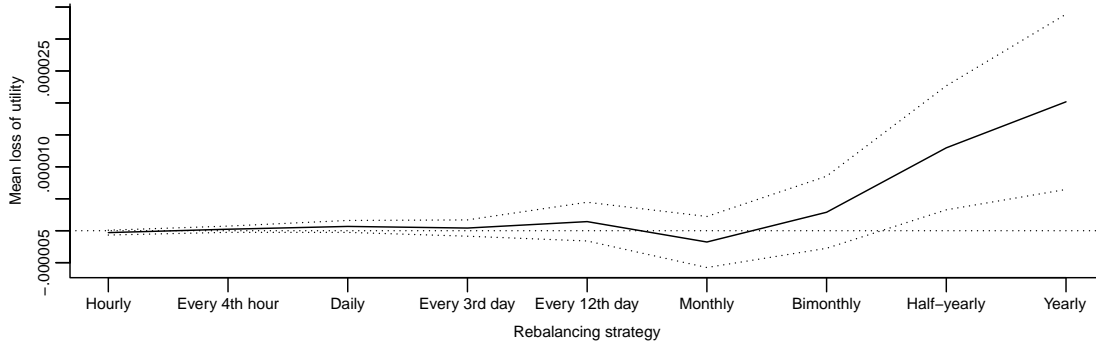


Figure 5.6: The mean losses of utility plotted against rebalancing strategies.





**Figure 5.7:** Mean loss of utility vs rebalancing interval.

Simulation model		Sample means				StDev Loss of utility
		Term. wealth	Loss of wealth	Term. utility	Loss of utility	
Rebalancing strategy	Hourly	Th	1.0609	0	1.0277	0
		Sim	1.0609	$-.0476 \times 10^{-5}$	1.0277	$-.0300 \times 10^{-5}$
	Every 4th hour	Th	1.0609	0	1.0277	0
		Sim	1.0609	$.0539 \times 10^{-5}$	1.0277	$.0233 \times 10^{-5}$
	Daily	Th	1.0610	0	1.0278	0
		Sim	1.0610	$.1044 \times 10^{-5}$	1.0278	$.0670 \times 10^{-5}$
	Every 3rd day	Th	1.0607	0	1.0276	0
		Sim	1.0607	$.0364 \times 10^{-5}$	1.0276	$.0414 \times 10^{-5}$
	Every 12th day	Th	1.0607	0	1.0276	0
		Sim	1.0607	$-.0534 \times 10^{-5}$	1.0276	$.1424 \times 10^{-5}$
	Monthly	Th	1.0608	0	1.0277	0
		Sim	1.0608	$-.9413 \times 10^{-5}$	1.0277	$-.1769 \times 10^{-5}$
	Bimonthly	Th	1.0610	0	1.0278	0
		Sim	1.0610	$-.6428 \times 10^{-5}$	1.0278	$.2887 \times 10^{-5}$
	Seminannualy	Th	1.0606	0	1.0276	0
		Sim	1.0606	$-1.0460 \times 10^{-5}$	1.0275	$1.2969 \times 10^{-5}$
	Annually	Th	1.0607	0	1.0276	0
		Sim	1.0607	$-3.1841 \times 10^{-5}$	1.0276	$2.0172 \times 10^{-5}$

**Table 5.2:** Mean losses of utility and other related statistics.

rebalancings during one year are associated with wider confidence intervals. This is of course a rather obvious feature considering that the potential size of the difference in utility will increase as the time since the last rebalancing took place, increases, leading to potentially larger differences in utility and hence, larger variance. This tells us that although the expected utility of an investors portfolio value will lie close to the expected utility of the theoretical wealth, as concluded above, the uncertainty of this prediction will increase as the time interval between rebalancings increases. This also points to the fact that strategies of infrequent rebalancings involve higher risk to the investor. This is not surprising considering

the fact that the optimal strategy of holding a constant fraction of the wealth in the risky asset, is meant to limit the risk.

### 5.2.6 Portfolio return and Sharpe ratio

To compare the performances of the different rebalancing strategies we will employ the Sharpe ratio. As described earlier in section 2.2, the Sharpe ratio measures the excess return per unit of risk of an investment portfolio. Also, there are two versions of the Sharpe Ratio, the ex ante version and the ex post version. To compare the rebalancing strategies we must use the ex post version. The ex ante version will serve as a baseline for the ex post Sharpe ratios. For both versions, the natural benchmark is the risk free rate of return,  $r$ . It can be shown that

$$E[X_t] = \left( \mu u^* + r(1 - u^*) - \frac{1}{2} \sigma^2 u^{*2} \right) t, \quad (5.7)$$

$$\text{Var}[X_t] = \sigma^2 u^{*2} t. \quad (5.8)$$

Substituting these expressions along with  $r$  into (2.6) yields

$$SR_t^{ea} = \frac{(\mu u^* + r(1 - u^*) - \frac{1}{2} \sigma^2 u^{*2})t - r}{\sigma u^* \sqrt{t}}.$$

After one year, that is at time  $t = 1$ , we have that  $SR_1^{ea} = -4.4060 \times 10^{-3}$ , which is a negative Sharpe ratio. Does this mean that the expected return of the portfolio is less than the expected value of the risk free asset? No, not necessarily, because in this thesis we use log returns instead of arithmetic returns. If we consider the expected theoretical wealth of the portfolio at time  $t$ ,

$$E[V_t] = V_0 \exp((\mu u^* + r(1 - u^*))t), \quad (5.9)$$

we observe that the return of this quantity is  $\exp((\mu u^* + r(1 - u^*))t) - 1$ , which also is the expected arithmetic return of the portfolio. The (expected) arithmetic return of an investment in the risk-free asset is  $\exp(rt) - 1$ . Thus, maybe the difference  $(\mu u^* + r(1 - u^*))t - rt$  would have been a more natural measure of the expected excess return of the portfolio investment versus the risk free investment. Using log returns we get an extra term  $-\frac{1}{2} \sigma^2 u^{*2} t$  in the expected excess return, which gives a negative ex ante Sharpe ratio. However, the main purpose of using the Sharpe ratio in this thesis is not to compare the portfolio performance against the risk free asset, but to compare the different rebalancing strategies relative to each other. In this context a negative Sharpe ratio is not a considerable problem.

Regarding the comparison of the different rebalancing strategies, we are interested in comparing the ex post Sharpe ratios at terminal time, that is after one year. In

order to make the ex post Sharpe ratio comparable to the ex ante Sharpe ratio, we need to annualize the ex post Sharpe ratio. This is achieved by using the annualized estimators of chapter 4, that is, equation (4.1) and equation (4.2).

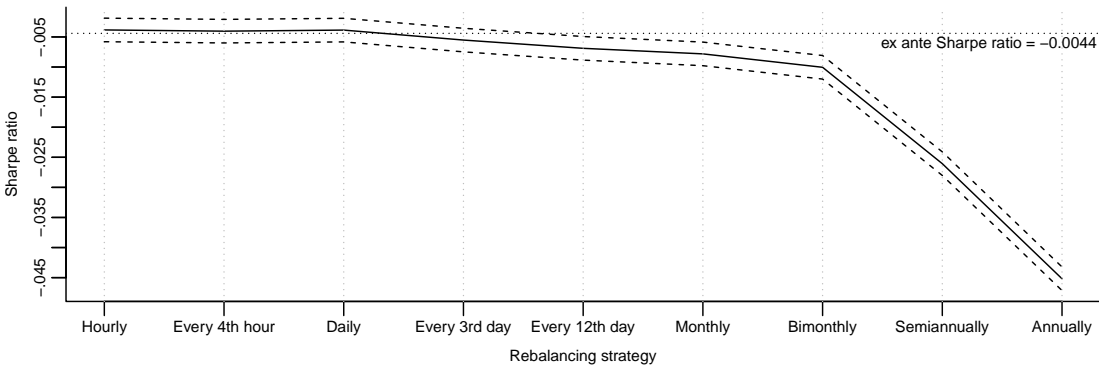
As mentioned earlier, log returns give a notational advantage over arithmetic returns. For instance, a series of log returns form a telescoping series. Assuming that  $\tilde{v}_0 = 1$ , the telescoping property of the log returns yields  $\tilde{\mu}_x = \frac{1}{n\Delta t} \log \tilde{v}_n$ . Further, the time increments are equidistant and assumed equal to  $\delta = 1/n$ . This means that  $\tilde{\mu}_x = \log \tilde{v}_n$ . As for the continuously compounding risk free return, it is for each time interval constant and approximately equal to  $r/n$ . As a consequence of this, the annualized risk free return is equal to  $r$ , as it should be. With the sample mean of the log returns equal to  $\log \tilde{v}_n/n$ , we have that the annualized sample standard deviation is formulated as

$$\hat{\sigma}_x = \sqrt{\frac{n}{n-1} \sum_{k=0}^{n-1} \left( x_k - \frac{\log \tilde{v}_n}{n} \right)^2}.$$

Given a time discretization (5.3), a set of log returns  $\{x_k\}_{k=1,\dots,n}$  and a set of risk free rents  $\{r_k\}_{k=1,\dots,n}$ , the annualized ex post Sharpe ratio at time  $t = 1$  is defined as

$$SR_n^a = \frac{\log \tilde{v}_n - r}{\sqrt{\frac{n}{n-1} \sum_{k=0}^{n-1} \left( x_k - \frac{\log \tilde{v}_n}{n} \right)^2}}. \quad (5.10)$$

To calculate the ex post Sharpe ratios we will use the same set of data of one million simulation runs for each rebalancing strategy, as was used in estimations of the losses of utility. For each and every simulation run the ex post Sharpe ratio is calculated by using equation (5.10). To compare the strategies we can for instance look at the sample mean of the ex post Sharpe ratios of each strategy.



**Figure 5.8:** Rebalancing strategy versus ex post Sharpe ratio.

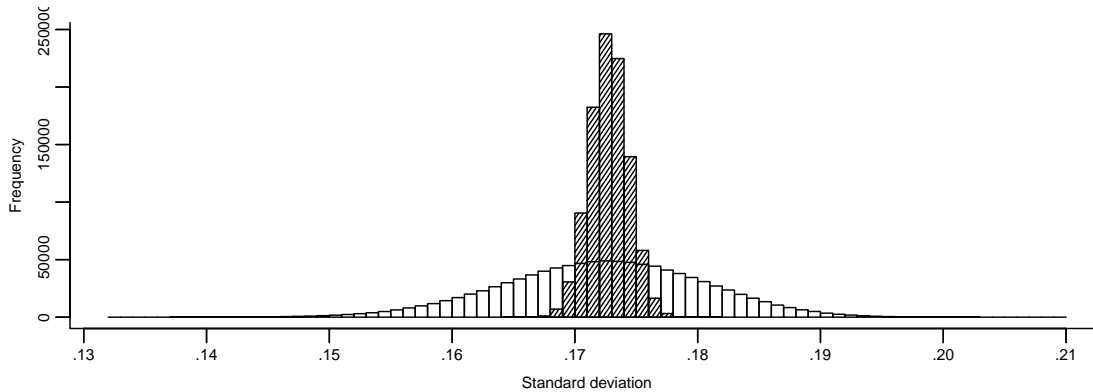
Table 5.3 summarizes the different rebalancing strategies' ex post Sharpe ratios. We observe that there is a clear positive correlation between Sharpe ratio and

Simulation model	Sample means			Vol. of vol.	Corr.	Rank		
	Terminal log return	Vol.	Sharpe ratio					
Rebalancing strategy	Hourly	Th	$4.4230 \times 10^{-2}$	.1728	$-.3880 \times 10^{-2}$	$.1570 \times 10^{-2}$	0	1
		Sim	$4.4230 \times 10^{-2}$	.1728	$-.3834 \times 10^{-2}$	$.1570 \times 10^{-2}$	-.0047	
	Every 4th hour	Th	$4.4197 \times 10^{-2}$	.1728	$-.4062 \times 10^{-2}$	$.1569 \times 10^{-2}$	-.0006	3
		Sim	$4.4197 \times 10^{-2}$	.1728	$-.4041 \times 10^{-2}$	$.1569 \times 10^{-2}$	-.0031	
	Daily	Th	$4.4252 \times 10^{-2}$	.1728	$-.3744 \times 10^{-2}$	$.1569 \times 10^{-2}$	-.0009	2
		Sim	$4.4251 \times 10^{-2}$	.1728	$-.3862 \times 10^{-2}$	$.1569 \times 10^{-2}$	.0113	
	Every 3rd day	Th	$4.3994 \times 10^{-2}$	.1728	$-.5245 \times 10^{-2}$	$.1571 \times 10^{-2}$	.0004	4
		Sim	$4.3994 \times 10^{-2}$	.1728	$-.5519 \times 10^{-2}$	$.1572 \times 10^{-2}$	.0302	
	Every 12th day	Th	$4.4037 \times 10^{-2}$	.1728	$-.5012 \times 10^{-2}$	$.1573 \times 10^{-2}$	.0019	5
		Sim	$4.4036 \times 10^{-2}$	.1728	$-.6891 \times 10^{-2}$	$.1618 \times 10^{-2}$	.2020	
	Monthly	Th	$4.4117 \times 10^{-2}$	.1728	$-.4529 \times 10^{-2}$	$.1571 \times 10^{-2}$	-.0007	6
		Sim	$4.4123 \times 10^{-2}$	.1727	$-.7816 \times 10^{-2}$	$.1705 \times 10^{-2}$	.3356	
	Bimonthly	Th	$4.4320 \times 10^{-2}$	.1728	$-.3363 \times 10^{-2}$	$.1573 \times 10^{-2}$	.0003	7
		Sim	$4.4320 \times 10^{-2}$	.1727	$-1.0044 \times 10^{-2}$	$.2064 \times 10^{-2}$	.5595	
	Semi-annually	Th	$4.3881 \times 10^{-2}$	.1728	$-.5896 \times 10^{-2}$	$.1570 \times 10^{-2}$	-.0002	8
		Sim	$4.3874 \times 10^{-2}$	.1726	$-2.6059 \times 10^{-2}$	$.4307 \times 10^{-2}$	.8045	
	Annually	Th	$4.4047 \times 10^{-2}$	.1728	$-.4920 \times 10^{-2}$	$.1572 \times 10^{-2}$	-.0018	9
		Sim	$4.4043 \times 10^{-2}$	.1722	$-4.5186 \times 10^{-2}$	$.8155 \times 10^{-2}$	.8451	

**Table 5.3:** The Sharpe ratios of the different rebalancing strategies along with other statistics.

rebalancing frequency. The rebalancing strategy which involves hourly rebalancings of the portfolio has the best Sharpe ratio by a slight margin, although both the strategies which involves rebalancings every fourth hour and rebalancings daily, perform very similarly. The annual strategy, which implies no rebalancings during one year (only allocation of the wealth according to  $u^*$  at the start of the year) has the worst performance. The plot of figure 5.8 shows the different rebalancing strategies versus Sharpe ratio. Also included are 95% confidence intervals, which show that the first four rebalancing strategies are not significantly different from the ex ante Sharpe ratio and that the strategies of semiannual and annual rebalancing strategies perform relatively much worse than the other strategies.

Now, why do the strategies that involve frequent rebalancings of the portfolio perform better according to the Sharpe ratio? Analysing the table we observe that the differences between the mean terminal log returns are small, which imply that the excess returns also are. Even the sample means of the estimated volatilities that the different strategies yield are very similar. The column named "Vol. of. vol", an abbreviation for the volatility of the volatilities, shows the sample standard deviations of the volatilities of each simulated portfolio for each strategy. Figure 5.9 shows the distributions of volatilities of the log returns of the two strategies that are furthest apart from each other, that is the hourly rebalancings-strategy (shaded) and the annual rebalancings-strategy. The volatilities of the



**Figure 5.9:** The distributions of the annualized sample standard deviations of the log returns of the "Hourly"-strategy (shaded) and the "Annually"-strategy.

annual-strategy are much more spread out compared to the volatilities of the hourly-strategy. We can conclude that the spread of the volatilities of the rebalancing strategies is negatively correlated with rebalancing frequency. This is not surprising considering that the purpose of the optimal rebalancing strategy is to limit risk. If the return on the risky asset is less than the return on the risk-free asset during the time interval between rebalancing time points, the investor puts the portfolio in a state of balance by reducing the amount invested in the risk-free asset and increasing the amount invested in the risky asset. If the risky asset performs worse than the risk-free asset over a time period, the investor can, by using this strategy, take advantage of a positive rebound of the risky asset. There is a chance however, that the value of the risky asset could decrease regularly over a long period of time. If this would be the case, the rebalancing strategy could actually increase the loss of wealth and utility, since the strategy implies that more and more wealth is reallocated into the risky asset. But the results of the simulations tell us that this is in fact not the case. By using the rebalancing strategy, the investor reduces the downside risk of the portfolio. In an opposite situation, where the risky asset performs better than the risk-free asset, the investor reduces the risky asset investment and increases the risk-free asset investment. This way, the investor is better off if the value of the risky asset goes down, compared to an investor who doesn't rebalance. If, however, the value of the risky asset increases strongly over a long period of time, frequent rebalancings of the portfolio will reduce the potential gain of wealth, compared to infrequent rebalancings or a non-rebalancing strategy. This explains the negative skew we observe in figure 5.5 of the distributions of losses of utility of the rebalancing strategies that involve infrequent rebalancings. We see that on rare occasions, when the development of the risky asset is extremely strong, the rebalancing strategies that involve infrequent rebalancings beat the theoretical strategy of continuous rebalancings by a clear margin. We can conclude that the strategy of holding a constant fraction of the wealth in the risky

asset reduces the potential upside gain as well as reduces the downside risk of the portfolio. But table 5.3 as well as 5.9 also tell us that the range of estimated volatilities of the annual-rebalancing strategy is much wider than the range of estimated volatilities of the hourly-rebalancing strategy. Some of the estimated volatilities of the annual-rebalancing strategy are indeed much lower. This points to the fact that even though the strategy of holding a constant fraction in the risky asset is the optimal strategy for a risk-averse investor, it does not mean that the investor wants to reduce risk at all costs. By reallocating wealth into the risky asset when the risky asset, over a time period, has performed worse than the risk-free asset, the investor is in fact increasing, in relative terms, the risk of the portfolio. The risk is increased in relative terms, because the risk or the potential change of the portfolio value, which in our model are governed by the risky asset drift, the risk-free rent and a Brownian motion, is scaled by the portfolio value itself. The fact that the optimal strategy implies both a relative increase in risk when the risky asset performs worse than risk-free asset and vice versa, makes the optimal strategy a risk-preserving strategy. One might say that, the goal of an investor using the optimal strategy, is to keep the level of risk as high as possible but at the same time below a certain threshold.

But why does this risk-preserving strategy give better Sharpe ratios? The numbers of the column named "Corr." in table 5.3 are measures of the correlations between the estimated log returns and volatilities of all the simulation runs within each rebalancing strategy. From these numbers it becomes clear that there is a negative correlation between rebalancing frequency and the correlation between log returns and volatilities. For the four strategies with the highest rebalancing frequencies, the correlations are close to zero. For the annual-rebalancing strategy the correlation is over 80%. As mentioned above, the risk or the potential change factor of the portfolio is scaled by the portfolio value itself. Higher portfolio values are associated with higher risk. By rebalancing the portfolio frequently, the association between risk and portfolio value is reduced. If the rebalancing frequency is high enough, this association is nearly completely zeroed out. For the rebalancing strategies that involve infrequent rebalancings, the correlation is stronger. This means, that for such rebalancing strategies, high log returns are associated with high volatilities. Remember that the Sharpe ratio is calculated as the ratio between the terminal excess return and the volatility of a portfolio value time series. If  $a, b, c > 0$  and  $a > b$ , then we have that  $\frac{c}{a} < \frac{c}{b}$ . This just means, when calculating the Sharpe ratio, that high excess returns are more likely to be "penalized" by a high estimated volatility, if the correlation between log returns and volatilities are high.

## 5.3 Simulation with transaction costs

### 5.3.1 Introduction

In the portfolio simulations so far we have assumed transaction costs equal to zero, which is a rather unrealistic assumption. To remedy this and to add more realism into the simulations, we will in this section take transaction costs into consideration. Transaction costs can be modelled in various ways, but to keep matters simple we will assume proportional transaction costs. Proportional transaction costs mean that the transactions costs are proportional to the values or the sizes of the asset transactions by a constant factor. There are written several articles addressing Merton's portfolio problem with transaction costs. In 1990, Davis and Norman [4] studied and solved the special case of proportional transaction costs. Their solution means that the incorporation of proportional transaction costs into Merton's portfolio problem changes the optimal asset allocation strategy, which entails that the Merton ratio (3.10) no longer is the optimal strategy. However, the focus of this thesis is to study simulations of portfolios using the optimal strategy found in the original problem as it was formulated by Merton.

In what way should the payments of the transaction costs be implemented? As mentioned earlier, we want the portfolio simulations to be as realistic as possible, and seen from a realistic point of view it is natural to perceive the risk free asset as a bank account. All payments of transaction costs will therefore be deducted from the bank account. The transaction costs can be paid in mainly two ways: one is to make the payment after the portfolio has been rebalanced. The other way is to require the portfolio to be rebalanced after the payment of the transaction cost has been carried out. Among the two methods, the first method is the crude and straightforward way and is probably the method that a real life portfolio manager would use. The second method is a little bit more sophisticated and maybe less realistic. However, it can be argued that the second method reflects the idea of a constant rebalancing strategy more correctly. Thus, both methods are interesting in the context of this thesis and both methods will therefore be implemented.

### 5.3.2 Simulation model II

Assume now that transaction costs are paid after the portfolio has been rebalanced (we will hereafter refer to this method as subsequent transaction costs). For the portfolio to be rebalanced in this setting, we need to recalculate the transaction size  $Q_k$ . Let the transaction cost proportionality constant be denoted by  $\lambda$ . Remember that the set of rebalancing time points is given by  $\mathcal{T}^{\text{reb}} = \{t_0, t_\epsilon, t_{2\epsilon}, \dots, t_n\}$ , and that the last rebalancing time point relative to

the current time point in which we want to simulate the portfolio, is given by  $t_{k^*}$ . Assume that  $t_k$  is a rebalancing time point and let the value of the transaction at time  $t_k$  be denoted by  $Q_k$ . The proportionality of the transaction cost simply means that the transaction cost is equal to  $\lambda|Q_k|$ . The value of the transaction itself is the same as in the setting with no transaction costs (5.4). Compared to simulation model I of section 5.2, the inclusion of subsequent transaction costs gives the following slightly modified simulation scheme:

### Simulation model II

Transaction costs: subsequent

Volatility: constant

$$\begin{aligned}
\tilde{V}_k^{\prime S} &= u^* \tilde{V}_{k^*} \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) \\
\tilde{V}_k^{\prime R} &= (1 - u^*) \tilde{V}_{k^*} (1 + r\delta)^{k-k^*} \\
Q_k &= (1 - u^*) \tilde{V}_k^{\prime S} - u^* \tilde{V}_k^{\prime R} \\
\tilde{V}_k^S &= \begin{cases} \tilde{V}_k^{\prime S} - Q_k, & t_k \in \mathcal{T}^{\text{reb}} \\ \tilde{V}_k^{\prime S}, & \text{otherwise} \end{cases} \\
\tilde{V}_k^R &= \begin{cases} \tilde{V}_k^{\prime R} + Q_k - \lambda|Q_k|, & t_k \in \mathcal{T}^{\text{reb}} \\ \tilde{V}_k^{\prime R}, & \text{otherwise} \end{cases} \\
\tilde{V}_k &= \tilde{V}_k^S + \tilde{V}_k^R.
\end{aligned} \tag{5.11}$$

Similarly to the simulation model I of section 5.2, this simulation scheme can also be restated in a more compact way,

$$\tilde{V}_k = \begin{cases} \begin{cases} u^* \tilde{V}_{k^*} \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) + (1 - u^*) \tilde{V}_{k^*} (1 + r\delta)^{k-k^*} \\ -\lambda u^* (1 - u^*) \tilde{V}_{k^*} \left| \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) - (1 + r\delta)^{k-k^*} \right| \end{cases}, & t_k \in \mathcal{T}^{\text{reb}} \\ u^* \tilde{V}_{k^*} \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) + (1 - u^*) \tilde{V}_{k^*} (1 + r\delta)^{k-k^*}, & \text{otherwise.} \end{cases}$$



### 5.3.3 Simulation model III

Assume instead that transaction costs are paid before the portfolio is rebalanced (we will hereafter refer to this method as preceding transaction costs). Let the difference between the return on the risky asset and the return on the risk-free asset since the last rebalancing time point  $t_{k^*}$  at time  $t_k$  be denoted by  $D_k$ , that is

$$D_k = \left( \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) - 1 \right) - ((1 + r\delta)^{k-k^*} - 1). \quad (5.12)$$

As we have seen earlier, it is clear that the direction of the transaction between the risky and the risk free asset investment to rebalance the portfolio, only depends on the sign of the difference in returns on the investments, that is the sign of  $D_k$ . Still assume that  $t_k$  is a rebalancing time point. Remember that  $\tilde{V}_k'^S$  and  $\tilde{V}_k'^R$  are the values of the risky asset and risk free asset investment, respectively, before the portfolio is rebalanced. For the portfolio to be rebalanced after transaction costs have been paid, the following relations have to be fulfilled:

$$u^* = \frac{\tilde{V}_k'^S - Q_k}{\tilde{V}_k},$$

$$1 - u^* = \begin{cases} \frac{\tilde{V}_k'^R + Q_k - \lambda Q_k}{\tilde{V}_k}, & D_k \geq 0 \\ \frac{\tilde{V}_k'^R + Q_k + \lambda Q_k}{\tilde{V}_k}, & D_k < 0 \end{cases}.$$

Solving these equations with respect to  $\tilde{V}_k$  and then putting the solutions together yields,

$$\frac{\tilde{V}_k'^S - Q_k}{u^*} = \begin{cases} \frac{\tilde{V}_k'^R + Q_k - \lambda Q_k}{1 - u^*}, & D_k \geq 0 \\ \frac{\tilde{V}_k'^R + Q_k + \lambda Q_k}{1 - u^*}, & D_k < 0 \end{cases}.$$

Finally, solving this equation with respect to  $Q_k$  gives the following solution:

$$Q_k = \begin{cases} \frac{(1 - u^*)\tilde{V}_k'^S - u^*\tilde{V}_k'^R}{1 - \lambda u^*}, & D_k \geq 0 \\ \frac{(1 - u^*)\tilde{V}_k'^S - u^*\tilde{V}_k'^R}{1 + \lambda u^*}, & D_k < 0 \end{cases}.$$

The solution is almost equal to the solution (5.4) of section 5.2 except for the additional expressions in the denominators. We notice that if  $Q_k \geq 0$ , then  $1 - \lambda u^* \leq 1$ , which reflects the fact that the portfolio manager has to take into account the deduction of the transaction cost from the bank account before the portfolio is rebalanced. As a consequence she has to make a bigger transfer from the risky asset investment to ensure that the portfolio becomes rebalanced, compared to the setting with no transaction costs or subsequent transactions costs. If  $Q_k < 0$  she needs to transfer less than before, since the deduction of the transaction cost itself contributes towards a rebalanced portfolio. The inclusion of preceding transaction costs gives the following, slightly modified, simulation scheme:

### Simulation model III

Transaction costs: preceding

Volatility: constant

$$\begin{aligned}
\tilde{V}_k'^S &= u^* \tilde{V}_{k^*} \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) \\
\tilde{V}_k'^R &= (1 - u^*) \tilde{V}_{k^*} (1 + r\delta)^{k-k^*} \\
D_k &= \left( \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) - 1 \right) - ((1 + r\delta)^{k-k^*} - 1) \\
Q_k &= \begin{cases} \frac{(1 - u^*) \tilde{V}_k'^S - u^* \tilde{V}_k'^R}{1 - \lambda u^*}, & D_k \geq 0 \\ \frac{(1 - u^*) \tilde{V}_k'^S - u^* \tilde{V}_k'^R}{1 + \lambda u^*}, & D_k < 0 \end{cases} \quad (5.13) \\
\tilde{V}_k^S &= \begin{cases} \tilde{V}_k'^S - Q_k, & t_k \in \mathcal{T}^{\text{reb}} \\ \tilde{V}_k'^S, & \text{otherwise} \end{cases} \\
\tilde{V}_k^R &= \begin{cases} \tilde{V}_k'^R + Q_k - \lambda|Q_k|, & t_k \in \mathcal{T}^{\text{reb}} \\ \tilde{V}_k'^R, & \text{otherwise} \end{cases} \\
\tilde{V}_k &= \tilde{V}_k^S + \tilde{V}_k^R.
\end{aligned}$$

A shorter representation of this simulation scheme is stated as follows,

$$\tilde{V}_k = \begin{cases} \begin{cases} u^* \tilde{V}_{k^*} \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) + (1 - u^*) \tilde{V}_{k^*} (1 + r\delta)^{k-k^*} \\ - \frac{\lambda u^* (1 - u^*) \tilde{V}_{k^*} \left| \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) - (1 + r\delta)^{k-k^*} \right|}{1 - \lambda u^* \operatorname{sgn} \left( \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) - (1 + r\delta)^{k-k^*} \right)}, & t_k \in \mathcal{T}^{\text{reb}} \end{cases} \\ u^* \tilde{V}_{k^*} \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma\Delta B_j) + (1 - u^*) \tilde{V}_{k^*} (1 + r\delta)^{k-k^*}, & \text{otherwise} \end{cases}$$

where the function  $\operatorname{sgn}(x)$  is defined as

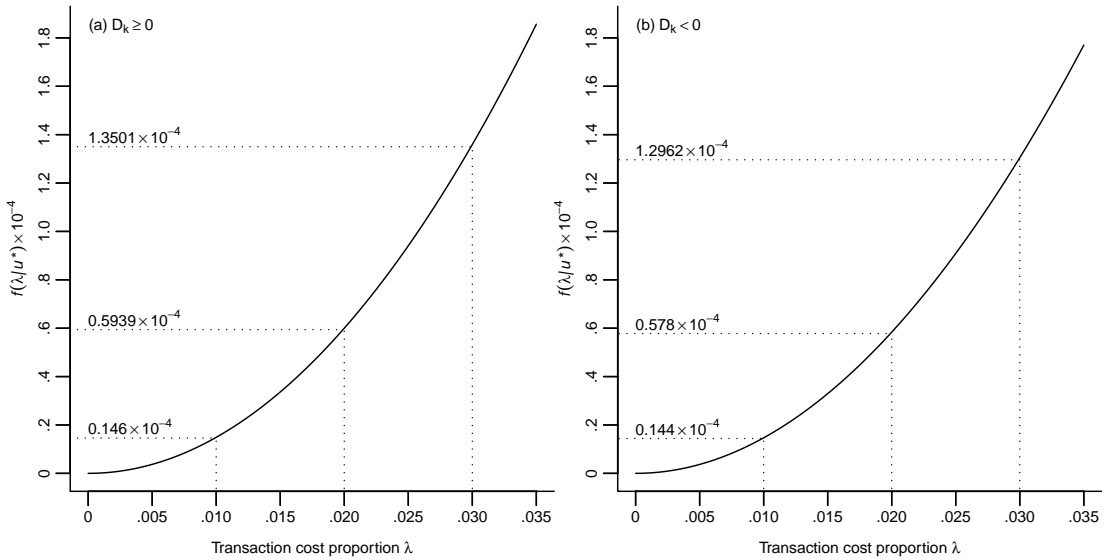
$$\operatorname{sgn}(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases}.$$

The question now is, how will the two slightly different simulation schemes perform and how will they compare against each other? Which strategy is the most profitable? The only difference between the strategies are the transaction costs  $\{\lambda|Q_k|\}_{k \in \mathcal{T}^{\text{reb}}}$ . Assume that  $\lambda|Q_k^{\text{pre}}|$  and  $\lambda|Q_k^{\text{sub}}|$  denote the transaction costs of the simulation scheme with preceding transaction costs and with subsequent transaction costs, respectively. We have that

$$\begin{aligned} \lambda|Q_k^{\text{pre}}| - \lambda|Q_k^{\text{sub}}| &= \begin{cases} \lambda Q_k^{\text{pre}} - \lambda Q_k^{\text{sub}}, & D_k \geq 0 \\ \lambda Q_k^{\text{sub}} - \lambda Q_k^{\text{pre}} = -(\lambda Q_k^{\text{pre}} - \lambda Q_k^{\text{sub}}), & D_k < 0 \end{cases} \\ &= \begin{cases} \frac{\lambda \left( (1 - u^*) \tilde{V}_k'^S - u^* \tilde{V}_k'^R \right)}{1 - \lambda u^*} - \lambda \left( (1 - u^*) \tilde{V}_k'^S - u^* \tilde{V}_k'^R \right), & D_k \geq 0 \\ - \left( \frac{\lambda \left( (1 - u^*) \tilde{V}_k'^S - u^* \tilde{V}_k'^R \right)}{1 + \lambda u^*} - \lambda \left( (1 - u^*) \tilde{V}_k'^S - u^* \tilde{V}_k'^R \right) \right), & D_k < 0 \end{cases} \\ &= \begin{cases} \frac{\lambda^2 u^*}{1 - \lambda u^*} \left( (1 - u^*) \tilde{V}_k'^S - u^* \tilde{V}_k'^R \right), & D_k \geq 0 \\ \frac{\lambda^2 u^*}{1 + \lambda u^*} \left( (1 - u^*) \tilde{V}_k'^S - u^* \tilde{V}_k'^R \right), & D_k < 0 \end{cases} \\ &= \begin{cases} \frac{\lambda^2 u^{*2} (1 - u^*)}{1 - \lambda u^*} \tilde{V}_{k^*} D_k, & D_k \geq 0 \\ \frac{\lambda^2 u^{*2} (1 - u^*)}{1 + \lambda u^*} \tilde{V}_{k^*} D_k, & D_k < 0 \end{cases} \end{aligned} \quad (5.14)$$

Given a portfolio value  $\tilde{V}_{k^*}$  at the previous rebalancing time point  $t_{k^*}$ , we see that the difference in transaction cost at time  $t_k$  is simply a function of the difference in

returns on the risky and the risk free asset investments times  $\tilde{V}_{k^*}$  times a constant. We also see that the difference between preceding and subsequent transaction cost depends on the direction of the transaction, which in turn depends on the difference in return on the risky asset and the risk-free asset, which is given by  $D_k$ .  $D_k > 0$  favours the subsequent transaction cost strategy, whereas  $D_k < 0$  favours the preceding transaction cost strategy. The plots of figure 5.10 shows



**Figure 5.10:** (a)  $f(\lambda|u^*) = (\lambda^2 u^{*2}(1-u^*)) / (1-\lambda u^*)$ , (b)  $f(\lambda|u^*) = (\lambda^2 u^{*2}(1-u^*)) / (1+\lambda u^*)$

how the constants  $(\lambda^2 u^{*2}(1-u^*)) / (1-\lambda u^*)$  and  $(\lambda^2 u^{*2}(1-u^*)) / (1+\lambda u^*)$  increases exponentially as a function of the proportionality constant  $\lambda$ .

What values of  $\lambda$  are reasonable seen from a realistic point of view? That could depend on various factors such as the size of the transaction, the size and power of the company involved in the transaction, the relation between the company and the broker and probably many other factors. According to the thesis supervisor .02 – .03 could be reasonable values for a small player in the market. A large enough player could perhaps achieve less than .01. To be on the safe side we will consider different values, .01, .02 and .03, for  $\lambda$  in the calculations of the transaction costs. In figure 5.10, these particular values on the horizontal axis and the corresponding values as a function of  $\lambda$  on the vertical axis are indicated by the dotted lines. The exponential relationship means for instance that a tripling of the transaction cost proportion from  $\lambda = \lambda_1 = .01$  to  $\lambda = \lambda_3 = .03$ ,

will imply

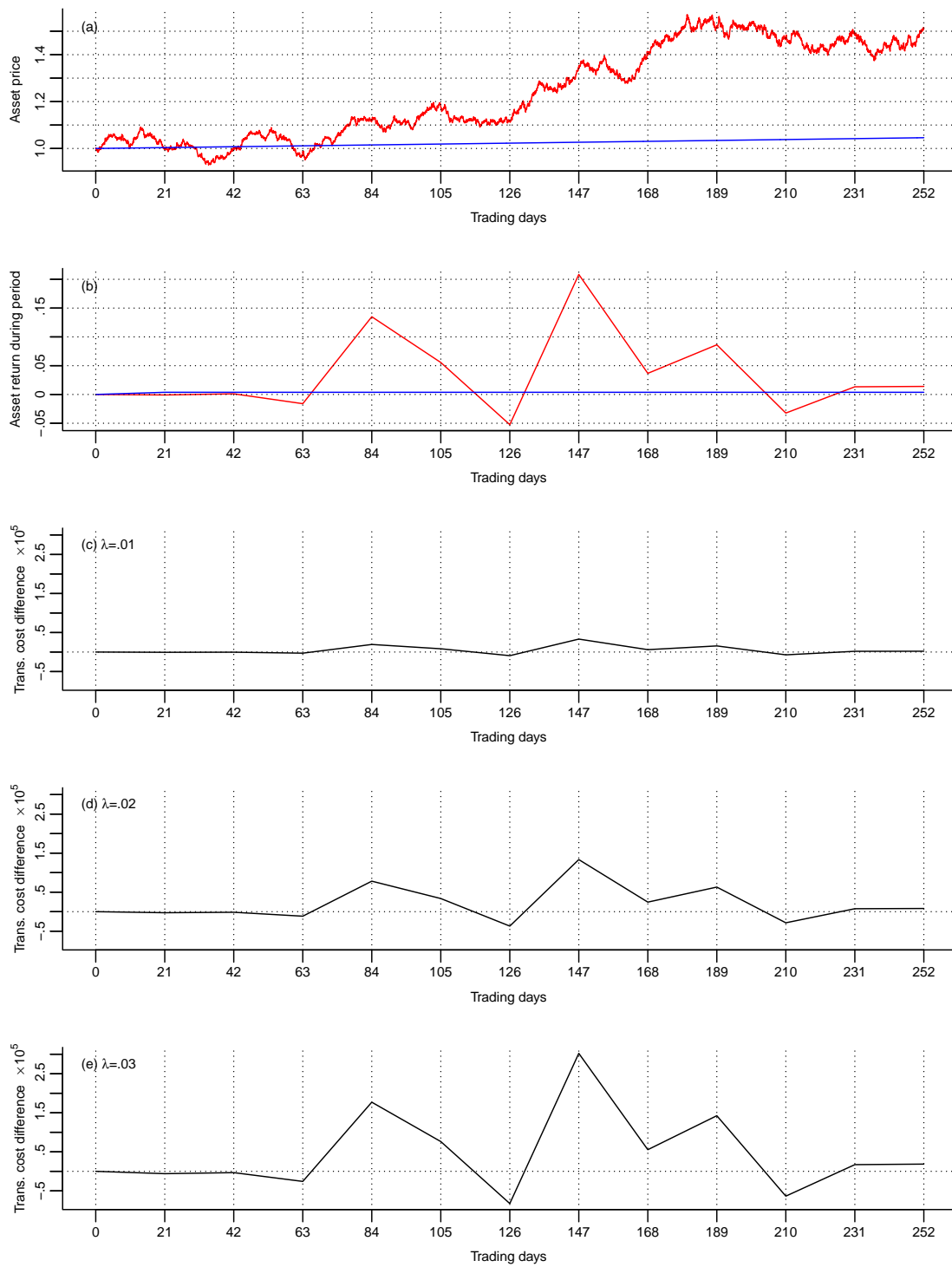
$$\begin{aligned}
& \left\{ \frac{\lambda_3^2 u^{*2} (1 - u^*)}{1 - \lambda_3 u^*} \bigg/ \frac{\lambda_1^2 u^{*2} (1 - u^*)}{1 - \lambda_1 u^*} = \frac{(3\lambda_1)^2 u^{*2} (1 - u^*)}{1 - 3\lambda_1 u^*} \bigg/ \frac{\lambda_1^2 u^{*2} (1 - u^*)}{1 - \lambda_1 u^*} \right. \\
& \left. \frac{\lambda_3^2 u^{*2} (1 - u^*)}{1 + \lambda_3 u^*} \bigg/ \frac{\lambda_1^2 u^{*2} (1 - u^*)}{1 + \lambda_1 u^*} = \frac{(3\lambda_1)^2 u^{*2} (1 - u^*)}{1 + 3\lambda_1 u^*} \bigg/ \frac{\lambda_1^2 u^{*2} (1 - u^*)}{1 + \lambda_1 u^*} \right. \\
& = \begin{cases} \frac{9(1 - \lambda_1 u^*)}{1 - 3\lambda_1 u^*} = 9.1251 \\ \frac{9(1 + \lambda_1 u^*)}{1 + 3\lambda_1 u^*} = 8.8799 \end{cases} \tag{5.15}
\end{aligned}$$

approximately, a nine-time increase in the transaction cost difference between the two transaction cost strategies, assuming equal values for  $\tilde{V}_{k^*}$  and  $D_k$ . In reality this difference will be slightly lower considering that the returns on the portfolio will be reduced due to the increased transaction costs. The equations (5.14) also tell us that the strategy of subsequent transaction costs is slightly better if the return on the risky asset investment is greater than the return on the risk free asset investment since the previous rebalancing time point  $k^*$ . If the return on the risk free asset is greater, then the strategy of preceding transaction costs is better. This might suggest that we ought to choose the strategy of subsequent transaction costs if we expect the risky asset to beat the risk free asset in the market, and vice versa. In the next section we will through two simulation test runs investigate this further.

### 5.3.4 Simulation test runs

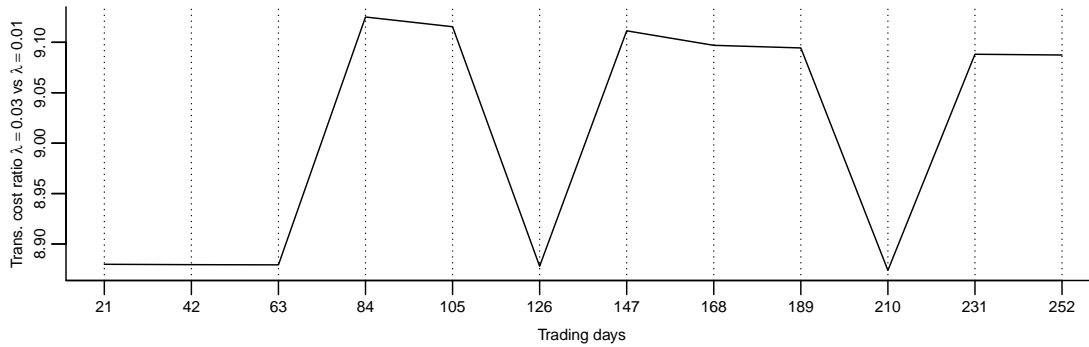
How does the incorporation of transaction costs into the portfolio model affect the development of the value and the utility of the portfolio over time? In this section we will try to answer this question through the analysis of a complete (one year) time series of simulated portfolio values, including both preceding and subsequent transaction costs. As in section 5.2, the simulation algorithm first simulates a Brownian motion time series. The Brownian motion is updated hourly over one year, that is 252 trading days, which result in a Brownian motion time series consisting of 6048 points. The same Brownian motion time series is then used to calculate a theoretical portfolio time series and to simulate a portfolio without transaction costs that will serve as an alternative baseline for comparison, a portfolio with preceding transaction costs (5.13) and a portfolio with subsequent transaction costs (5.11). By using the same Brownian motion in the calculation of the theoretical portfolio and in the, up till now, three different portfolio simulation schemes, it will be easier to make comparisons.

The plots of figure 5.11 continues on the discussion of preceding versus subsequent transaction costs of the previous subsection. The plot of subfigure (a) shows a



**Figure 5.11:** (a) asset prices, (b) asset returns, (c) transaction cost differences  $\lambda = .01$ , (d)  $\lambda = .02$  and (e)  $\lambda = .03$ .

random development of the two possible investment objects, namely the risky asset (red) and the risk free asset (blue). In this particular simulation run the risky asset beats the risk free asset by a clear margin. This is reflected in the plot of subfigure (b) which shows the returns on each asset during the time periods between the rebalancing time points. The plots of subfigures (c), (d) and (e) show the transaction cost differences  $\{\lambda|Q_k^{\text{pre}}| - \lambda|Q_k^{\text{sub}}|\}_{k \in \mathcal{T}^{\text{pre}}}$  at each rebalancing time point for different values of the transaction cost proportionality constant  $\lambda$ . We observe that large changes in the risky asset value require large (in relative terms) transactions to rebalance the portfolio. These plots also confirm that the transaction cost differences are a simple function of the risky and the risk-free asset returns and that the differences increase exponentially as a function of  $\lambda$ .



**Figure 5.12:** Transaction cost difference ratio,  $\lambda = .03$  versus  $\lambda = .01$ .

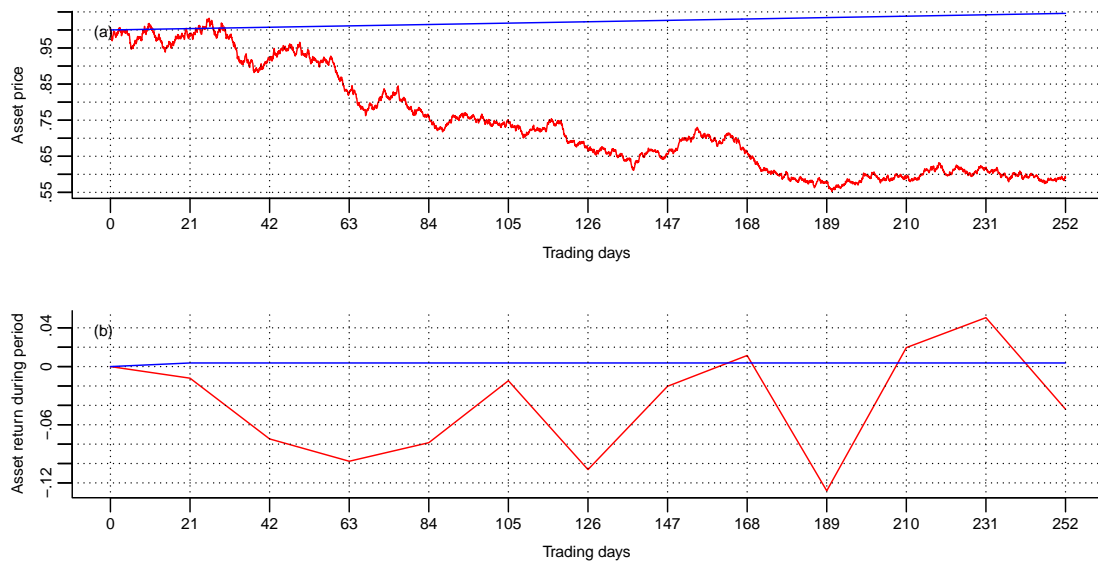
The plot of figure 5.12 displays transaction costs difference ratios at each rebalancing time point, comparing  $\lambda = .03$  versus  $\lambda = .01$  using the same Brownian motion. They are in agreement with the previous calculations (5.15). Table 5.4

	Simulation model	Terminal wealth	Loss of wealth	Terminal utility	Loss of utility	Total transaction costs	
Transaction costs	Theoretical	1.3537	0	1.1725	0	0	
	None	1.3535	$.1758 \times 10^{-3}$	1.1724	$.0800 \times 10^{-3}$	0	
	Preceding	$\lambda = .01$	1.3517	$1.9476 \times 10^{-3}$	1.1716	$.8868 \times 10^{-3}$	$1.5722 \times 10^{-3}$
		$\lambda = .02$	1.3500	$3.7323 \times 10^{-3}$	1.1708	$1.6999 \times 10^{-3}$	$3.1561 \times 10^{-3}$
		$\lambda = .03$	1.3482	$5.5303 \times 10^{-3}$	1.1700	$2.5196 \times 10^{-3}$	$4.7521 \times 10^{-3}$
	Subsequent	$\lambda = .01$	1.3517	$1.9404 \times 10^{-3}$	1.1716	$.8835 \times 10^{-3}$	$1.5656 \times 10^{-3}$
		$\lambda = .02$	1.3500	$3.7032 \times 10^{-3}$	1.1708	$1.6867 \times 10^{-3}$	$3.1294 \times 10^{-3}$
$\lambda = .03$		1.3482	$5.4641 \times 10^{-3}$	1.1700	$2.4894 \times 10^{-3}$	$4.6914 \times 10^{-3}$	

**Table 5.4:** Summary of the first simulation run with transaction costs incorporated.

summarizes the first simulation run with transaction costs incorporated. It is of

course clear that the incorporation of transaction costs into the portfolio simulation model entail a loss of both wealth and utility, compared to the theoretical model or the model with no transaction costs. The losses are however quite small. Even for the models with the highest transaction costs ( $\lambda = .3$ ), the loss of wealth is only about .4% and the loss of utility is only about .2%. Of course, for a portfolio of great value this loss could still be quite significant. As for the transaction cost totals, they make up about 80-90% of the total losses of wealth for their respective portfolio models. The remainder of the total losses is made up of the lower returns on the portfolio. Which transaction cost strategy gives the smallest loss? The differences between the strategies are very small, but the losses of the portfolios that use the strategy of subsequent transaction costs are slightly smaller, which is as expected based on the conclusions of the previous section. For the sake of comparison we will include a second simulation run.



**Figure 5.13:** (a) asset prices, (b) asset returns

The plots of figure 5.13 show the price developments (a) and the returns during the time periods between rebalancing time points (b) of the risky asset (red) and the risk free asset (blue) of the second simulation run. This time the performance of the risky asset is quite bad: the risky asset price decrease nearly 50% over the simulated time period of one year. By observing the quantities of table 5.5 we can conclude that for the second simulation run, the strategy of preceding transaction costs gives slightly smaller losses of wealth and utility, compared to the strategy of subsequent transaction costs. This is the opposite conclusion of the first simulation run and in accordance with the conclusions of the previous section. If we take a look at the total transaction costs we see that they actually are larger than the total losses of wealth for their respective strategies and parameter configurations.



	Simulation scheme	Terminal wealth	Loss of wealth	Terminal utility	Loss of utility	Total transaction costs	
	Theoretical	0.7153	0	0.8386	0	0	
Transaction costs	None	0.7151	$0.2083 \times 10^{-3}$	0.8384	$0.1283 \times 10^{-3}$	0	
	Preceding	$\lambda = .01$	0.7140	$1.3149 \times 10^{-3}$	0.8378	$0.8104 \times 10^{-3}$	$1.2377 \times 10^{-3}$
		$\lambda = .02$	0.7129	$2.4081 \times 10^{-3}$	0.8371	$1.4847 \times 10^{-3}$	$2.4602 \times 10^{-3}$
		$\lambda = .03$	0.7118	$3.4883 \times 10^{-3}$	0.8364	$2.1514 \times 10^{-3}$	$3.6680 \times 10^{-3}$
	Subsequent	$\lambda = .01$	0.7140	$1.3210 \times 10^{-3}$	0.8378	$0.8141 \times 10^{-3}$	$1.2446 \times 10^{-3}$
		$\lambda = .02$	0.7129	$2.4322 \times 10^{-3}$	0.8371	$1.4995 \times 10^{-3}$	$2.4877 \times 10^{-3}$
		$\lambda = .03$	0.7118	$3.5419 \times 10^{-3}$	0.8364	$2.1845 \times 10^{-3}$	

**Table 5.5:** Summary of the second simulation run with transaction costs incorporated.

Let  $\tilde{V}_k$  denote the simulated portfolio value at time  $t_k$  without transaction costs and let  $\tilde{V}_k^{\text{tc}}$  denote the simulated portfolio value at time  $t_k$  with transaction costs. Assume that  $\tilde{V}_0 = \tilde{V}_0^{\text{tc}}$ . Remember that the set of rebalancing time points  $\mathcal{T}^{\text{reb}}$  can be defined by the distance  $\epsilon$  between the rebalancing time points indices, that is  $\mathcal{T}^{\text{reb}} = \{t_\epsilon, t_{2\epsilon}, \dots, t_n\}$  ( $\frac{n}{\epsilon} \in \mathcal{N}$ ). Hence, the first rebalancing time point is  $t_\epsilon$ . At the first rebalancing time point, we have that

$$\begin{aligned} \tilde{V}_\epsilon - \tilde{V}_\epsilon^{\text{tc}} &= u^* \tilde{V}_0 \prod_{j=0}^{\epsilon-1} (1 + \mu\delta + \sigma\Delta B_j) + (1 - u^*) \tilde{V}_0 (1 + r\delta)^\epsilon \\ &\quad - \left( u^* \tilde{V}_0^{\text{tc}} \prod_{j=0}^{\epsilon-1} (1 + \mu\delta + \sigma\Delta B_j) + (1 - u^*) \tilde{V}_0^{\text{tc}} (1 + r\delta)^\epsilon - \lambda |Q_\epsilon| \right) \\ &= \lambda |Q_\epsilon|. \end{aligned}$$

At the second rebalancing time point, we have that

$$\begin{aligned} \tilde{V}_{2\epsilon} - \tilde{V}_{2\epsilon}^{\text{tc}} &= u^* \tilde{V}_\epsilon \prod_{j=\epsilon}^{2\epsilon-1} (1 + \mu\delta + \sigma\Delta B_j) + (1 - u^*) \tilde{V}_\epsilon (1 + r\delta)^\epsilon \\ &\quad - \left( u^* \tilde{V}_\epsilon^{\text{tc}} \prod_{j=\epsilon}^{2\epsilon-1} (1 + \mu\delta + \sigma\Delta B_j) + (1 - u^*) \tilde{V}_\epsilon^{\text{tc}} (1 + r\delta)^\epsilon - \lambda |Q_{2\epsilon}| \right) \\ &= (\tilde{V}_\epsilon - \tilde{V}_\epsilon^{\text{tc}}) \left( u^* \prod_{j=\epsilon}^{2\epsilon-1} (1 + \mu\delta + \sigma\Delta B_j) + (1 - u^*) (1 + r\delta)^\epsilon \right) + \lambda |Q_{2\epsilon}| \\ &= \lambda |Q_\epsilon| \left( u^* \prod_{j=\epsilon}^{2\epsilon-1} (1 + \mu\delta + \sigma\Delta B_j) + (1 - u^*) (1 + r\delta)^\epsilon \right) + \lambda |Q_{2\epsilon}|, \end{aligned}$$

and so on. This means that the difference between a simulated portfolio value without transaction costs incorporated and one with, is not simply the sum of the

transaction costs at each rebalancing time point (except at the first rebalancing time point). If the expression in the parentheses is less than one, the loss of wealth will be smaller than the sum of the transaction costs, and vice versa. This is also what we basically observed in the comparisons of the two simulation runs.

### 5.3.5 Mean loss of utility

In this subsection we will look at the results of 100,000 simulation runs<sup>1</sup> of each combination of the input parameter triplet, rebalancing strategy, transaction cost strategy and transaction cost proportion. Regarding the rebalancing strategies, we will include the same strategies as we have done so far in our simulations. As for the transaction cost strategies, we will include both preceding and subsequent transaction costs. We will also include transaction costs proportions equal to .01, .02 and .03 in our simulations as well as no transaction costs for the sake of comparison. More specifically, for each pairwise combination of transaction cost proportion and rebalancing strategy we will do 100,000 simulation runs. For each simulation run, we will generate one Brownian motion consisting of 6048 points that will be used to simulate a time series of theoretical portfolio values, a time series of simulated portfolio values with no transaction costs, a time series of simulated portfolio values using preceding transaction costs and a time series of simulated portfolio values using subsequent transaction costs. This will result in a total of 10.8 million portfolio value time series, from a total of 2.7 million Brownian motions, each consisting of 6048 time points, giving the possibility to calculate 36 mean losses of utility.

The first batch of simulation runs assumes transaction cost proportion  $\lambda = .01$ . The plots of figure 5.14 show the mean losses of utility versus number of simulations for a selection of four rebalancing strategies, namely hourly, daily, monthly and annually. The choice of transaction cost method is also included in the plots. According to the plots, the mean losses of utility seem to converge toward a constant value, similar to what we saw with simulation model I in section 5.2.

Table 5.6 summarizes the results of the first batch of simulation runs with transaction cost proportion equal to .01. The abbreviations of the third column of the table need an explanation. "Th" is an abbreviation for "theoretical". The "Th"-rows show the results of the simulations of the theoretical portfolio values, calculated according to equation (5.2), that is the exact solution of the continuous

---

<sup>1</sup>In the previous section, where the focus was on simulation model I, we performed 1,000,000 simulation runs for each rebalancing strategy. Because of the added complexity of simulation model II and III, which entails both an increased number of parameter combinations that need to be simulated, and more complex, slower running simulation algorithms, we need to reduce the number of simulation runs for each parameter combination, in order to attain acceptable simulation running times.

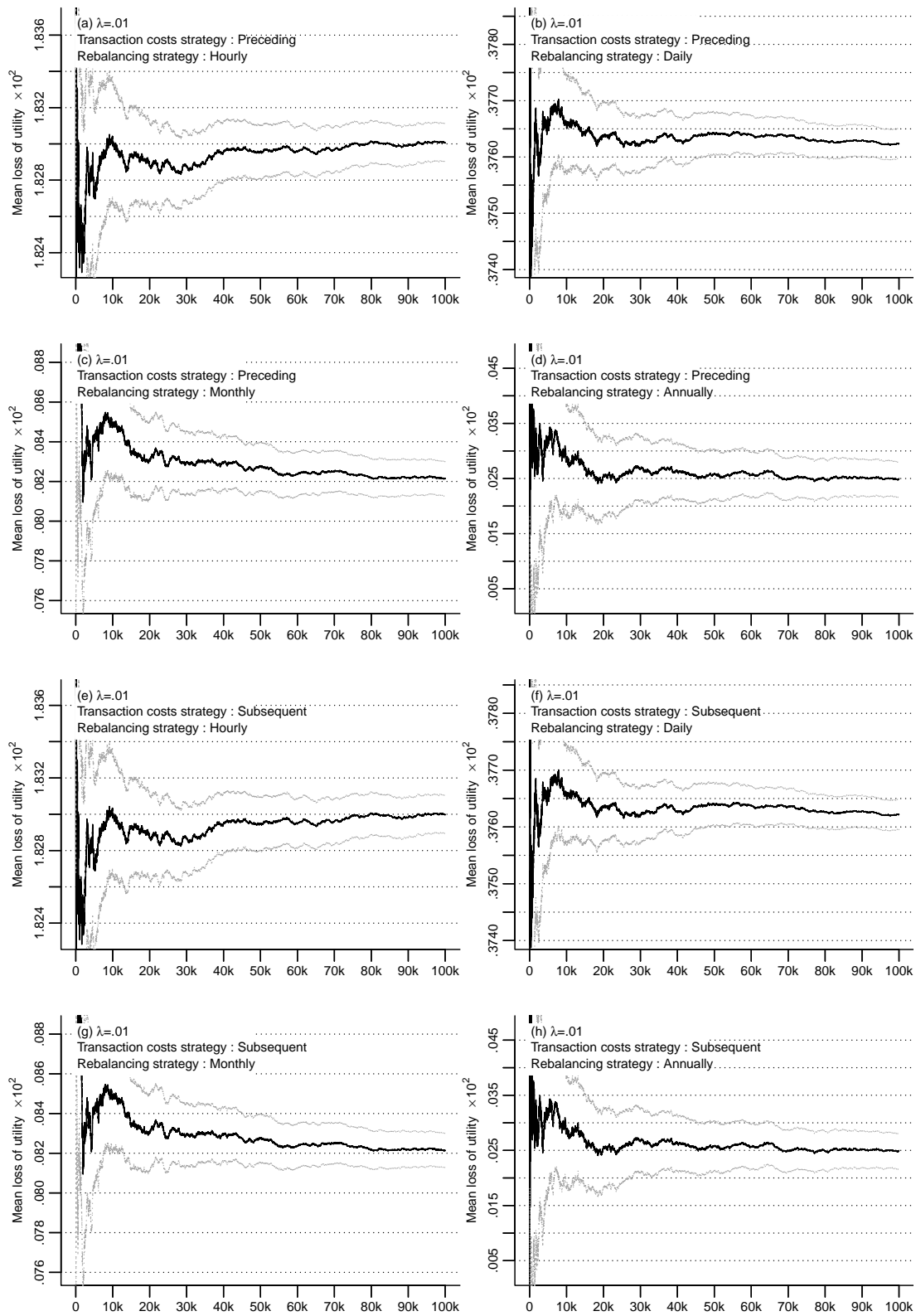


Figure 5.14: The mean losses of utility with transaction cost proportion  $\lambda = .01$ . (a)-(d) preceding transaction costs and (e)-(h) subsequent transaction costs.

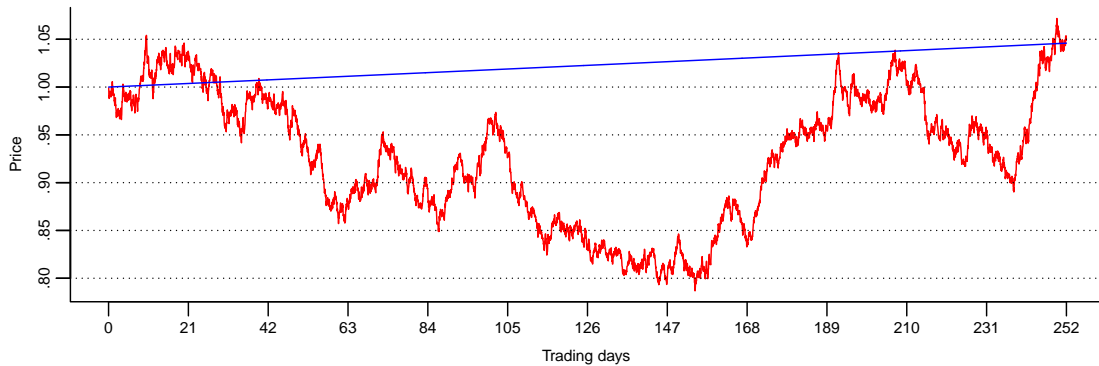
$\lambda = .01$		Sample means				StDev	
		Term. wealth	Total cost	Term. utility	Loss of utility	Loss of utility	
Rebalancing strategy	Hourly	Th	1.0607	-	1.0276	-	-
		No	1.0607	-	1.0276	$-.0001 \times 10^{-2}$	$.1464 \times 10^{-3}$
		Pre	1.0250	$3.4621 \times 10^{-2}$	1.0093	$1.8301 \times 10^{-2}$	$1.6891 \times 10^{-3}$
		Sub	1.0250	$3.4619 \times 10^{-2}$	1.0093	$1.8300 \times 10^{-2}$	$1.6870 \times 10^{-3}$
	Every 4th hour	Th	1.0607	-	1.0276	-	-
		No	1.0607	-	1.0276	0	$.1893 \times 10^{-3}$
		Pre	1.0428	$1.7458 \times 10^{-2}$	1.0185	$.9191 \times 10^{-2}$	$.8713 \times 10^{-3}$
		Sub	1.0428	$1.7457 \times 10^{-2}$	1.0185	$.9191 \times 10^{-2}$	$.8693 \times 10^{-3}$
	Daily	Th	1.0608	-	1.0277	-	-
		No	1.0608	-	1.0277	0	$.3619 \times 10^{-3}$
		Pre	1.0534	$.7168 \times 10^{-2}$	1.0239	$.3762 \times 10^{-2}$	$.4267 \times 10^{-3}$
		Sub	1.0534	$.7167 \times 10^{-2}$	1.0239	$.3762 \times 10^{-2}$	$.4251 \times 10^{-3}$
	Every 3rd day	Th	1.0611	-	1.0278	-	-
		No	1.0611	-	1.0278	$-.0001 \times 10^{-2}$	$.4954 \times 10^{-3}$
		Pre	1.0559	$.5075 \times 10^{-2}$	1.0252	$.2662 \times 10^{-2}$	$.4315 \times 10^{-3}$
		Sub	1.0559	$.5075 \times 10^{-2}$	1.0252	$.2662 \times 10^{-2}$	$.4304 \times 10^{-3}$
	Every 12th day	Th	1.0603	-	1.0274	-	-
		No	1.0603	-	1.0274	$.0006 \times 10^{-2}$	$1.1737 \times 10^{-3}$
		Pre	1.0582	$.2074 \times 10^{-2}$	1.0263	$.1092 \times 10^{-2}$	$1.0142 \times 10^{-3}$
		Sub	1.0582	$.2074 \times 10^{-2}$	1.0263	$.1092 \times 10^{-2}$	$1.0140 \times 10^{-3}$
	Monthly	Th	1.0617	-	1.0281	-	-
		No	1.0617	-	1.0281	$-.0002 \times 10^{-2}$	$1.5543 \times 10^{-3}$
		Pre	1.0601	$.1574 \times 10^{-2}$	1.0273	$.0821 \times 10^{-2}$	$1.3880 \times 10^{-3}$
		Sub	1.0601	$.1574 \times 10^{-2}$	1.0273	$.0821 \times 10^{-2}$	$1.3880 \times 10^{-3}$
Bimonthly	Th	1.0604	-	1.0275	-	-	
	No	1.0604	-	1.0275	$-.0004 \times 10^{-2}$	$2.2016 \times 10^{-3}$	
	Pre	1.0593	$.1115 \times 10^{-2}$	1.0269	$.0577 \times 10^{-2}$	$2.0325 \times 10^{-3}$	
	Sub	1.0593	$.1114 \times 10^{-2}$	1.0269	$.0577 \times 10^{-2}$	$2.0325 \times 10^{-3}$	
Semiannualy	Th	1.0610	-	1.0277	-	-	
	No	1.0610	-	1.0277	0	$3.7754 \times 10^{-3}$	
	Pre	1.0603	$.0651 \times 10^{-2}$	1.0274	$.0335 \times 10^{-2}$	$3.6069 \times 10^{-3}$	
	Sub	1.0603	$.0650 \times 10^{-2}$	1.0274	$.0335 \times 10^{-2}$	$3.6070 \times 10^{-3}$	
Annually	Th	1.0609	-	1.0277	-	-	
	No	1.0610	-	1.0277	$.0014 \times 10^{-2}$	$5.3259 \times 10^{-3}$	
	Pre	1.0605	$.0465 \times 10^{-2}$	1.0275	$.0250 \times 10^{-2}$	$5.1586 \times 10^{-3}$	
	Sub	1.0605	$.0464 \times 10^{-2}$	1.0275	$.0250 \times 10^{-2}$	$5.1587 \times 10^{-3}$	

**Table 5.6:** Mean losses of utility and other statistics,  $\lambda = .01$ .

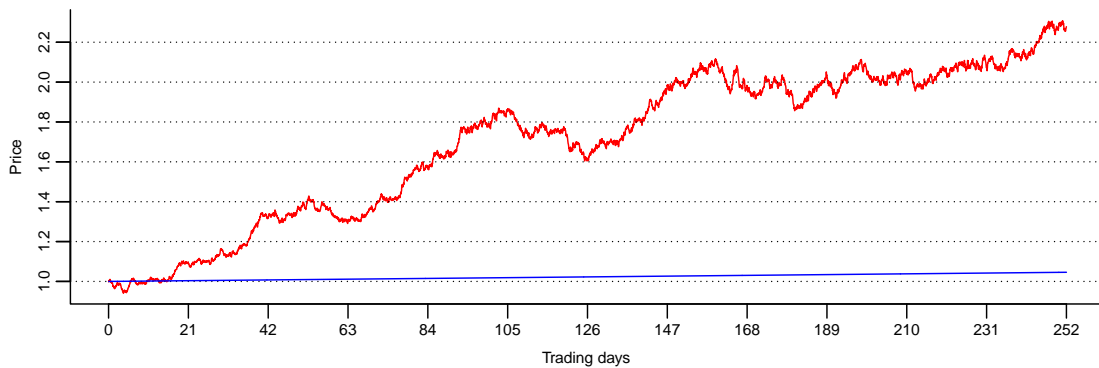
time SDE (3.2), which implies no transaction costs and continuous rebalancing of the portfolio. This is the same baseline as was used in section 5.2. This baseline will also be used in calculations of loss of utility. Also included are discrete time portfolio simulations without transaction costs, i.e. simulations based on simulation model I of section 5.2. The results of these simulations are given in the rows named "No", which are an abbreviation for "no transaction costs". These results function as an alternative baseline. The "Pre"-rows show the results of the simulated discrete time portfolio values assuming preceding transaction costs, i.e. simulations based on simulation model III. Finally, the rows of the "Sub"-category are the results of the simulated discrete time portfolio values assuming subsequent transaction costs, i.e. simulations based on simulation model II.

So what do the summary of results of table 5.6 tell us about the mean loss of utility? Clearly, the introduction of transaction costs into the simulation model has a significant negative effect on both terminal wealth and terminal utility. The effect is most noticeable for the rebalancing strategies that involve frequent rebalancings of the portfolio. For example for the hourly-rebalancing strategy the mean loss of utility is approximately  $1.8300 \times 10^{-2}$  for both transaction cost methods, which is a loss of about 1.8% compared to both the mean utility of the terminal theoretical wealth and the mean utility of the discrete time terminal wealth of the portfolio with no transaction costs. We also observe that the mean loss of utility for the hourly-rebalancing strategy is about twice as large as the 'every 4th hour'-rebalancing strategy, which in turn is about 2.4 times as large as the daily-rebalancing strategy. In fact, it seems like the mean loss of utility approximately is multiplied by a factor  $\sqrt{n}$ , if the rebalancing frequency is multiplied by a factor  $n$ .

The increased mean losses of utility are of course a direct consequence of the added transaction costs. But why is the mean total transaction costs so much higher for the high-frequency rebalancing strategies? Well, the reason is that for a high-frequency rebalancing strategy, frequent but small transactions are needed to rebalance the portfolio frequently, whereas a less frequent rebalancing strategy would imply fewer, but potentially larger transactions. But there is always a chance for a scenario in which the returns on both the risky and the risk-free assets could be nearly equal after a long period of time. Figure 5.15 exhibit such a scenario over a duration of one year. We observe that there is a lot of variation in the risky asset price during the year, but after one year we observe that the risky asset price almost becomes equal to the risk-free asset price. This means that in this particular scenario, the total transaction cost for an investor that rebalances her portfolio at an hourly basis, would amount to a total of  $9.3348 \times 10^{-2}$ , assuming preceding transaction costs and  $\lambda = .03$ . The total transaction cost for an investor that uses an annual-rebalancing strategy, would amount to a total of  $2.9800 \times 10^{-5}$ , since the transaction amount needed to rebalance the portfolio after one year would be very small. Figure 5.16 shows



**Figure 5.15:** Example of price developments of the risky asset (red) and the risk-free asset (blue) that give approximately equal returns after one year.



**Figure 5.16:** Example of price developments of the risky asset (red) and the risk-free asset (blue) that give extremely different returns after one year.

a completely different scenario in which the development of the risky asset is very strong compared to the development of the risk-free asset. A scenario like this would lead to a high total transaction cost even for the annual-rebalancing strategy, because of the in the end huge difference between the return on risky and the risk-free asset. In this specific scenario, the total transaction costs for the hourly-rebalancing strategy are .1419. For the annual-rebalancing strategy the total costs are  $8.1830 \times 10^{-3}$ . For the monthly-rebalancing strategy the total costs are  $1.0053 \times 10^{-2}$ . We see that the difference in total transaction costs between the annual-strategy and the monthly-strategy are relatively small. The reason for this is the relative steady increase of the risky asset price. We can conclude that for a high-frequency rebalancing strategy, frequent but small transactions is the norm, whereas a less frequent rebalancing strategy would imply fewer but potentially larger transaction.

The above reasoning explains the shapes of the distributions of the total transaction costs of each rebalancing strategy, given by the histograms of figure 5.17. An interesting comparison in the context of this example can be found by compar-

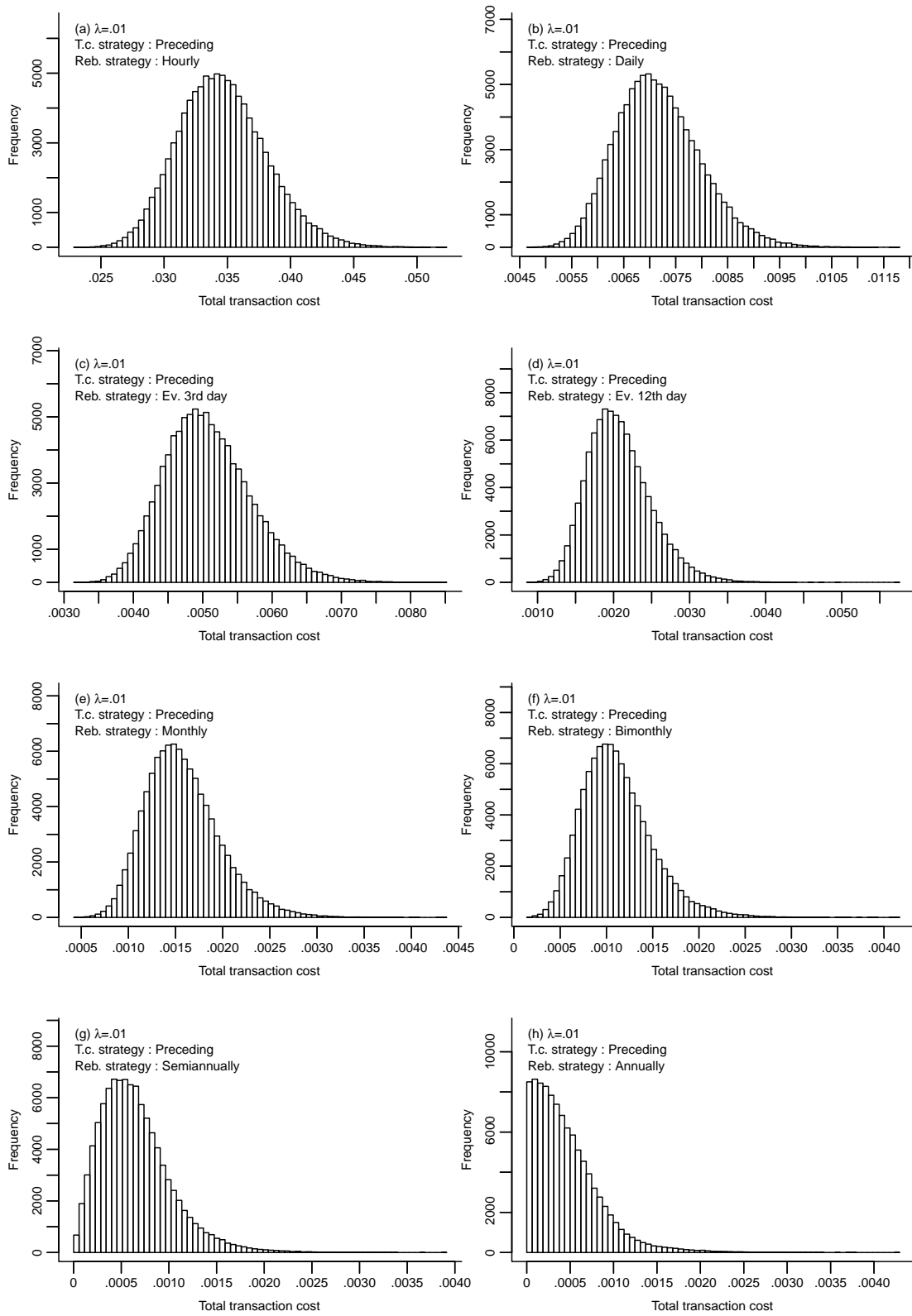


Figure 5.17: Distributions of total transaction costs.

ing the histograms of (e) the monthly-rebalancing strategy and (h) the annual-rebalancing strategy. The range of the total transaction costs of the monthly-strategy is  $[4.1961 \times 10^{-4}, 4.3752 \times 10^{-3}]$ , whereas the range of the annual-rebalancing strategy is  $[4.3000 \times 10^{-8}, 4.2977 \times 10^{-3}]$ , i.e. quite similar ranges. The distributions are however very different. We observe that the distribution of the total transaction costs of the hourly-strategy is similar to a normal distribution, but as the rebalancing frequency is reduced, the bell-shape is gradually "transformed" into a right skewed distribution.

Table 5.7 show the results of the simulations with transaction cost proportion  $\lambda = .02$ . It is clear that with  $\lambda = .02$  the mean total transaction costs of each rebalancing strategy are approximately doubled compared to the previous simulation scenario with  $\lambda = .01$ . This doubling of mean total transaction costs also results in an approximate doubling of the mean loss of wealth. The mean losses of utility are also approximately doubled. There are in other words no surprises in table 5.7.

Table 5.8 show the results of the simulations with transaction cost proportion  $\lambda = .03$ . Compared to the simulation scenario with  $\lambda = .01$ , we observe an approximate tripling of the mean losses of wealth, the mean total transaction costs and the mean losses of utility.

The plots of figure 5.18 summarizes the findings so far. Regarding mean loss of utility, it is clear that the introduction of transaction costs into the simulation model has the biggest influence on the high-frequency rebalancing strategies. We can conclude that there is a positive relationship between rebalancing frequency and mean loss of utility. As for the size of the transaction cost proportion  $\lambda$ , it only serves to proportionally scale the mean losses of utility.

### 5.3.6 Portfolio return and Sharpe ratio

Table 5.9 show the Sharpe ratio of each rebalancing strategy along with other related statistics. As we saw with the mean losses of utility in tables 5.6, 5.7 and 5.8 of the previous subsection, the introduction of transaction costs has the most negative effect on the high-frequency rebalancing strategies. We observe for example that the mean terminal log returns are reduced quite a bit for such rebalancing strategies, compared to the simulations without transaction costs. Using equation (5.10) to calculate the Sharpe ratio, we see that this reduction in log returns has a direct impact on the Sharpe ratio. As a result, the performances of portfolios using high-frequency rebalancing strategies are quite poor according to the Sharpe ratio. We observe for example that the hourly-strategy and the 'every 4th hour'-strategy have got the worst Sharpe ratio scores and are ranked last and second-last according to the ratio. This is a completely different picture



$\lambda = .02$		Sample means				StDev	
		Term. wealth	Total cost	Term. utility	Loss of utility	Loss of utility	
Rebalancing strategy	Hourly	Th	1.0611	-	1.0278	-	-
		No	1.0611	-	1.0278	0	$0.1472 \times 10^{-3}$
		Pre	0.9909	$6.8106 \times 10^{-2}$	0.9915	$3.6288 \times 10^{-2}$	$3.3461 \times 10^{-3}$
		Sub	0.9909	$6.8093 \times 10^{-2}$	0.9915	$3.6281 \times 10^{-2}$	$3.3377 \times 10^{-3}$
	Every 4th hour	Th	1.0597	-	1.0271	-	-
		No	1.0597	-	1.0271	$-.0001 \times 10^{-2}$	$.1892 \times 10^{-3}$
		Pre	1.0241	$3.4611 \times 10^{-2}$	1.0088	$1.8294 \times 10^{-2}$	$1.7144 \times 10^{-3}$
		Sub	1.0241	$3.4604 \times 10^{-2}$	1.0088	$1.8291 \times 10^{-2}$	$1.7063 \times 10^{-3}$
	Daily	Th	1.0609	-	1.0278	-	-
		No	1.0609	-	1.0278	$-.0002 \times 10^{-2}$	$.3619 \times 10^{-3}$
		Pre	1.0462	$1.4291 \times 10^{-2}$	1.0202	$.7514 \times 10^{-2}$	$.7334 \times 10^{-3}$
		Sub	1.0462	$1.4287 \times 10^{-2}$	1.0202	$.7513 \times 10^{-2}$	$.7257 \times 10^{-3}$
	Every 3rd day	Th	1.0606	-	1.0275	-	-
		No	1.0606	-	1.0275	$-.0001 \times 10^{-2}$	$.4980 \times 10^{-3}$
		Pre	1.0501	$1.0126 \times 10^{-2}$	1.0222	$.5317 \times 10^{-2}$	$.5605 \times 10^{-3}$
		Sub	1.0501	$1.0123 \times 10^{-2}$	1.0222	$.5316 \times 10^{-2}$	$.5534 \times 10^{-3}$
	Every 12th day	Th	1.0618	-	1.0282	-	-
		No	1.0618	-	1.0282	$.0007 \times 10^{-2}$	$1.1806 \times 10^{-3}$
		Pre	1.0575	$.4148 \times 10^{-2}$	1.0260	$.2180 \times 10^{-2}$	$.8785 \times 10^{-3}$
		Sub	1.0576	$.4146 \times 10^{-2}$	1.0260	$.2179 \times 10^{-2}$	$.8768 \times 10^{-3}$
	Monthly	Th	1.0607	-	1.0276	-	-
		No	1.0607	-	1.0276	$.0006 \times 10^{-2}$	$1.5531 \times 10^{-3}$
		Pre	1.0575	$.3139 \times 10^{-2}$	1.0260	$.1648 \times 10^{-2}$	$1.2317 \times 10^{-3}$
		Sub	1.0575	$.3138 \times 10^{-2}$	1.0260	$.1648 \times 10^{-2}$	$1.2309 \times 10^{-3}$
	Bimonthly	Th	1.0605	-	1.0275	-	-
		No	1.0605	-	1.0275	$.0008 \times 10^{-2}$	$2.1770 \times 10^{-3}$
		Pre	1.0582	$.2226 \times 10^{-2}$	1.0264	$.1170 \times 10^{-2}$	$1.8471 \times 10^{-3}$
		Sub	1.0582	$.2225 \times 10^{-2}$	1.0264	$.1170 \times 10^{-2}$	$1.8468 \times 10^{-3}$
Semiannually	Th	1.0619	-	1.0282	-	-	
	No	1.0619	-	1.0282	$.0004 \times 10^{-2}$	$3.7852 \times 10^{-3}$	
	Pre	1.0606	$.1301 \times 10^{-2}$	1.0275	$.0674 \times 10^{-2}$	$3.4497 \times 10^{-3}$	
	Sub	1.0606	$.1299 \times 10^{-2}$	1.0275	$.0674 \times 10^{-2}$	$3.4501 \times 10^{-3}$	
Annually	Th	1.0614	-	1.028	-	-	
	No	1.0614	-	1.0279	$.0029 \times 10^{-2}$	$5.3368 \times 10^{-3}$	
	Pre	1.0605	$.0930 \times 10^{-2}$	1.0275	$.0500 \times 10^{-2}$	$5.0048 \times 10^{-3}$	
	Sub	1.0605	$.0928 \times 10^{-2}$	1.0275	$.0500 \times 10^{-2}$	$5.0054 \times 10^{-3}$	

Table 5.7: Mean losses of utility and other statistics,  $\lambda = .02$ .

$\lambda = .03$		Sample means				StDev	
		Term. wealth	Total cost	Term. utility	Loss of utility	Loss of utility	
Rebalancing strategy	Hourly	Th	1.0614	-	1.0280	-	-
		No	1.0614	-	1.0280	0	$.1474 \times 10^{-3}$
		Pre	.9578	$10.0463 \times 10^{-2}$	.9740	$5.3970 \times 10^{-2}$	$4.9752 \times 10^{-3}$
		Sub	.9579	$10.0420 \times 10^{-2}$	.9740	$5.3948 \times 10^{-2}$	$4.9559 \times 10^{-3}$
	Every 4th hour	Th	1.0611	-	1.0278	-	-
		No	1.0611	-	1.0278	0	$.1896 \times 10^{-3}$
		Pre	1.0080	$5.1514 \times 10^{-2}$	1.0005	$2.7345 \times 10^{-2}$	$2.5674 \times 10^{-3}$
		Sub	1.0080	$5.1491 \times 10^{-2}$	1.0005	$2.7334 \times 10^{-2}$	$2.5489 \times 10^{-3}$
	Daily	Th	1.0602	-	1.0274	-	-
		No	1.0602	-	1.0274	0	$.3616 \times 10^{-3}$
		Pre	1.0382	$2.1355 \times 10^{-2}$	1.0161	$1.1250 \times 10^{-2}$	$1.0941 \times 10^{-3}$
		Sub	1.0382	$2.1343 \times 10^{-2}$	1.0162	$1.1245 \times 10^{-2}$	$1.0766 \times 10^{-3}$
	Every 3rd day	Th	1.0602	-	1.0274	-	-
		No	1.0602	-	1.0274	$-.0001 \times 10^{-2}$	$.4937 \times 10^{-3}$
		Pre	1.0446	$1.5144 \times 10^{-2}$	1.0194	$.7965 \times 10^{-2}$	$.7967 \times 10^{-3}$
		Sub	1.0446	$1.5135 \times 10^{-2}$	1.0194	$.7962 \times 10^{-2}$	$.7794 \times 10^{-3}$
	Every 12th day	Th	1.0610	-	1.0277	-	-
		No	1.0610	-	1.0277	$-.0005 \times 10^{-2}$	$1.1791 \times 10^{-3}$
		Pre	1.0546	$.6228 \times 10^{-2}$	1.0245	$.3258 \times 10^{-2}$	$.7686 \times 10^{-3}$
		Sub	1.0546	$.6223 \times 10^{-2}$	1.0245	$.3257 \times 10^{-2}$	$.7619 \times 10^{-3}$
	Monthly	Th	1.0609	-	1.0277	-	-
		No	1.0609	-	1.0277	$-.0005 \times 10^{-2}$	$1.5459 \times 10^{-3}$
		Pre	1.0561	$.4718 \times 10^{-2}$	1.0253	$.2463 \times 10^{-2}$	$1.0857 \times 10^{-3}$
		Sub	1.0561	$.4713 \times 10^{-2}$	1.0253	$.2462 \times 10^{-2}$	$1.0823 \times 10^{-3}$
	Bimonthly	Th	1.0613	-	1.0279	-	-
		No	1.0613	-	1.0279	0	$2.1959 \times 10^{-3}$
		Pre	1.0579	$.3345 \times 10^{-2}$	1.0262	$.1744 \times 10^{-2}$	$1.7048 \times 10^{-3}$
		Sub	1.0579	$.3340 \times 10^{-2}$	1.0262	$.1744 \times 10^{-2}$	$1.7039 \times 10^{-3}$
Seminannually	Th	1.0613	-	1.0279	-	-	
	No	1.0614	-	1.0279	$-.0021 \times 10^{-2}$	$3.8273 \times 10^{-3}$	
	Pre	1.0594	$.1957 \times 10^{-2}$	1.0269	$.0987 \times 10^{-2}$	$3.3234 \times 10^{-3}$	
	Sub	1.0594	$.1953 \times 10^{-2}$	1.0269	$.0987 \times 10^{-2}$	$3.3240 \times 10^{-3}$	
Annually	Th	1.0606	-	1.0276	$0 \times 10^{-2}$	$0 \times 10^{-3}$	
	No	1.0607	-	1.0276	$-.0005 \times 10^{-2}$	$5.3980 \times 10^{-3}$	
	Pre	1.0593	$.1400 \times 10^{-2}$	1.0269	$.0705 \times 10^{-2}$	$4.8986 \times 10^{-3}$	
	Sub	1.0593	$.1397 \times 10^{-2}$	1.0269	$.0705 \times 10^{-2}$	$4.8998 \times 10^{-3}$	

Table 5.8: Mean losses of utility and other statistics,  $\lambda = .03$ .

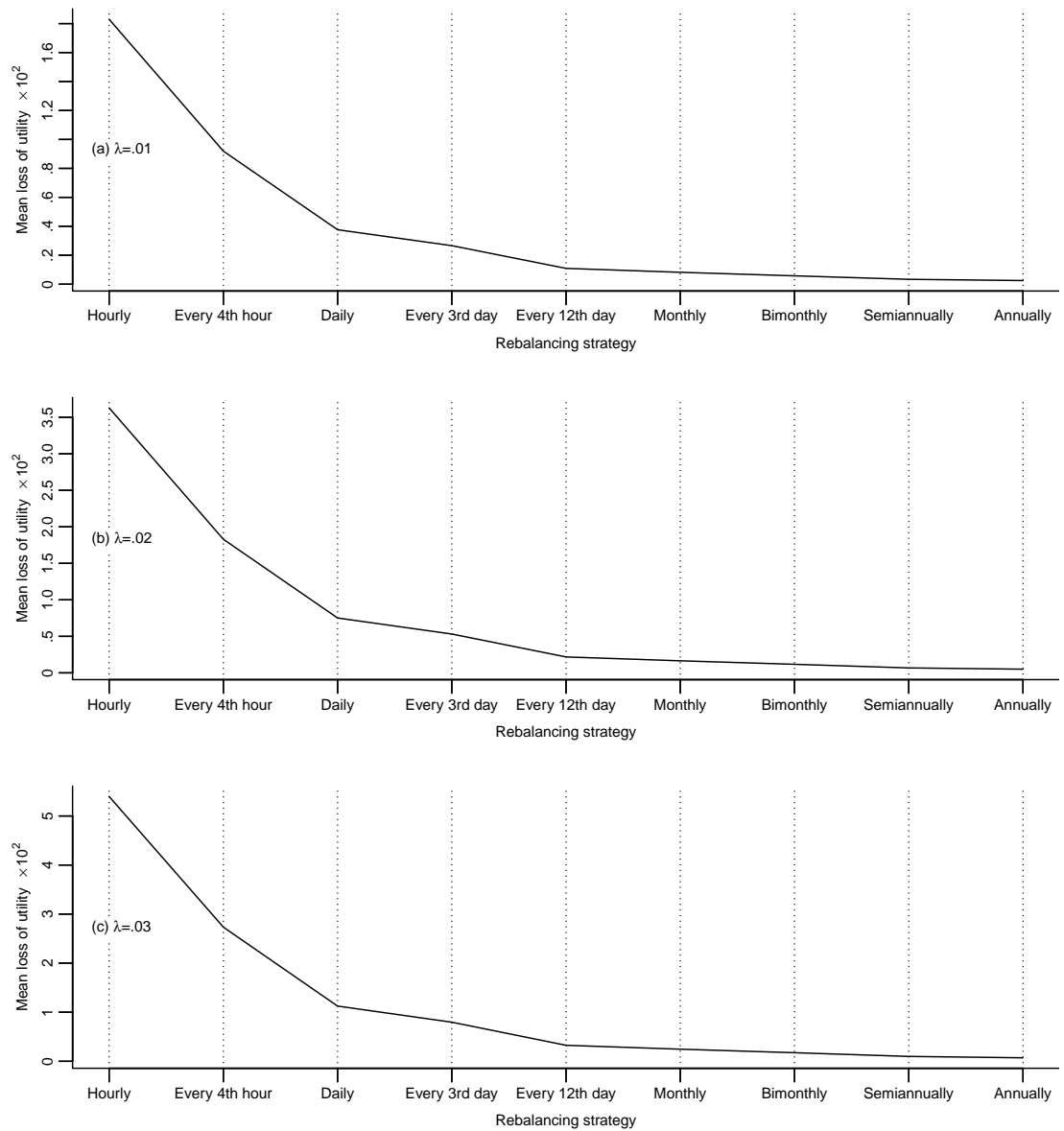


Figure 5.18: Mean losses of utility: (a)  $\lambda = .01$ , (b)  $\lambda = .02$  and (c)  $\lambda = .03$ .

$\lambda = .01$		Sample means			Vol.	Corr.	Rank	
Simulation model		Terminal log return	Vol.	Sharpe ratio	of vol.			
Rebalancing strategy	Hourly	Th	$4.4128 \times 10^{-2}$	.1728	-.0045	$1.5629 \times 10^{-3}$	-.0004	9
		No	$4.4129 \times 10^{-2}$	.1728	-.0044	$1.5630 \times 10^{-3}$	-.0051	
		Pre	$.9934 \times 10^{-2}$	.1728	-.2023	$1.5631 \times 10^{-3}$	-.0105	
		Sub	$.9935 \times 10^{-2}$	.1728	-.2023	$1.5631 \times 10^{-3}$	-.0105	
	Every 4th hour	Th	$4.4074 \times 10^{-2}$	.1728	-.0048	$1.5696 \times 10^{-3}$	.0004	8
		No	$4.4074 \times 10^{-2}$	.1728	-.0048	$1.5696 \times 10^{-3}$	-.0021	
		Pre	$2.6977 \times 10^{-2}$	.1728	-.1037	$1.5697 \times 10^{-3}$	-.0048	
		Sub	$2.6978 \times 10^{-2}$	.1728	-.1037	$1.5697 \times 10^{-3}$	-.0048	
	Daily	Th	$4.4093 \times 10^{-2}$	.1728	-.0047	$1.5713 \times 10^{-3}$	.0006	6
		No	$4.4093 \times 10^{-2}$	.1728	-.0048	$1.5716 \times 10^{-3}$	.0128	
		Pre	$3.7113 \times 10^{-2}$	.1728	-.0452	$1.5716 \times 10^{-3}$	.0117	
		Sub	$3.7113 \times 10^{-2}$	.1728	-.0452	$1.5716 \times 10^{-3}$	.0117	
	Every 3rd day	Th	$4.4313 \times 10^{-2}$	.1728	-.0034	$1.5697 \times 10^{-3}$	.0004	5
		No	$4.4315 \times 10^{-2}$	.1728	-.0037	$1.5709 \times 10^{-3}$	.0304	
		Pre	$3.9379 \times 10^{-2}$	.1728	-.0322	$1.5710 \times 10^{-3}$	.0297	
		Sub	$3.9379 \times 10^{-2}$	.1728	-.0322	$1.5710 \times 10^{-3}$	.0297	
	Every 12th day	Th	$4.3583 \times 10^{-2}$	.1728	-.0077	$1.5706 \times 10^{-3}$	.0034	3
		No	$4.3576 \times 10^{-2}$	.1728	-.0096	$1.6162 \times 10^{-3}$	.2041	
		Pre	$4.1562 \times 10^{-2}$	.1728	-.0212	$1.6162 \times 10^{-3}$	.2038	
		Sub	$4.1562 \times 10^{-2}$	.1728	-.0212	$1.6162 \times 10^{-3}$	.2038	
Monthly	Th	$4.4947 \times 10^{-2}$	.1728	.0002	$1.5784 \times 10^{-3}$	.0060	1	
	No	$4.4952 \times 10^{-2}$	.1727	-.0031	$1.7153 \times 10^{-3}$	.3406		
	Pre	$4.3428 \times 10^{-2}$	.1727	-.0119	$1.7153 \times 10^{-3}$	.3404		
	Sub	$4.3428 \times 10^{-2}$	.1727	-.0119	$1.7153 \times 10^{-3}$	.3404		
Bimonthly	Th	$4.3660 \times 10^{-2}$	.1728	-.0072	$1.5709 \times 10^{-3}$	-.0003	2	
	No	$4.3671 \times 10^{-2}$	.1727	-.0138	$2.0635 \times 10^{-3}$	.5602		
	Pre	$4.2594 \times 10^{-2}$	.1727	-.0201	$2.0634 \times 10^{-3}$	.5601		
	Sub	$4.2593 \times 10^{-2}$	.1727	-.0201	$2.0634 \times 10^{-3}$	.5601		
Seminannualy	Th	$4.4203 \times 10^{-2}$	.1728	-.0040	$1.5656 \times 10^{-3}$	.0009	4	
	No	$4.4217 \times 10^{-2}$	.1726	-.0241	$4.3086 \times 10^{-3}$	.8052		
	Pre	$4.3594 \times 10^{-2}$	.1726	-.0277	$4.3085 \times 10^{-3}$	.8052		
	Sub	$4.3594 \times 10^{-2}$	.1726	-.0277	$4.3085 \times 10^{-3}$	.8052		
Annually	Th	$4.4210 \times 10^{-2}$	.1728	-.0040	$1.5678 \times 10^{-3}$	.0036	7	
	No	$4.4218 \times 10^{-2}$	.1723	-.0443	$8.1598 \times 10^{-3}$	.8474		
	Pre	$4.3780 \times 10^{-2}$	.1723	-.0469	$8.1595 \times 10^{-3}$	.8476		
	Sub	$4.3779 \times 10^{-2}$	.1723	-.0469	$8.1595 \times 10^{-3}$	.8476		

**Table 5.9:** The Sharpe ratio versus rebalancing strategy and other statistics,  $\lambda = .01$ .

compared to what we saw in connection with the simulations without transaction costs, where the hourly-strategy was ranked first. But similarly to what we saw in connection with the simulations without transaction costs, the Sharpe ratio is also affected by the spread of the volatilities of the log returns, that is the volatility of the volatilities, and the correlation between log returns and volatilities. And as before, the annual-rebalancing strategy has the highest volatility of volatilities and correlation, which means that this strategy, according to the Sharpe ratio, ranks seventh, even though the mean total transaction costs for this strategy were lowest. In connection with the simulations assuming no transaction costs, this strategy ranked last. The best rebalancing strategy, assuming  $\lambda = .01$ , is the monthly-strategy. Even though this strategy has higher mean total transaction costs than the bimonthly, semiannual and annual strategy, it has, according to the Sharpe ratio, lower risk. The monthly-strategy has in other words the best trade-off between return and risk.

Table 5.10 show that increasing the transaction cost proportion  $\lambda$  from .01 to .02 approximately doubles the differences between the theoretical log returns and the log returns of the portfolio models with transaction costs. This obviously causes the Sharpe ratios to decrease compared to the Sharpe ratios of the simulated portfolios with  $\lambda = .01$ . If we use the Sharpe ratios of the simulated portfolios with no transaction costs (the 'No' category) as baseline, increasing  $\lambda$  from .01 to .02 causes a doubling of the Sharpe ratios in the negative direction. This time, according to the Sharpe ratio, the 'every 12th day'-strategy performs best, but if we use the Sharpe ratio of the 'Th'-category as reference point we find that the bimonthly-strategy performs best. An increase in transaction cost proportion should disfavour high-frequency rebalancing strategies relatively more than rebalancing strategies with lower rebalancing frequencies, because of the bigger increase in total transaction costs of the high-frequency rebalancing strategies. This is just what we see, comparing the Sharpe ratios of the first batch of simulations with  $\lambda = .01$  with the second batch of simulations with  $\lambda = .02$ . The best performing rebalancing strategy moves towards a rebalancing strategy with fewer rebalancings when the transaction cost proportion increases so to speak.

Table 5.11 show the mean Sharpe ratios of the simulated portfolios when we assume  $\lambda = .03$ . The discussion in this situation is basically the same as for the previous discussion, where we assumed  $\lambda = .02$ , except that all references to "double" and "doubling" must be changed with "triple" and "tripling". With  $\lambda = .03$ , according to the Sharpe ratio, the bimonthly-strategy performs best even when we use the mean Sharpe ratios of the 'Th'-category as the point of reference.

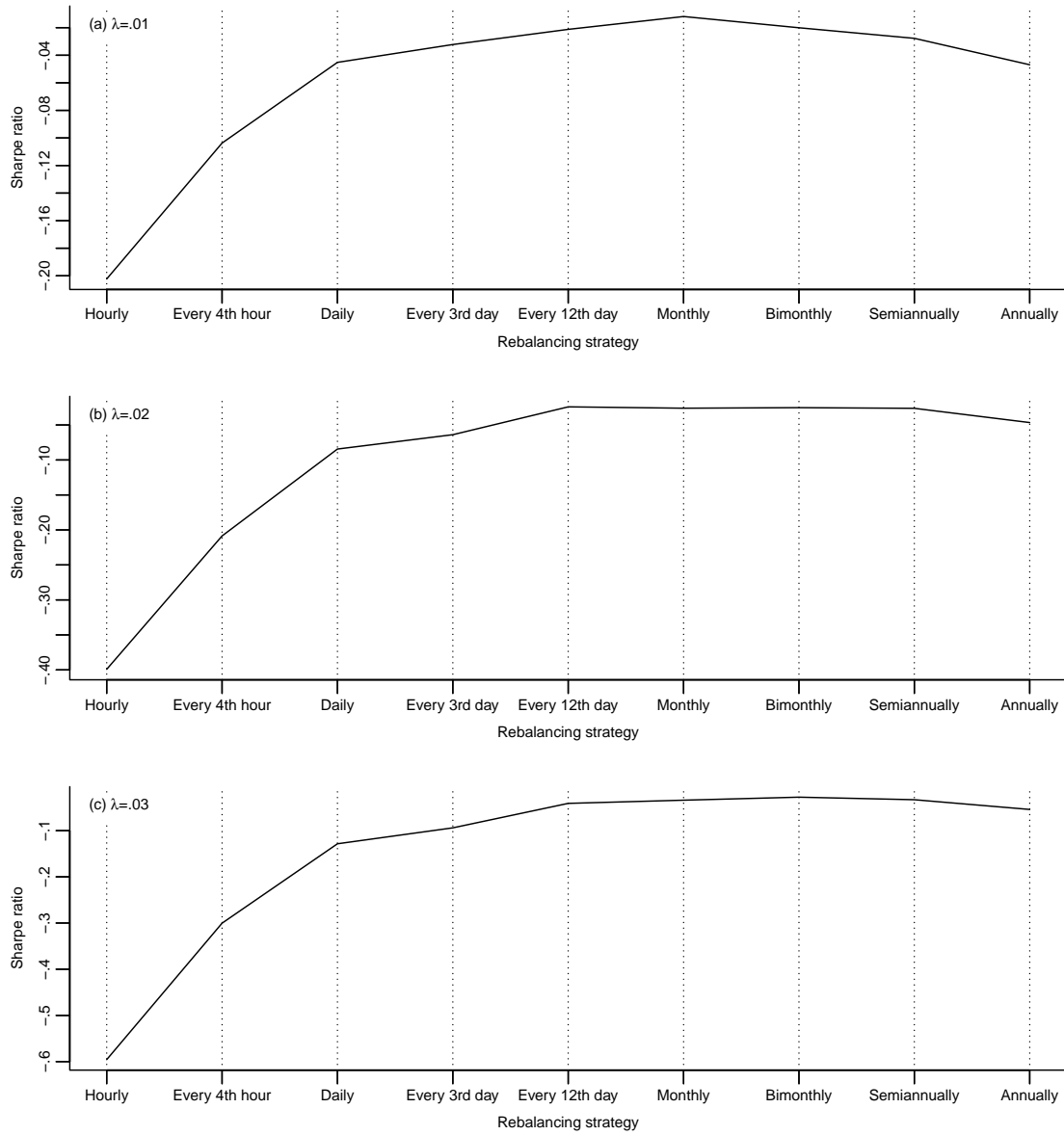
The plots of figure 5.19 show each rebalancing strategy plotted against its corresponding Sharpe ratio for (a)  $\lambda = .01$ , (b)  $\lambda = .02$  and (c)  $\lambda = .03$ . The plots show what we already have discussed, that the Sharpe ratios of high-frequency

$\lambda = .02$		Sample means			Vol.	Corr.	Rank
Simulation model		Terminal log return	Vol.	Sharpe ratio	of vol.		
Hourly	Th	$4.4315 \times 10^{-2}$	.1728	-.0034	$1.5704 \times 10^{-3}$	-.0034	9
	No	$4.4315 \times 10^{-2}$	.1728	-.0033	$1.5705 \times 10^{-3}$	-.0081	
	Pre	$-2.4085 \times 10^{-2}$	.1728	-.3992	$1.5706 \times 10^{-3}$	-.0190	
	Sub	$-2.4074 \times 10^{-2}$	.1728	-.3991	$1.5707 \times 10^{-3}$	-.0190	
Every 4th hour	Th	$4.3043 \times 10^{-2}$	.1728	-.0107	$1.5719 \times 10^{-3}$	-.0041	8
	No	$4.3044 \times 10^{-2}$	.1728	-.0107	$1.5720 \times 10^{-3}$	-.0066	
	Pre	$.8844 \times 10^{-2}$	.1728	-.2086	$1.5721 \times 10^{-3}$	-.0121	
	Sub	$.8849 \times 10^{-2}$	.1728	-.2086	$1.5722 \times 10^{-3}$	-.0121	
Daily	Th	$4.4297 \times 10^{-2}$	.1728	-.0035	$1.5729 \times 10^{-3}$	.0007	7
	No	$4.4301 \times 10^{-2}$	.1728	-.0036	$1.5732 \times 10^{-3}$	.0128	
	Pre	$3.0335 \times 10^{-2}$	.1728	-.0844	$1.5733 \times 10^{-3}$	.0106	
	Sub	$3.0336 \times 10^{-2}$	.1728	-.0844	$1.5733 \times 10^{-3}$	.0106	
Every 3rd day	Th	$4.3775 \times 10^{-2}$	.1728	-.0065	$1.5743 \times 10^{-3}$	-.0023	6
	No	$4.3777 \times 10^{-2}$	.1728	-.0068	$1.5753 \times 10^{-3}$	.0277	
	Pre	$3.3903 \times 10^{-2}$	.1728	-.0639	$1.5753 \times 10^{-3}$	.0261	
	Sub	$3.3903 \times 10^{-2}$	.1728	-.0639	$1.5753 \times 10^{-3}$	.0261	
Every 12th day	Th	$4.5095 \times 10^{-2}$	.1728	.0011	$1.5722 \times 10^{-3}$	-.0021	1
	No	$4.5084 \times 10^{-2}$	.1728	-.0008	$1.6168 \times 10^{-3}$	.1987	
	Pre	$4.1059 \times 10^{-2}$	.1728	-.0241	$1.6169 \times 10^{-3}$	.1981	
	Sub	$4.1058 \times 10^{-2}$	.1728	-.0241	$1.6169 \times 10^{-3}$	.1981	
Monthly	Th	$4.4022 \times 10^{-2}$	.1728	-.0051	$1.5743 \times 10^{-3}$	-.0011	3
	No	$4.4015 \times 10^{-2}$	.1727	-.0084	$1.7072 \times 10^{-3}$	.3338	
	Pre	$4.0971 \times 10^{-2}$	.1727	-.0260	$1.7072 \times 10^{-3}$	.3334	
	Sub	$4.0970 \times 10^{-2}$	.1727	-.0260	$1.7072 \times 10^{-3}$	.3333	
Bimonthly	Th	$4.3852 \times 10^{-2}$	.1728	-.0061	$1.5716 \times 10^{-3}$	.0057	2
	No	$4.3844 \times 10^{-2}$	.1727	-.0128	$2.0619 \times 10^{-3}$	.5604	
	Pre	$4.1691 \times 10^{-2}$	.1727	-.0253	$2.0620 \times 10^{-3}$	.5602	
	Sub	$4.1690 \times 10^{-2}$	.1727	-.0253	$2.0619 \times 10^{-3}$	.5601	
Seminannually	Th	$4.5077 \times 10^{-2}$	.1728	.0011	$1.5696 \times 10^{-3}$	-.0045	4
	No	$4.5081 \times 10^{-2}$	.1726	-.0190	$4.2968 \times 10^{-3}$	.8031	
	Pre	$4.3838 \times 10^{-2}$	.1726	-.0262	$4.2963 \times 10^{-3}$	.8032	
	Sub	$4.3837 \times 10^{-2}$	.1726	-.0262	$4.2962 \times 10^{-3}$	.8032	
Annually	Th	$4.4697 \times 10^{-2}$	.1728	-.0012	$1.5723 \times 10^{-3}$	.0012	5
	No	$4.4675 \times 10^{-2}$	.1723	-.0414	$8.1477 \times 10^{-3}$	.8450	
	Pre	$4.3800 \times 10^{-2}$	.1723	-.0465	$8.1464 \times 10^{-3}$	.8453	
	Sub	$4.3798 \times 10^{-2}$	.1723	-.0466	$8.1463 \times 10^{-3}$	.8453	

**Table 5.10:** The Sharpe ratio versus rebalancing strategy and other statistics,  $\lambda = .02$ .

$\lambda = .03$		Sample means			Vol.	Corr.	Rank	
Simulation model		Terminal log return	Vol.	Sharpe ratio	of vol.			
Rebalancing strategy	Hourly	Th	$4.4601 \times 10^{-2}$	.1728	-.0017	$1.5722 \times 10^{-3}$	.0008	9
		No	$4.4601 \times 10^{-2}$	.1728	-.0017	$1.5723 \times 10^{-3}$	-.0039	
		Pre	$-5.8024 \times 10^{-2}$	.1728	-.5957	$1.5725 \times 10^{-3}$	-.0201	
		Sub	$-5.7984 \times 10^{-2}$	.1728	-.5953	$1.5728 \times 10^{-3}$	-.0201	
	Every 4th hour	Th	$4.4315 \times 10^{-2}$	.1728	-.0034	$1.5702 \times 10^{-3}$	.0038	8
		No	$4.4314 \times 10^{-2}$	.1728	-.0034	$1.5702 \times 10^{-3}$	.0013	
		Pre	$-.6996 \times 10^{-2}$	.1728	-.3003	$1.5703 \times 10^{-3}$	-.0068	
		Sub	$-.6978 \times 10^{-2}$	.1728	-.3002	$1.5704 \times 10^{-3}$	-.0068	
	Daily	Th	$4.3672 \times 10^{-2}$	.1728	-.0071	$1.5694 \times 10^{-3}$	.0033	7
		No	$4.3672 \times 10^{-2}$	.1728	-.0072	$1.5696 \times 10^{-3}$	.0154	
		Pre	$2.2724 \times 10^{-2}$	.1728	-.1285	$1.5698 \times 10^{-3}$	.0122	
		Sub	$2.2729 \times 10^{-2}$	.1728	-.1284	$1.5698 \times 10^{-3}$	.0122	
	Every 3rd day	Th	$4.3517 \times 10^{-2}$	.1728	-.0080	$1.5726 \times 10^{-3}$	-.0009	6
		No	$4.3518 \times 10^{-2}$	.1728	-.0083	$1.5738 \times 10^{-3}$	.0289	
		Pre	$2.8708 \times 10^{-2}$	.1728	-.0940	$1.5740 \times 10^{-3}$	.0265	
		Sub	$2.8711 \times 10^{-2}$	.1728	-.0939	$1.5740 \times 10^{-3}$	.0265	
	Every 12th day	Th	$4.4182 \times 10^{-2}$	.1728	-.0041	$1.5728 \times 10^{-3}$	-.0025	4
		No	$4.4192 \times 10^{-2}$	.1728	-.0060	$1.6176 \times 10^{-3}$	.1991	
		Pre	$3.8142 \times 10^{-2}$	.1728	-.0410	$1.6178 \times 10^{-3}$	.1982	
		Sub	$3.8142 \times 10^{-2}$	.1728	-.0410	$1.6178 \times 10^{-3}$	.1982	
	Monthly	Th	$4.4146 \times 10^{-2}$	.1728	-.0044	$1.5696 \times 10^{-3}$	.0001	3
		No	$4.4159 \times 10^{-2}$	.1727	-.0076	$1.7081 \times 10^{-3}$	.3372	
		Pre	$3.9585 \times 10^{-2}$	.1728	-.0341	$1.7084 \times 10^{-3}$	.3366	
		Sub	$3.9584 \times 10^{-2}$	.1728	-.0341	$1.7084 \times 10^{-3}$	.3365	
Bimonthly	Th	$4.4515 \times 10^{-2}$	.1728	-.0022	$1.5757 \times 10^{-3}$	-.0066	1	
	No	$4.4518 \times 10^{-2}$	.1727	-.0088	$2.0600 \times 10^{-3}$	.5552		
	Pre	$4.1287 \times 10^{-2}$	.1727	-.0275	$2.0600 \times 10^{-3}$	.5549		
	Sub	$4.1286 \times 10^{-2}$	.1727	-.0276	$2.0599 \times 10^{-3}$	.5548		
Seminannualy	Th	$4.4488 \times 10^{-2}$	.1728	-.0024	$1.5700 \times 10^{-3}$	.0012	2	
	No	$4.4546 \times 10^{-2}$	.1726	-.0222	$4.3092 \times 10^{-3}$	.8052		
	Pre	$4.2677 \times 10^{-2}$	.1726	-.0331	$4.3080 \times 10^{-3}$	.8053		
	Sub	$4.2675 \times 10^{-2}$	.1726	-.0331	$4.3077 \times 10^{-3}$	.8052		
Annually	Th	$4.3852 \times 10^{-2}$	.1728	-.0061	$1.5683 \times 10^{-3}$	.0036	5	
	No	$4.3892 \times 10^{-2}$	.1722	-.0465	$8.2000 \times 10^{-3}$	.8455		
	Pre	$4.2574 \times 10^{-2}$	.1722	-.0542	$8.1970 \times 10^{-3}$	.8459		
	Sub	$4.2572 \times 10^{-2}$	.1722	-.0542	$8.1965 \times 10^{-3}$	.8459		

**Table 5.11:** The Sharpe ratio versus rebalancing strategy and other statistics,  $\lambda = .03$ .



**Figure 5.19:** Sharpe ratios versus rebalancing strategies: (a)  $\lambda = .01$ , (b)  $\lambda = .02$  and (c)  $\lambda = .03$ .



rebalancing strategies are punished by the high transaction costs of such strategies. The annual rebalancing strategy are punished by higher risk. The 'every 12th day', the monthly, the bimonthly and the semiannual rebalancing strategy are the best strategies according to the Sharpe ratio.

## 5.4 Simulation with stochastic volatility

### 5.4.1 Introduction

The first simulation model, that is simulation model I, was a rather unrealistic model. In section 5.3 we increased the complexity and the realism of the model by introducing transaction costs. One shortcoming of simulation models I, II and III is the assumption of constant volatility. As the plots of short-term volatilities of figure 4.4 of section 4.3 show, this assumption is rather unrealistic. In this section we will further increase the complexity and realism of the simulation model by assuming stochastic volatility. The new simulation model will, perhaps not so surprisingly, be dubbed simulation model IV.

### 5.4.2 Stochastic volatility

There exists many different models for modelling stochastic volatility. One class of models are driven by Brownian motion(s), such as the CEV model, the SABR volatility model, the GARCH model, the 3/2 model, the Chen model and other models. Another class of stochastic volatility models are the Levy driven models. We will in this thesis use a Brownian motion driven stochastic volatility model, namely the Heston model. The definition of this model is stated in section 2.1. The SDE (2.4), which describes the dynamics of the volatility, is, as stated in that section, a so-called CIR-process. One important property of the CIR-process is mean reversion, which means that in the long run, the process tends to drift towards its long-term mean. This mean reversion tendency is in accordance with evidence from equity markets [6].

A standard method of simulating a Heston stochastic volatility process is through its Euler approximation. Assuming equidistant time increments, the Euler approximation of the SDE (2.4) is straightforwardly

$$\nu_{k+1} = \nu_k + \kappa(\theta - \nu_k)\delta + \xi\sqrt{\nu_k}\Delta B_k^\nu.$$

With the introduction of stochastic volatility we need to reconsider the Merton ratio. Remember that the optimal allocation strategy given by the Merton ratio

(3.10) has so far been constant. We now have to take into consideration that the volatility will vary with time when we calculate the Merton ratio, so the optimal allocation strategy has to be redefined as

$$u_t^* = \frac{\mu - r}{\nu_t(1 - \gamma)}. \quad (5.16)$$

We see that the optimal allocation strategy now indirectly has become a stochastic quantity. We also see that if the volatility increases, the investor will invest less in the risky asset and vice versa, as it should be reflecting the risk-aversion of the investor.

Compared to simulation model II and III, the transaction quantity  $Q_k$  at time  $t_k$ , will in simulation model IV, as a consequence of the introduction of stochastic volatility, be slightly altered. We will in the new simulation model only consider preceding transaction costs, not subsequent transaction costs. As argued for before, preceding transaction costs reflect the idea of a rebalanced portfolio at each rebalancing time point more accurately and, as the simulations of section 5.3 did show, the difference in total transaction costs between the two transaction cost methods is minute. Another consequence of the introduction of stochastic volatility is that the direction of the transaction between the risky asset investment and risk-free asset investment at rebalancing time points, is no longer only determined by the difference in returns (5.12) between the risky asset and the risk-free asset since the previous rebalancing time point. The optimal allocation strategy at rebalancing time points must now also be taken into consideration. Assume that  $t_k$  is a rebalancing time point. One possible scenario is for example that the return on the risky asset investment since the previous rebalancing time point  $t_k^*$  is higher than the return on the risk-free asset investment. Such a scenario would in simulation model I, II and III imply a reduction of the risky asset investment and a corresponding increase of the risk-free asset investment (before the deduction of transaction costs) at time  $t_k$ . With stochastic volatility, a high value of  $u_k$  could require a reverse transaction even though the return on the risky asset investment is higher. To determine the direction of the transaction in simulation model IV, we need to replace (5.12) with

$$D_k = (1 - u_k^*)u_{k^*}^* \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma_{j+1}\Delta B_j^S) - u_k^*(1 - u_{k^*}^*)(1 + r\delta)^{k-k^*}$$

Similar to the previous calculations of the transaction quantity, for the portfolio

to become rebalanced, we require that

$$u_k^* = \frac{\tilde{V}_k'^S - Q_k}{\tilde{V}_k},$$

$$1 - u_k^* = \begin{cases} \frac{\tilde{V}_k'^R + Q_k - \lambda Q_k}{\tilde{V}_k}, & D_k \geq 0 \\ \frac{\tilde{V}_k'^R + Q_k + \lambda Q_k}{\tilde{V}_k}, & D_k < 0. \end{cases}$$

The solution with respect to  $Q_k$  is

$$Q_k = \begin{cases} \frac{(1 - u_k^*)\tilde{V}_k'^S - u_k^*\tilde{V}_k'^R}{1 - \lambda u_k^*}, & D_k \geq 0 \\ \frac{(1 - u_k^*)\tilde{V}_k'^S - u_k^*\tilde{V}_k'^R}{1 + \lambda u_k^*}, & D_k < 0. \end{cases}$$

It is clear that the stochastic volatility induces extra variability into the simulation model. The question is, how will this added variability effect the outcomes of the simulations?

### 5.4.3 Simulation model IV

#### Simulation model IV

Transaction costs: Preceding

Volatility: Stochastic

$$\begin{aligned}\sigma_k &= \sqrt{\nu_k} = \sqrt{\nu_{k-1} + \kappa(\theta - \nu_{k-1})\delta + \xi\sqrt{\nu_{k-1}}\Delta B_{k-1}^\nu} \\ u_k^* &= \frac{\mu - r}{\sigma_k^2(1 - \gamma)} \\ \tilde{V}_k'^S &= u_k^* \tilde{V}_{k^*} \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma_{j+1}\Delta B_j^S) \\ \tilde{V}_k'^R &= (1 - u_{k^*}^*) \tilde{V}_{k^*} (1 + r\delta)^{k-k^*} \\ D_k &= (1 - u_k^*) u_{k^*}^* \prod_{j=k^*}^{k-1} (1 + \mu\delta + \sigma_{j+1}\Delta B_j^S) - u_k^* (1 - u_{k^*}^*) (1 + r\delta)^{k-k^*} \\ Q_k &= \begin{cases} \frac{(1 - u_k^*) \tilde{V}_k'^S - u_k^* \tilde{V}_k'^R}{1 - \lambda u_k^*}, & D_k \geq 0 \\ \frac{(1 - u_k^*) \tilde{V}_k'^S - u_k^* \tilde{V}_k'^R}{1 + \lambda u_k^*}, & D_k < 0 \end{cases} \\ \tilde{V}_k^S &= \begin{cases} \tilde{V}_k'^S - Q_k, & t_k \in \mathcal{T}^{\text{reb}} \\ \tilde{V}_k'^S, & \text{otherwise} \end{cases} \\ \tilde{V}_k^R &= \begin{cases} \tilde{V}_k'^R + Q_k - \lambda|Q_k|, & t_k \in \mathcal{T}^{\text{reb}} \\ \tilde{V}_k'^R, & \text{otherwise} \end{cases} \\ \tilde{V}_k &= \tilde{V}_k^S + \tilde{V}_k^R.\end{aligned}$$

The above framed equations show the required equations of the stochastic volatility simulation model, namely simulation model IV.

### 5.4.4 Implementation

For the actual simulations of the volatility we will use the estimates found through the linear regression estimation of section 4.3. These estimates are summarized in table 5.12. As for the estimates of  $\mu$ ,  $r$  and  $\gamma$ , we use the same parameter estimates as we used in the thesis so far, that is the parameter estimates of table 4.1. Due to the increased complexity and, hence, slower run time of the simulation model

Parameter	Estimate
$\nu_0$	$6.6105 \times 10^{-2}$
$\kappa$	320.1192
$\theta$	$6.7456 \times 10^{-2}$
$\xi$	.0590
$\rho$	$2.6706 \times 10^{-2}$

**Table 5.12:** The parameter estimations of the Heston model.

IV, a total 50,000 simulations of each combination of transaction cost proportion and rebalancing strategy, that is a total of 36 combinations (including  $\lambda = 0$ , that is no transaction costs) were run.

So far in this thesis we have used the theoretical portfolio value (5.2) as the point of reference when measuring the loss of wealth and the loss of utility. The introduction of stochastic volatility gives a slightly modified expression for the SDE (3.2) of the portfolio value process. The new SDE of the portfolio value process is

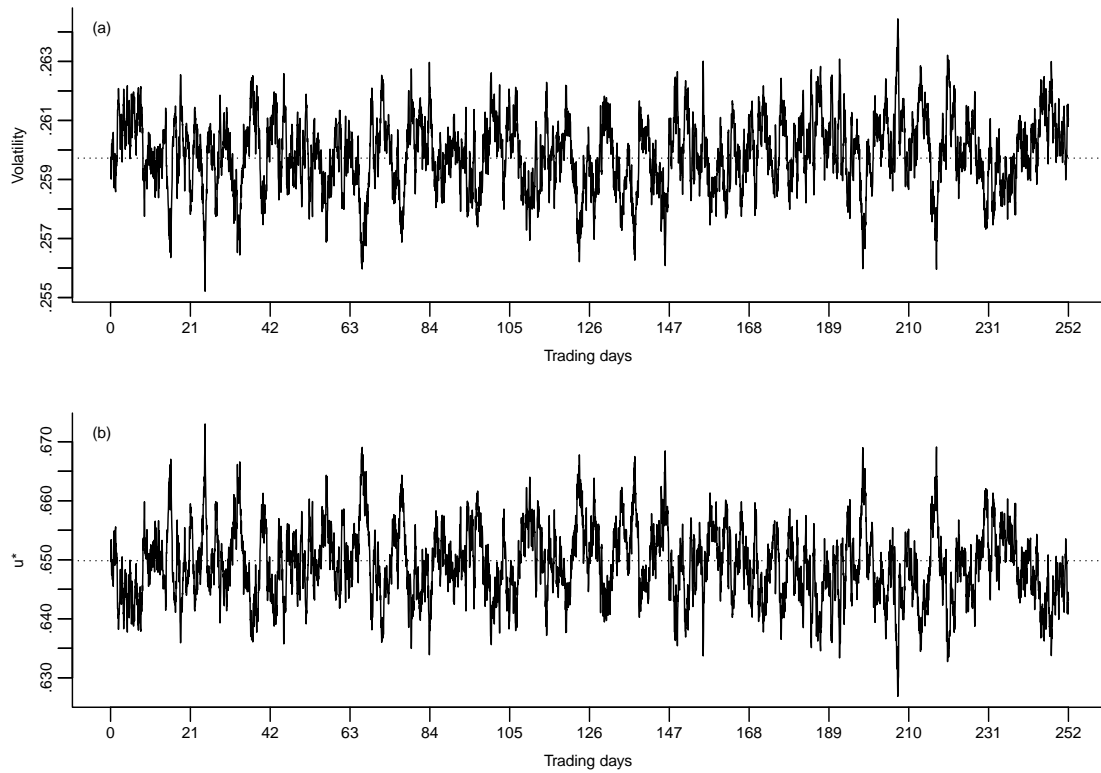
$$dV_t = (\mu u_t + r(1 - u_t))V_t dt + \sqrt{\nu_t} u_t V_t dB_t. \quad (5.17)$$

Since a closed form solution of (5.17) similar to (5.2) doesn't exist, it is natural to use a different point of reference. One such point of reference is the simulated discrete time portfolio with constant volatility, that is simulation model III or I, depending on whether we assume preceding transaction costs or not. A natural choice of the constant volatility of the new reference portfolio is the square root of the long-term mean  $\theta$ . The new constant volatility also implies a modified value of the optimal allocation strategy  $u^*$ . Using the estimate of  $\theta$ , that is  $6.7456 \times 10^{-2}$  along with the estimates of  $\mu$ ,  $r$  and  $\gamma$  of table 4.1 yields  $u^* = .6498$ .

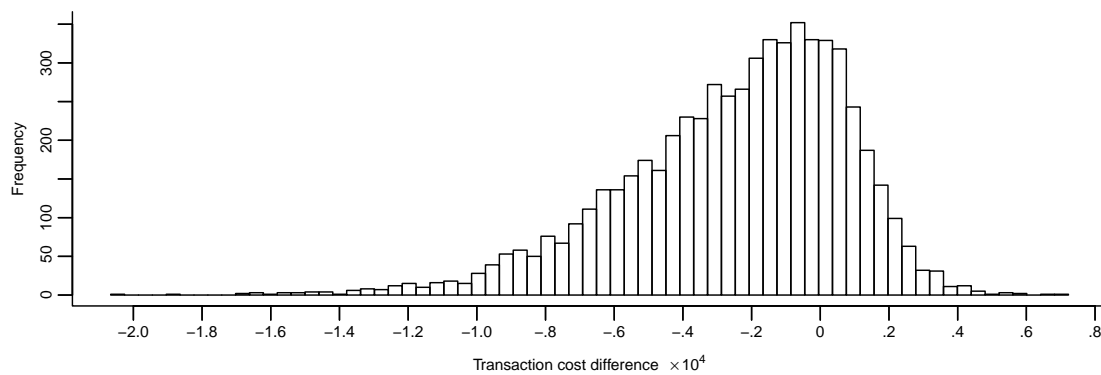
### 5.4.5 Simulation test run

Figure 5.20 (a) show an example of a stochastic volatility time series, simulated over one year (252 trading days) with hourly updates. The horizontal dotted line indicates the square root of the long-term mean  $\theta$  of the Heston stochastic volatility process. Figure 5.20 (b) show the corresponding stochastic optimal allocation strategy (5.16). Here, the horizontal dotted line indicates the constant optimal allocation strategy  $u^*$  with volatility  $\sqrt{\theta}$ . These plots just confirm the fact that  $u_k^* = \text{constant} \cdot \sigma_k^{-2}$ .

As mentioned above, one method of comparing stochastic volatility to constant volatility is to use simulation model I or III with constant volatility equal to the square root of the long-term mean  $\theta$  as a reference. The histogram of figure 5.21

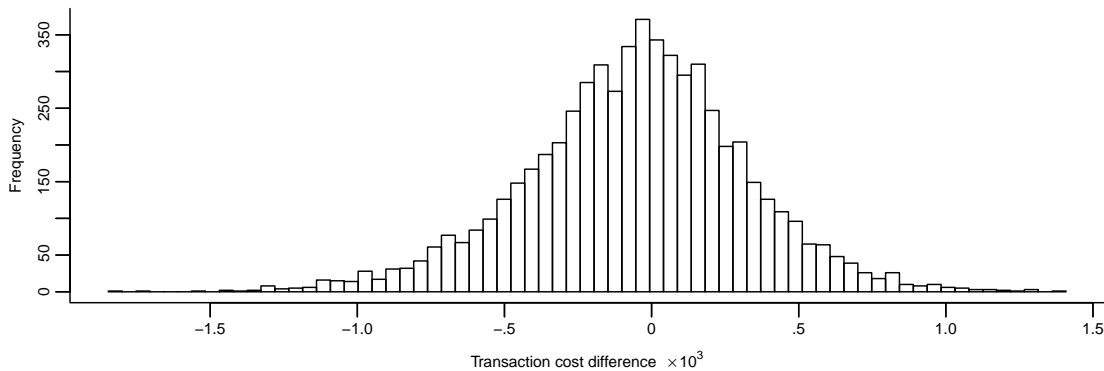


**Figure 5.20:** Stochastic volatility versus  $u^*$ .



**Figure 5.21:** Distribution of transaction cost differences between constant volatility model and stochastic volatility model using daily-rebalancing strategy.

show the distribution of the transaction cost differences between such a constant volatility reference portfolio and a portfolio with stochastic volatility, using the same underlying Brownian motion time series for the simulation of the risky asset and assuming hourly rebalancings and transaction cost proportion  $\lambda = .03$ . An obvious question is: how will the added variability of the stochastic volatility and thereby the added variability of the non-constant optimal allocation strategy affect the transaction costs? Will the added variability increase the total variability and thereby increase the transaction costs, will the added variability in the long run zero out and thereby not change the transaction costs in total or will the added variability counteract the already existing variability caused by the Brownian motion of the risky asset and thereby reduce the transaction costs? The histogram of figure 5.21 give us a mixed picture, but in general we see that the introduction of stochastic volatility and non-constant optimal allocation strategy increases the overall variability of the portfolio and thereby increases the total transaction costs. The total transaction costs of the simulated portfolio assuming constant volatility is .1005. The total transaction costs of the simulated portfolio assuming constant volatility is .2509, which is considerably more. And compared to an initial portfolio value of 1 it is extremely high. Considering the fact that one portfolio simulation run consists of 6048 time points and equally many transaction cost differences (when we assume portfolio rebalancings at an hourly basis) means that the distribution of figure 5.21 as well as the transaction cost totals, should be quite indicative about the general relation between the transaction costs of simulated portfolios assuming constant volatility and simulated portfolios assuming stochastic volatility.



**Figure 5.22:** Distribution of transaction cost differences between constant volatility model and stochastic volatility model using monthly-rebalancing strategy.

When we use the monthly-rebalancing strategy we get a different picture. The histogram of figure 5.22 shows the distribution of 6000 transaction cost differences<sup>2</sup>. Now, the differences in transaction costs between portfolios assuming

<sup>2</sup>Since one portfolio simulation run using the monthly-strategy only gives twelve transaction cost differences per run, the 6000 differences was obtained from 500 portfolio simulation runs.

constant volatility and portfolios assuming stochastic volatility are fairly evenly distributed around a mean approximately equal to zero.

### 5.4.6 Mean loss of utility

$\lambda = 0$		Sample means				
		Term. wealth	Total cost	Term. utility	Loss of utility	
Rebalancing strategy	Hourly	Const	1.0601	-	1.0275	-
		Stoch	1.0613	-	1.0281	$-5.7842 \times 10^{-4}$
	Every 4th hour	Const	1.0605	-	1.0277	-
		Stoch	1.0610	-	1.0279	$-2.1399 \times 10^{-4}$
	Daily	Const	1.0598	-	1.0273	-
		Stoch	1.0600	-	1.0275	$-1.4043 \times 10^{-4}$
	Every 3rd day	Const	1.0605	-	1.0277	-
		Stoch	1.0614	-	1.0282	$-4.7085 \times 10^{-4}$
	Every 12th day	Const	1.0601	-	1.0275	-
		Stoch	1.0616	-	1.0282	$-7.3880 \times 10^{-4}$
	Monthly	Const	1.0596	-	1.0273	-
		Stoch	1.0601	-	1.0275	$-2.4584 \times 10^{-4}$
	Bimonthly	Const	1.0601	-	1.0275	-
		Stoch	1.0613	-	1.0281	$-6.2473 \times 10^{-4}$
	Seminannually	Const	1.0606	-	1.0277	-
		Stoch	1.0601	-	1.0275	$2.6877 \times 10^{-4}$
	Annually	Const	1.0607	-	1.0278	-
		Stoch	1.0602	-	1.0275	$3.0116 \times 10^{-4}$

**Table 5.13:** The mean losses of utility of each rebalancing strategy and other related statistics,  $\lambda = 0$ .

Table 5.13 and figure 5.23 (a) show the mean losses of utility of each rebalancing strategy when assuming no transaction costs. Included in table 5.13 are also related statistics. The category "Const" in the table refers to the constant volatility (assumed equal to  $\sqrt{\theta}$ ) portfolio simulations using simulation model I. The results of this category serve as a reference point for assessing the impact of assuming stochastic volatility instead of constant volatility. The category "Stoch" refers to the stochastic volatility portfolio simulations using simulation model IV. All of the statistics of table 5.13 were calculated on a basis of 150,000 portfolio simulation runs for each rebalancing strategy for both constant volatility portfolios and stochastic volatility portfolios, but this time, the simulations were not done in parallel.

The estimates of table 5.13 as well as the plot of figure 5.23 (a) might suggest that the mean terminal utilities of the constant volatility portfolios are not significantly

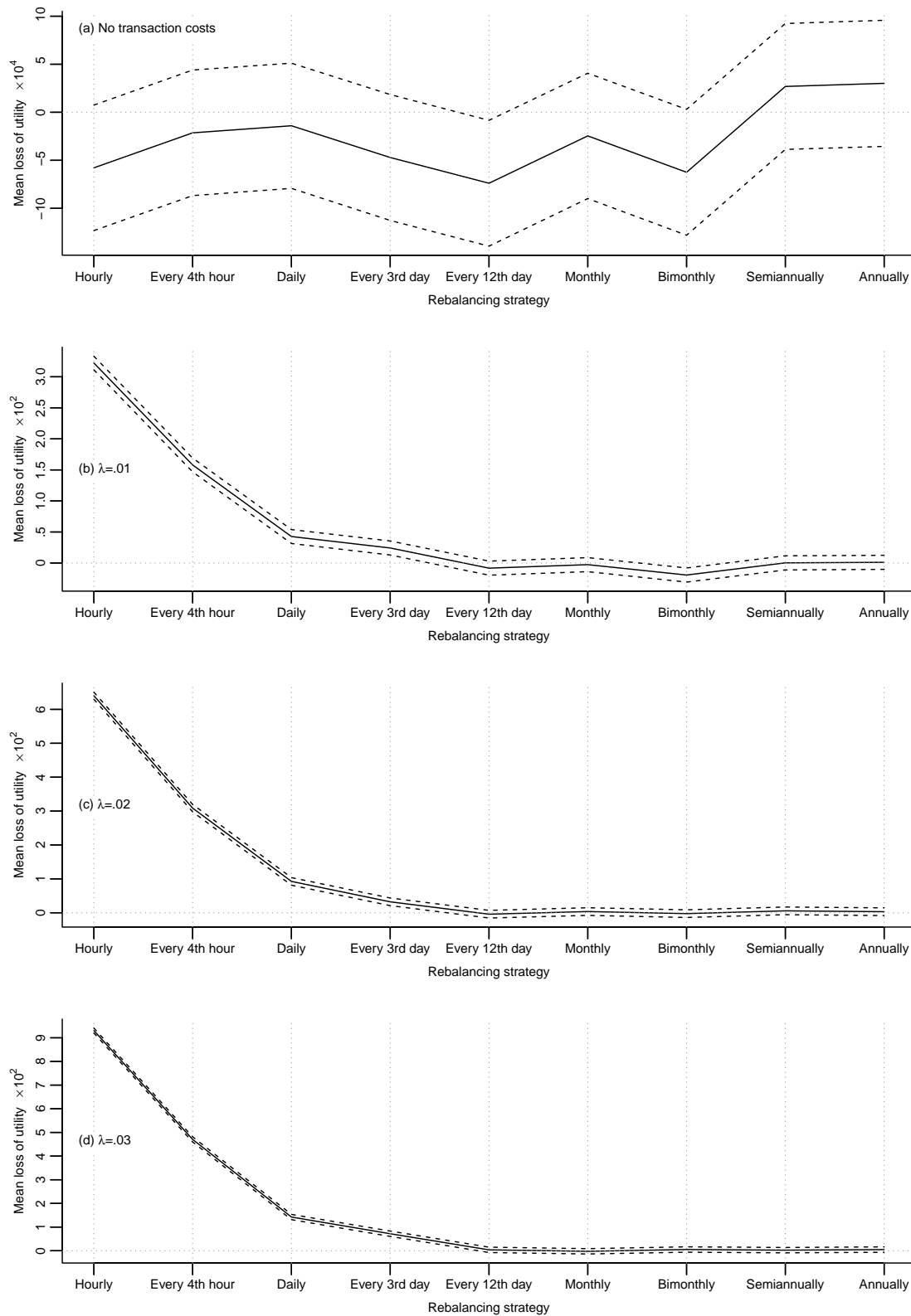


different from the mean terminal utilities of the stochastic volatility portfolios when we assume no transaction costs.

		Simulation model		$\lambda = .01$			
				Sample means			
			Term. wealth	Total cost	Term. utility	Loss of utility	
Rebalancing strategy	Hourly	Const	1.0219	$3.7062 \times 10^{-2}$	1.0079	-	
		Stoch	.9607	$9.8763 \times 10^{-2}$	.9756	$3.222 \times 10^{-2}$	
	Every 4th hour	Const	1.0411	$1.8711 \times 10^{-2}$	1.0177	-	
		Stoch	1.0105	$4.8909 \times 10^{-2}$	1.0019	$1.5796 \times 10^{-2}$	
	Daily	Const	1.0515	$.7676 \times 10^{-2}$	1.0231	-	
		Stoch	1.0431	$1.6667 \times 10^{-2}$	1.0189	$.4264 \times 10^{-2}$	
	Every 3rd day	Const	1.0563	$.5442 \times 10^{-2}$	1.0255	-	
		Stoch	1.0515	$.9980 \times 10^{-2}$	1.0231	$.2421 \times 10^{-2}$	
	Every 12th day	Const	1.0580	$.2227 \times 10^{-2}$	1.0265	-	
		Stoch	1.0598	$.2657 \times 10^{-2}$	1.0273	$-.0830 \times 10^{-2}$	
	Monthly	Const	1.0576	$.1685 \times 10^{-2}$	1.0262	-	
		Stoch	1.0581	$.1870 \times 10^{-2}$	1.0265	$-.0264 \times 10^{-2}$	
	Bimonthly	Const	1.0582	$.1194 \times 10^{-2}$	1.0265	-	
		Stoch	1.0620	$.1266 \times 10^{-2}$	1.0284	$-.1936 \times 10^{-2}$	
	Seminannualy	Const	1.0606	$.0701 \times 10^{-2}$	1.0277	-	
		Stoch	1.0604	$.0708 \times 10^{-2}$	1.0276	$.0023 \times 10^{-2}$	
	Annually	Const	1.0598	$.0495 \times 10^{-2}$	1.0274	-	
		Stoch	1.0597	$.0503 \times 10^{-2}$	1.0272	$.0110 \times 10^{-2}$	

**Table 5.14:** Mean losses of utility of each rebalancing strategy and other statistics,  $\lambda = .01$ .

As discussed earlier, table 5.14, 5.15 and 5.16 as well as figure 5.23 (b), (c) and (d) show that the introduction of transaction costs has a significant effect on the transaction cost totals of the high-frequency rebalancing strategies, compared to the transaction cost totals of the constant volatility portfolios. This relation is visualized in the histograms of figure 5.24 as well as in the histograms of figure A.5 and figure A.6 in the appendix. Here the distributions of the constant volatility portfolio transaction costs are given as the shaded histograms. The stochastic volatility portfolio transaction costs are in white. According to the confidence intervals of figure 5.23 (b), (c) and (d), there are significant differences from zero for the hourly-, the 'every 4th hour'-, the daily- and the 'every 3rd day'-rebalancing strategies. The value of the transaction cost proportion  $\lambda$  only serve to scale the transaction cost totals.



**Figure 5.23:** Mean losses of utility of each rebalancing strategy with 95% confidence intervals, (a)  $\lambda = 0$ , (b)  $\lambda = .01$ , (c)  $\lambda = .02$  and (d)  $\lambda = .03$ .

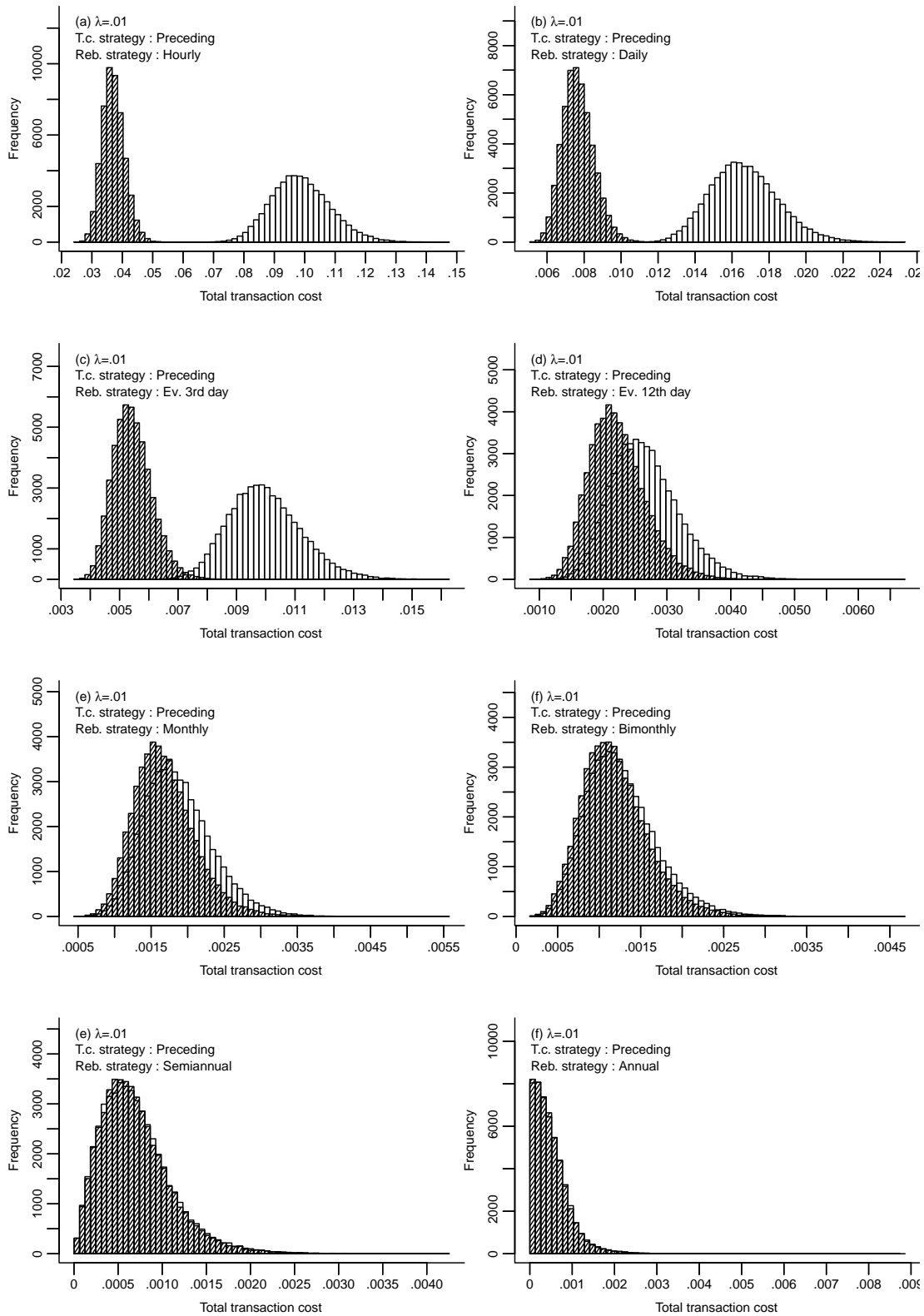


Figure 5.24: Distributions of total transaction costs of stochastic volatility portfolios and constant volatility portfolios (shaded),  $\lambda = .01$ .

		$\lambda = .02$				
		Simulation model	Term. wealth	Total cost	Term. utility	Loss of utility
Rebalancing strategy	Hourly	Const	.9849	$7.2801 \times 10^{-2}$	.9885	-
		Stoch	.8671	$18.7700 \times 10^{-2}$	.9245	$6.4033 \times 10^{-2}$
	Every 4th hour	Const	1.0217	$3.7072 \times 10^{-2}$	1.0078	-
		Stoch	.9629	$9.5487 \times 10^{-2}$	.9769	$3.0871 \times 10^{-2}$
	Daily	Const	1.0435	$1.5303 \times 10^{-2}$	1.0190	-
		Stoch	1.0255	$3.3041 \times 10^{-2}$	1.0098	$.9258 \times 10^{-2}$
	Every 3rd day	Const	1.0477	$1.0840 \times 10^{-2}$	1.0212	-
		Stoch	1.0415	$1.9869 \times 10^{-2}$	1.0179	$.3230 \times 10^{-2}$
	Every 12th day	Const	1.0558	$.4446 \times 10^{-2}$	1.0253	-
		Stoch	1.0567	$.5303 \times 10^{-2}$	1.0257	$-.0419 \times 10^{-2}$
	Monthly	Const	1.0564	$.3368 \times 10^{-2}$	1.0256	-
		Stoch	1.0557	$.3745 \times 10^{-2}$	1.0253	$.0357 \times 10^{-2}$
	Bimonthly	Const	1.0572	$.2386 \times 10^{-2}$	1.0260	-
		Stoch	1.0578	$.2525 \times 10^{-2}$	1.0263	$-.0253 \times 10^{-2}$
	Semiannually	Const	1.0590	$.1390 \times 10^{-2}$	1.0270	-
		Stoch	1.0580	$.1412 \times 10^{-2}$	1.0264	$.0572 \times 10^{-2}$
	Annually	Const	1.0595	$.1001 \times 10^{-2}$	1.0271	-
		Stoch	1.0588	$.1000 \times 10^{-2}$	1.0268	$.0313 \times 10^{-2}$

**Table 5.15:** Mean losses of utility of each rebalancing strategy and other statistics,  $\lambda = .02$ .

		$\lambda = .03$				
		Simulation model	Term. wealth	Total cost	Term. utility	Loss of utility
Rebalancing strategy	Hourly	Const	.9499	$10.7296 \times 10^{-2}$	.9699	-
		Stoch	.7840	$26.8417 \times 10^{-2}$	.8769	$9.3053 \times 10^{-2}$
	Every 4th hour	Const	1.0046	$5.5147 \times 10^{-2}$	.9989	-
		Stoch	.9165	$13.9884 \times 10^{-2}$	.9518	$4.7043 \times 10^{-2}$
	Daily	Const	1.0370	$2.2875 \times 10^{-2}$	1.0157	-
		Stoch	1.0096	$4.9213 \times 10^{-2}$	1.0015	$1.4247 \times 10^{-2}$
	Every 3rd day	Const	1.0441	$1.6238 \times 10^{-2}$	1.0193	-
		Stoch	1.0301	$2.9631 \times 10^{-2}$	1.0121	$.7190 \times 10^{-2}$
	Every 12th day	Const	1.0528	$.6667 \times 10^{-2}$	1.0238	-
		Stoch	1.0522	$.7932 \times 10^{-2}$	1.0234	$.0373 \times 10^{-2}$
	Monthly	Const	1.0545	$.5043 \times 10^{-2}$	1.0246	-
		Stoch	1.0550	$.5608 \times 10^{-2}$	1.0249	$-.0240 \times 10^{-2}$
	Bimonthly	Const	1.0575	$.3589 \times 10^{-2}$	1.0261	-
		Stoch	1.0564	$.3764 \times 10^{-2}$	1.0256	$.0516 \times 10^{-2}$
	Semiannually	Const	1.0580	$.2084 \times 10^{-2}$	1.0265	-
		Stoch	1.0576	$.2122 \times 10^{-2}$	1.0262	$.0247 \times 10^{-2}$
	Annually	Const	1.0599	$.1502 \times 10^{-2}$	1.0273	-
		Stoch	1.0589	$.1509 \times 10^{-2}$	1.0268	$.0489 \times 10^{-2}$

**Table 5.16:** Mean losses of utility of each rebalancing strategy and other statistics,  $\lambda = .03$ .

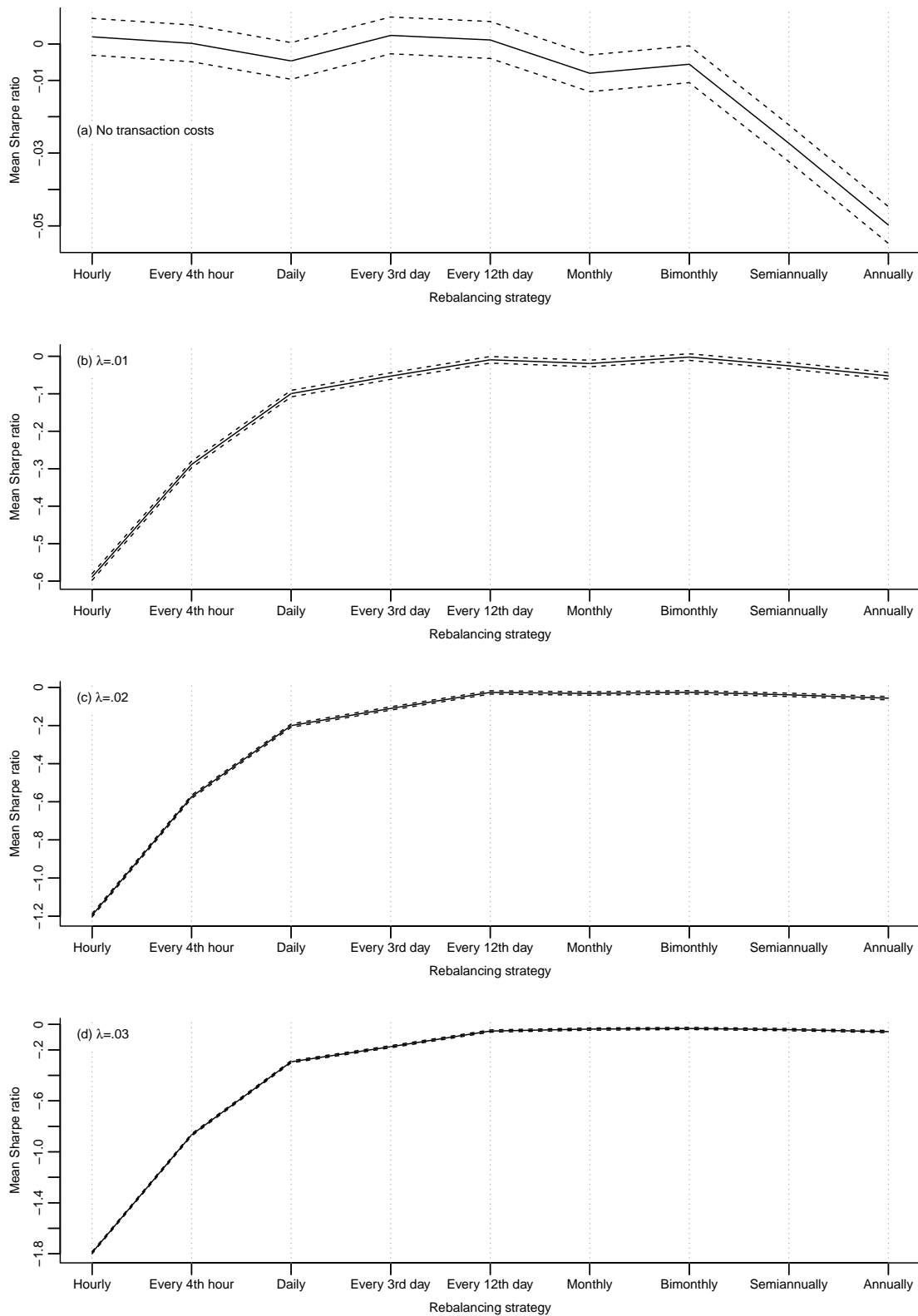
## 5.4.7 Portfolio return and Sharpe ratio

$\lambda = 0$		Sample means			Vol. of vol.	Corr.	Rank	
		Simulation model	Terminal log return	Vol.				Sharpe ratio
Rebalancing strategy	Hourly	Const	$4.4192 \times 10^{-2}$	.1688	$-.4145 \times 10^{-2}$	$1.5343 \times 10^{-3}$	-.0056	2
		Stock	$4.5225 \times 10^{-2}$	.1688	$.1968 \times 10^{-2}$	$1.5360 \times 10^{-3}$	-.0049	
	Every 4th hour	Const	$4.4526 \times 10^{-2}$	.1688	$-.2207 \times 10^{-2}$	$1.5386 \times 10^{-3}$	-.0007	4
		Stoch	$4.4925 \times 10^{-2}$	.1688	$.0176 \times 10^{-2}$	$1.5302 \times 10^{-3}$	-.0031	
	Daily	Const	$4.3856 \times 10^{-2}$	.1688	$-.6348 \times 10^{-2}$	$1.5332 \times 10^{-3}$	.0177	5
		Stoch	$4.4137 \times 10^{-2}$	.1688	$-.4662 \times 10^{-2}$	$1.5382 \times 10^{-3}$	.0156	
	Every 3rd day	Const	$4.4417 \times 10^{-2}$	.1688	$-.3171 \times 10^{-2}$	$1.5381 \times 10^{-3}$	.0336	1
		Stoch	$4.5351 \times 10^{-2}$	.1688	$.2358 \times 10^{-2}$	$1.5372 \times 10^{-3}$	.0346	
	Every 12th day	Const	$4.4167 \times 10^{-2}$	.1688	$-.6466 \times 10^{-2}$	$1.5870 \times 10^{-3}$	.2258	3
		Stoch	$4.5440 \times 10^{-2}$	.1688	$.1108 \times 10^{-2}$	$1.6178 \times 10^{-3}$	.2173	
	Monthly	Const	$4.3719 \times 10^{-2}$	.1687	$-1.0707 \times 10^{-2}$	$1.6989 \times 10^{-3}$	.3691	7
		Stoch	$4.4163 \times 10^{-2}$	.1688	$-.8058 \times 10^{-2}$	$1.7483 \times 10^{-3}$	.3571	
	Bimonthly	Const	$4.4080 \times 10^{-2}$	.1687	$-1.2349 \times 10^{-2}$	$2.1204 \times 10^{-3}$	.5955	6
		Stoch	$4.5230 \times 10^{-2}$	.1687	$-.5567 \times 10^{-2}$	$2.2039 \times 10^{-3}$	.5755	
	Semi- annually	Const	$4.4613 \times 10^{-2}$	.1686	$-2.4237 \times 10^{-2}$	$4.6609 \times 10^{-3}$	.8151	8
		Stoch	$4.4103 \times 10^{-2}$	.1686	$-2.7296 \times 10^{-2}$	$4.7218 \times 10^{-3}$	.8049	
	Annually	Const	$4.4725 \times 10^{-2}$	.1682	$-4.6118 \times 10^{-2}$	$8.9094 \times 10^{-3}$	.8488	9
		Stoch	$4.4119 \times 10^{-2}$	.1682	$-4.9781 \times 10^{-2}$	$8.8958 \times 10^{-3}$	.8480	

**Table 5.17:** Sharpe ratios of each rebalancing strategy and other related statistics,  $\lambda = 0$ .

Table 5.17 displays, among other statistics, the Sharpe ratios of each rebalancing strategy when assuming no transaction costs. The same Sharpe ratios with 95% confidence intervals are plotted in figure 5.25 (a). The picture is basically similar to what we get when assuming constant volatility: The best performing rebalancing strategies according to the Sharpe ratio are the high-frequency rebalancing strategies. We see that the 'every 3rd day'-strategy is ranked as number one, but with more simulations and consequently more precise estimates, the hourly-strategy would probably be ranked first, similar to the rankings of section 5.2.6.

Tables 5.18, 5.19 and 5.20 assumes transaction costs. Similar to what we saw in section 5.3.6, the introduction of transaction costs has the most negative effect on high-frequency rebalancing strategies. By comparing the 'Stoch'- with the 'Const'-category, we see that this effect is much stronger when we in addition assume stochastic volatility. We also see that the best performing rebalancing strategy this time is the bimonthly-strategy. This is the case for all three transaction cost proportions, although by small margin.



**Figure 5.25:** Sharpe ratios with 95% confidence intervals, (a)  $\lambda = 0$ , (b)  $\lambda = .01$ , (c)  $\lambda = .02$  and (d)  $\lambda = .03$ .

$\lambda = .01$		Sample means			Vol. of vol.	Corr.	Rank	
		Simulation model	Terminal log return	Vol.				Sharpe ratio
Rebalancing strategy	Hourly	Const	$.7422 \times 10^{-2}$	.1688	-.2220	$1.5315 \times 10^{-3}$	-.0086	9
		Stoch	$-5.4509 \times 10^{-2}$	.1688	-.5889	$1.5385 \times 10^{-3}$	-.0115	
	Every 4th hour	Const	$2.5900 \times 10^{-2}$	.1688	-.1126	$1.5385 \times 10^{-3}$	-.0030	8
		Stoch	$-.3836 \times 10^{-2}$	.1688	-.2888	$1.5280 \times 10^{-3}$	-.0006	
	Daily	Const	$3.6028 \times 10^{-2}$	.1688	-.0527	$1.5378 \times 10^{-3}$	.0134	7
		Stoch	$2.8118 \times 10^{-2}$	.1688	-.0996	$1.5311 \times 10^{-3}$	.0184	
	Every 3rd day	Const	$4.0425 \times 10^{-2}$	.1688	-.0269	$1.5446 \times 10^{-3}$	.0377	6
		Stoch	$3.6018 \times 10^{-2}$	.1688	-.0529	$1.5399 \times 10^{-3}$	.0299	
	Every 12th day	Const	$4.2247 \times 10^{-2}$	.1688	-.0179	$1.5918 \times 10^{-3}$	.2279	2
		Stoch	$4.3668 \times 10^{-2}$	.1688	-.0094	$1.6110 \times 10^{-3}$	.2230	
	Monthly	Const	$4.1755 \times 10^{-2}$	.1687	-.0223	$1.6951 \times 10^{-3}$	.3672	3
		Stoch	$4.2273 \times 10^{-2}$	.1688	-.0193	$1.7447 \times 10^{-3}$	.3611	
	Bimonthly	Const	$4.2232 \times 10^{-2}$	.1687	-.0233	$2.1240 \times 10^{-3}$	.5958	1
		Stoch	$4.5814 \times 10^{-2}$	.1687	-.0022	$2.2065 \times 10^{-3}$	.5790	
	Semi- annually	Const	$4.4347 \times 10^{-2}$	.1686	-.0262	$4.6881 \times 10^{-3}$	.8179	4
		Stoch	$4.4413 \times 10^{-2}$	.1686	-.0254	$4.7186 \times 10^{-3}$	.8044	
	Annually	Const	$4.4053 \times 10^{-2}$	.1682	-.0496	$8.8753 \times 10^{-3}$	.8472	5
		Stoch	$4.3685 \times 10^{-2}$	.1682	-.0524	$8.8856 \times 10^{-3}$	.8480	

**Table 5.18:** Sharpe ratios of each rebalancing strategy and other related statistics,  $\lambda = .01$ .

$\lambda = .02$		Sample means			Vol. of vol.	Corr.	Rank	
		Simulation model	Terminal log return	Vol.				Sharpe ratio
Rebalancing strategy	Hourly	Const	$-2.9378 \times 10^{-2}$	.1688	-.4400	$1.5331 \times 10^{-3}$	-.0194	9
		Stoch	$-15.6862 \times 10^{-2}$	.1688	-1.1955	$1.5325 \times 10^{-3}$	-.0097	
	Every 4th hour	Const	$.7239 \times 10^{-2}$	.1688	-.2231	$1.5413 \times 10^{-3}$	-.0112	8
		Stoch	$-5.1978 \times 10^{-2}$	.1688	-.5739	$1.5315 \times 10^{-3}$	-.0092	
	Daily	Const	$2.8353 \times 10^{-2}$	.1688	-.0982	$1.5250 \times 10^{-3}$	.0166	7
		Stoch	$1.1048 \times 10^{-2}$	.1688	-.2007	$1.5479 \times 10^{-3}$	.0150	
	Every 3rd day	Const	$3.2349 \times 10^{-2}$	.1688	-.0746	$1.5296 \times 10^{-3}$	.0257	6
		Stoch	$2.6277 \times 10^{-2}$	.1688	-.1106	$1.5332 \times 10^{-3}$	.0319	
	Every 12th day	Const	$4.0068 \times 10^{-2}$	.1688	-.0307	$1.5786 \times 10^{-3}$	.2196	2
		Stoch	$4.0788 \times 10^{-2}$	.1688	-.0264	$1.6269 \times 10^{-3}$	.2148	
	Monthly	Const	$4.0712 \times 10^{-2}$	.1687	-.0285	$1.6968 \times 10^{-3}$	.3684	3
		Stoch	$4.0073 \times 10^{-2}$	.1688	-.0322	$1.7472 \times 10^{-3}$	.3530	
	Bimonthly	Const	$4.1463 \times 10^{-2}$	.1687	-.0278	$2.1187 \times 10^{-3}$	.5917	1
		Stoch	$4.1845 \times 10^{-2}$	.1687	-.0257	$2.2022 \times 10^{-3}$	.5780	
	Semi- annually	Const	$4.3199 \times 10^{-2}$	.1686	-.0325	$4.6486 \times 10^{-3}$	.8136	4
		Stoch	$4.2087 \times 10^{-2}$	.1685	-.0392	$4.7045 \times 10^{-3}$	.8055	
	Annually	Const	$4.3483 \times 10^{-2}$	.1682	-.0538	$8.9180 \times 10^{-3}$	.8498	5
		Stoch	$4.2930 \times 10^{-2}$	.1682	-.0569	$8.9166 \times 10^{-3}$	.8478	

**Table 5.19:** Sharpe ratios of each rebalancing strategy and other related statistics,  $\lambda = .02$ .

$\lambda = .03$		Sample means			Vol. of vol.	Corr.	Rank	
		Simulation model	Terminal log return	Vol.				Sharpe ratio
Rebalancing strategy	Hourly	Const	$-6.5555 \times 10^{-2}$	.1687	-.6544	$1.5387 \times 10^{-3}$	-.0243	9
		Stoch	$-25.7451 \times 10^{-2}$	.1688	-1.7915	$1.5371 \times 10^{-3}$	-.0077	
	Every 4th hour	Const	$-.9602 \times 10^{-2}$	.1688	-.3229	$1.5363 \times 10^{-3}$	-.0057	8
		Stoch	$-10.1419 \times 10^{-2}$	.1688	-.8667	$1.5319 \times 10^{-3}$	-.0070	
	Daily	Const	$2.2271 \times 10^{-2}$	.1688	-.1342	$1.5374 \times 10^{-3}$	.0159	7
		Stoch	$-.4647 \times 10^{-2}$	.1688	-.2936	$1.5366 \times 10^{-3}$	.0099	
	Every 3rd day	Const	$2.8717 \times 10^{-2}$	.1688	-.0962	$1.5402 \times 10^{-3}$	.0323	6
		Stoch	$1.5383 \times 10^{-2}$	.1688	-.1752	$1.5396 \times 10^{-3}$	.0389	
	Every 12th day	Const	$3.7227 \times 10^{-2}$	.1688	-.0476	$1.5912 \times 10^{-3}$	.2280	4
		Stoch	$3.6426 \times 10^{-2}$	.1688	-.0523	$1.6164 \times 10^{-3}$	.2123	
	Monthly	Const	$3.8898 \times 10^{-2}$	.1688	-.0393	$1.7053 \times 10^{-3}$	.3703	2
		Stoch	$3.9259 \times 10^{-2}$	.1688	-.0371	$1.7535 \times 10^{-3}$	.3558	
	Bimonthly	Const	$4.1618 \times 10^{-2}$	.1687	-.0270	$2.1186 \times 10^{-3}$	.5984	1
		Stoch	$4.0734 \times 10^{-2}$	.1687	-.0321	$2.2030 \times 10^{-3}$	.5687	
	Semi- annually	Const	$4.2305 \times 10^{-2}$	.1685	-.0377	$4.6438 \times 10^{-3}$	.8141	3
		Stoch	$4.1753 \times 10^{-2}$	.1686	-.0414	$4.7405 \times 10^{-3}$	.8050	
	Annually	Const	$4.3823 \times 10^{-2}$	.1683	-.0518	$8.9301 \times 10^{-3}$	.8503	5
		Stoch	$4.2909 \times 10^{-2}$	.1682	-.0570	$8.8796 \times 10^{-3}$	.8491	

**Table 5.20:** Sharpe ratios of each rebalancing strategy and other related statistics,  $\lambda = .03$ .



# Chapter 6

## Conclusion

To recapitulate, in the most basic version of Merton's portfolio problem we assume that an investor has two investment choices, a risky asset, where the price dynamics is described by an SDE known as a geometric Brownian motion, and a risk-free asset, where the price dynamics is described by deterministic differential equation. The solution to Merton's portfolio problem, that is the optimal allocation strategy or trading strategy, is to keep a constant fraction  $u^*$  of the wealth in the risky asset and consequently a constant fraction  $1 - u^*$  of the wealth in the risk-free asset. This is a frequently used strategy among different investors such as banks, investment funds etc.

To answer the question about how the constant allocation strategy performs in a more realistic discrete time scenario, we introduced a time discretization and transferred the continuous SDE of the portfolio value into a discrete time counterpart by an Euler approximation. This gave us a simple, iterative method of simulating portfolios using the constant allocation strategy. Each portfolio simulation run, simulates the portfolio value over a period of one year, that is 252 trading days.

The constant allocation strategy requires that the investor rebalances the portfolio. In Merton's portfolio problem the investor is allowed to rebalance the portfolio continuously in time. In our discrete time simulation scenario the investor is only allowed to rebalance the portfolio at discrete time points, which is a more realistic assumption. By only allowing the investor to rebalance the portfolio at certain subsets of the complete set of time points, we were able to simulate and compare different rebalancing strategies.

To measure the impact of discretization, different rebalancing strategies and later transaction costs and stochastic volatility, we calculated the mean losses of utility and the Sharpe ratios of the different outcomes of the different simulation model configurations.

In simulation model I, we made the rather naive assumptions of no transaction costs and that the volatility of the risky asset is constant. Under these assumptions we found that to rebalance the portfolio as frequently as possible gave the best results, both in terms of mean loss of utility and mean Sharpe ratio, although the mean losses of utility of the semiannual-strategy and the annual-strategy weren't very far from zero.

In simulation model II and III we introduced transaction costs. We assumed proportional transaction costs, which means that the transaction costs were calculated as a proportionality constant times the amount transacted. The introduction of transaction costs had the biggest impact on the high-frequency rebalancing strategies. We concluded that for such strategies, small but frequent transactions were the norm. Rebalancing strategies with longer time intervals between each portfolio rebalancing entailed fewer, but potentially larger transactions at each rebalancing time point. "Potentially" is the keyword here, because even though high-frequency strategies meant small transactions, the sum of many small such transactions and consequently the sum of many small transactions costs, turned out in sum to generally be much more costly than to rebalance the portfolio less frequently. As a consequence we found that in terms of mean loss of utility, the best strategy is to rebalance the portfolio as seldom as possible. A simulation time interval of one year meant that the annual-rebalancing strategy was the best choice in terms of mean loss of utility and that the hourly-strategy was the worst. In terms of mean Sharpe ratio the picture was a little bit more complicated. In the simulation model without transaction costs we saw that for the low-frequency rebalancing strategies, such as the semiannual or the annual-strategy, the Sharpe ratio indicated a lower reward-to-risk ratio for such strategies, because of higher correlation between return and risk. This specific picture was more or less the same after the introduction of transaction costs, but the transaction costs meant that also high-frequency rebalancing strategies got low Sharpe ratios, not because of increased risk or correlation between return and volatility, but because of high transaction cost totals and consequently lower returns. The combined effect of the correlation between return and volatility and high transaction cost totals for the high-frequency rebalancing strategies, meant that the medium-frequency rebalancing strategies such as the monthly or the bimonthly strategy got the best Sharpe ratios in this scenario. We also looked at two different approaches with regard to the calculation of the transaction costs themselves. At each rebalancing time point, one approach was to rebalance the portfolio first and then deduct the transaction cost from the bank account (the risk-free asset). We referred to this approach as subsequent transaction costs. The second approach was to require the portfolio to be rebalanced after the transaction had been deducted. This approach, we referred to as preceding transaction costs. We found that the differences between these two approaches were minimal, and in practice perhaps not very relevant. We also looked at three , different transaction cost proportions,

$\lambda = .01$ ,  $\lambda = .02$  and  $\lambda = .03$ . We found that the  $\lambda$  scaled both the mean losses of utility and the mean Sharpe ratios proportionally.

In the last simulation model, simulation model IV, we did the more realistic assumption of stochastic volatility as opposed to the more unrealistic assumption of constant volatility. For modelling the stochastic volatility we used the well-known Heston model. The new stochastic volatility also implied a non-constant optimal allocation strategy. We found that this additional variability had a very negative impact on the mean losses of utility and the mean Sharpe ratios of portfolios with transaction costs using high-frequency rebalancing strategies. Compared to the mean losses of utility and the mean Sharpe ratios of constant volatility portfolio simulations, only portfolios using the hourly-, the 'every 4th hour'-, the daily- and the 'every 3rd day'-rebalancing strategy performed significantly worse. In terms of mean Sharpe ratio we basically saw the same picture as we did in the constant volatility scenario, but with even worse ratios for the four most frequent strategies due to the increased transaction cost totals. The best performing rebalancing strategy in the stochastic volatility scenario was the bimonthly-strategy.

A main focus of this thesis has been to build more or less realistic simulation models for assessing the performance of the constant allocation strategy, predicted as the optimal allocation strategy by Merton, in a discrete time scenario. Even though the simulation models of this thesis surely are more realistic than the continuous-time-, no transaction costs-, constant volatility-model that is assumed in Merton's portfolio problem, it has to be acknowledged that there are a lot of shortcomings in this thesis' simulation models as well. Firstly, we assume that the risky asset dynamics follow a geometric Brownian motion which implies normally distributed log returns. Research show that this is an unrealistic assumption, at least for the distribution of short-term log returns. Secondly, one might question the ability of the Heston model to simulate daily volatilities realistically. It seems that a better stochastic volatility model could have increased the realism of the simulations quite a bit. There are also many other ways of increasing the realism, for example by introducing stochastic drift, stochastic risk-free rate of return, better risky asset models and so forth. Of course, the disadvantage of making a model extremely complex is that it might lose generality and even become too complex to analyse and interpret.



# Appendix A

## Additional plots

### A.1 Simulation model II and III

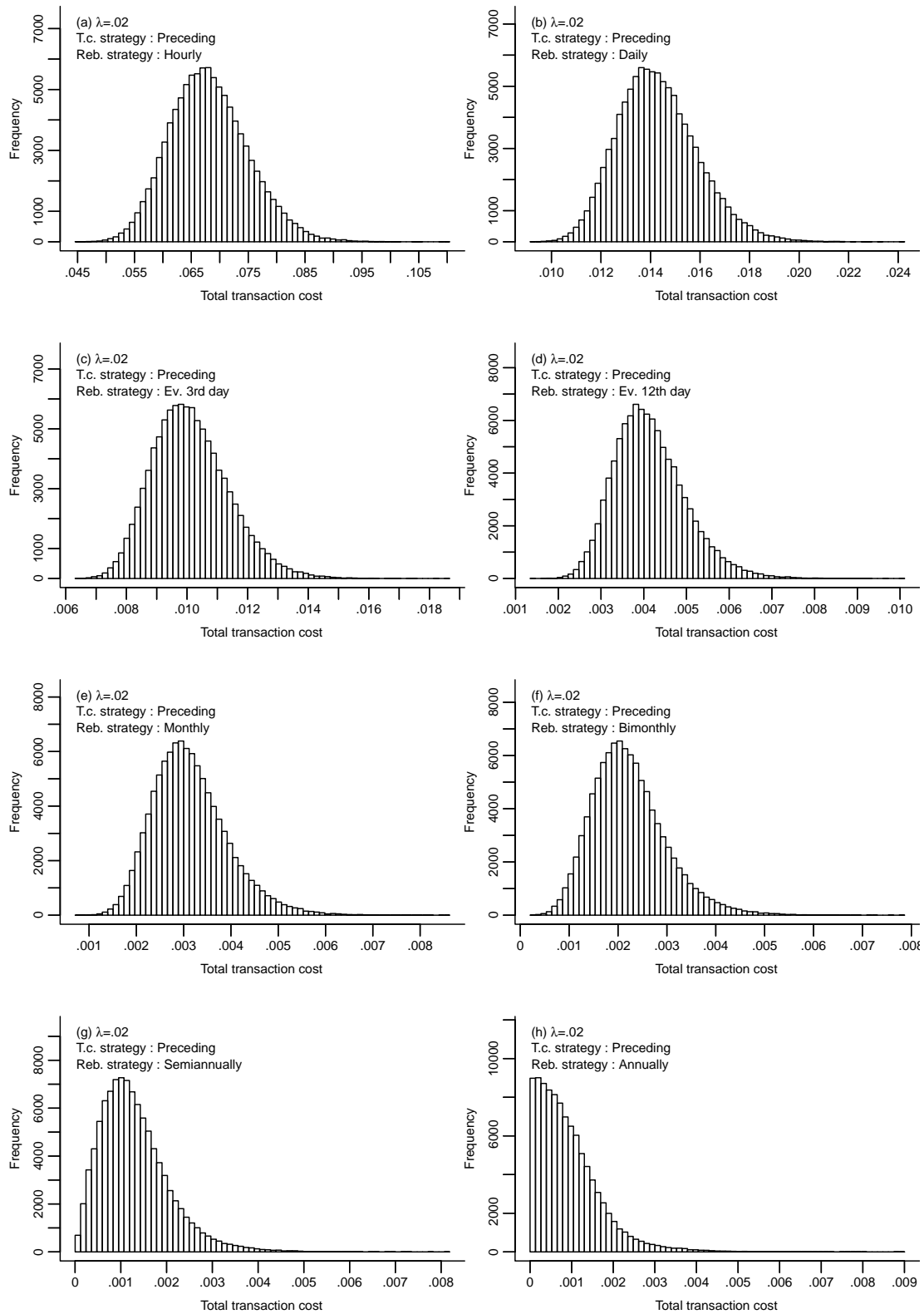


Figure A.1: Distributions of total transaction costs,  $\lambda = .02$ .

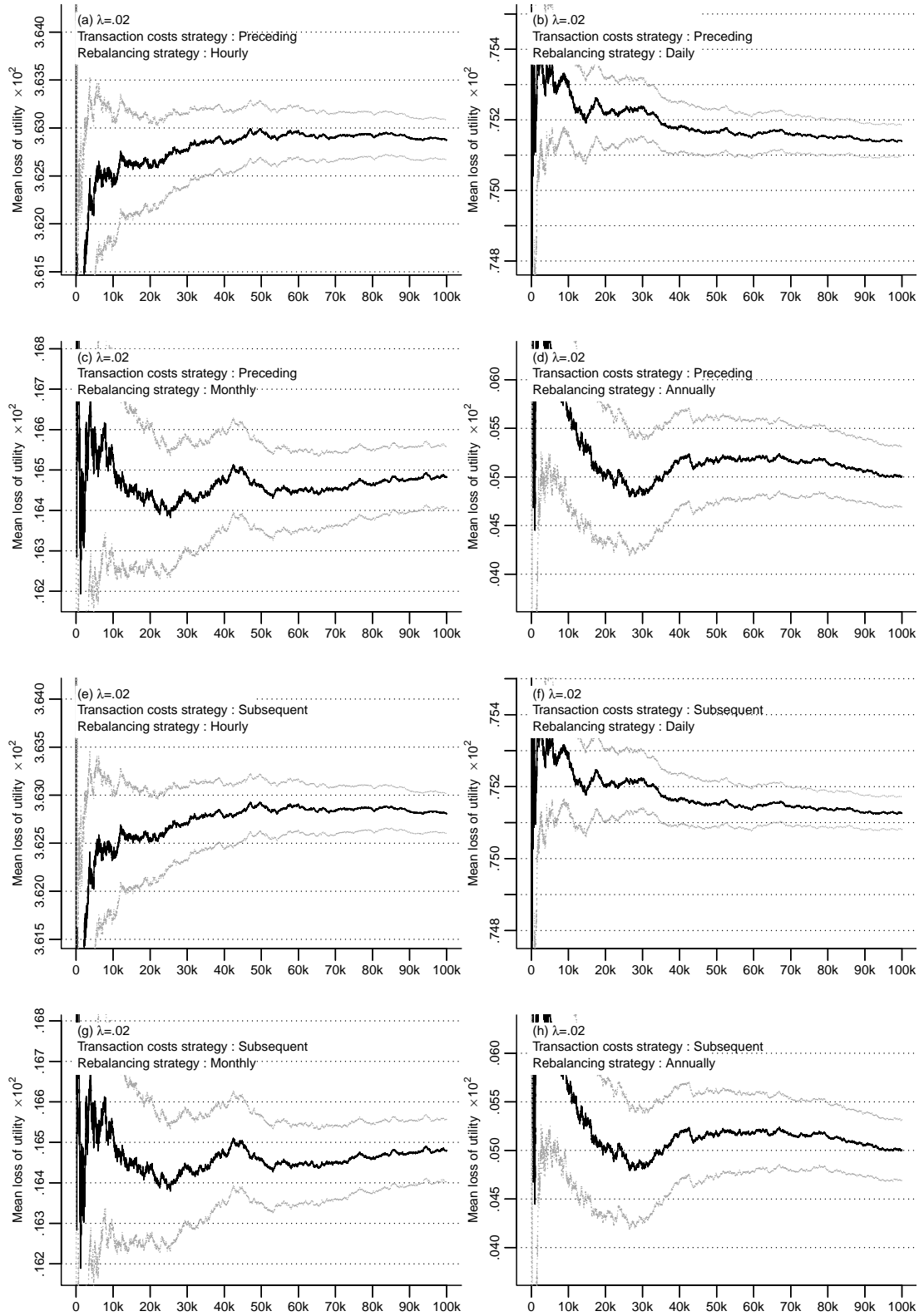


Figure A.2: The mean losses of utility with transaction cost proportion  $\lambda = .02$ . (a)-(d) preceding transaction costs and (e)-(h) subsequent transaction costs.

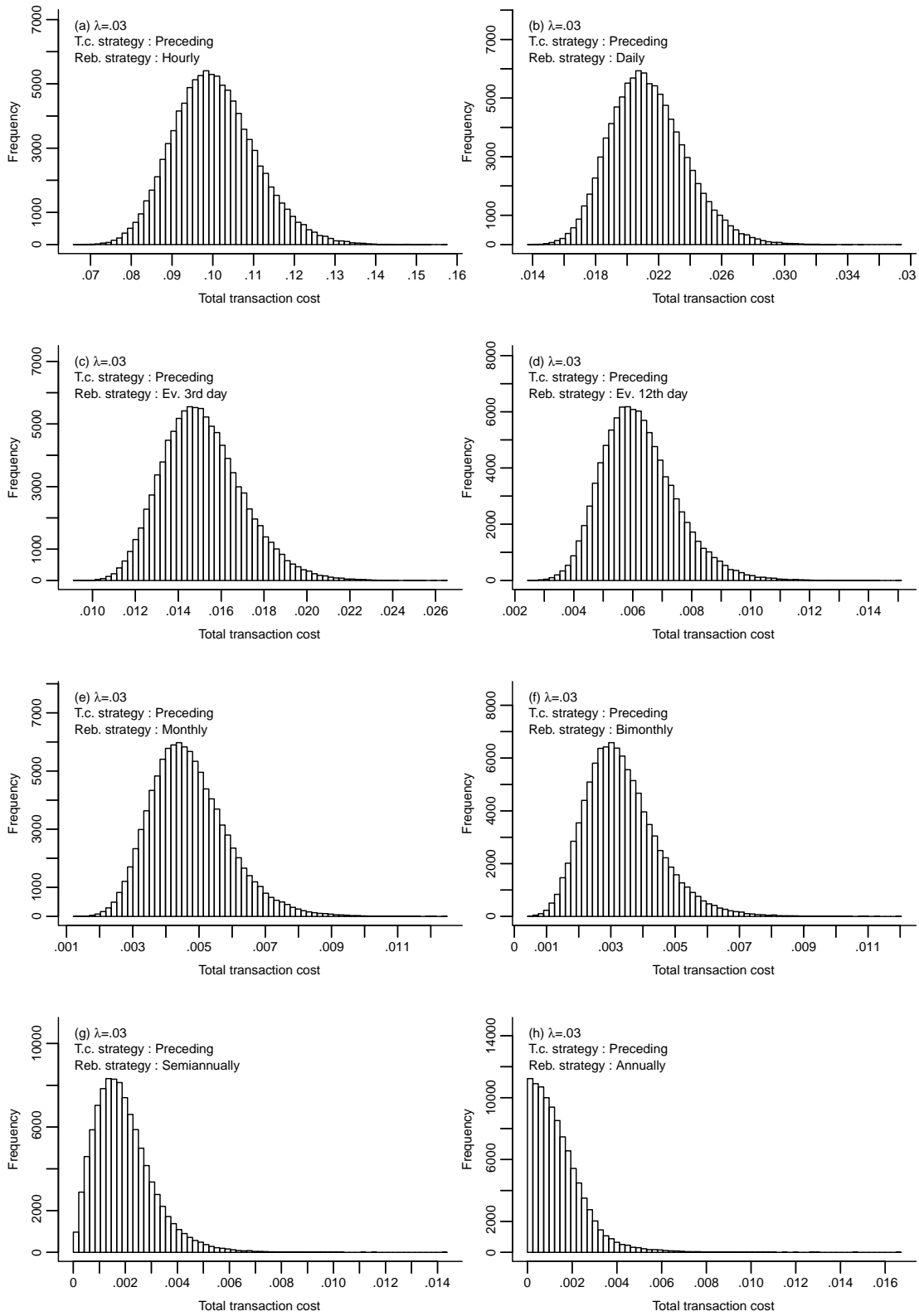


Figure A.3: Distributions of total transaction costs,  $\lambda = .03$ .



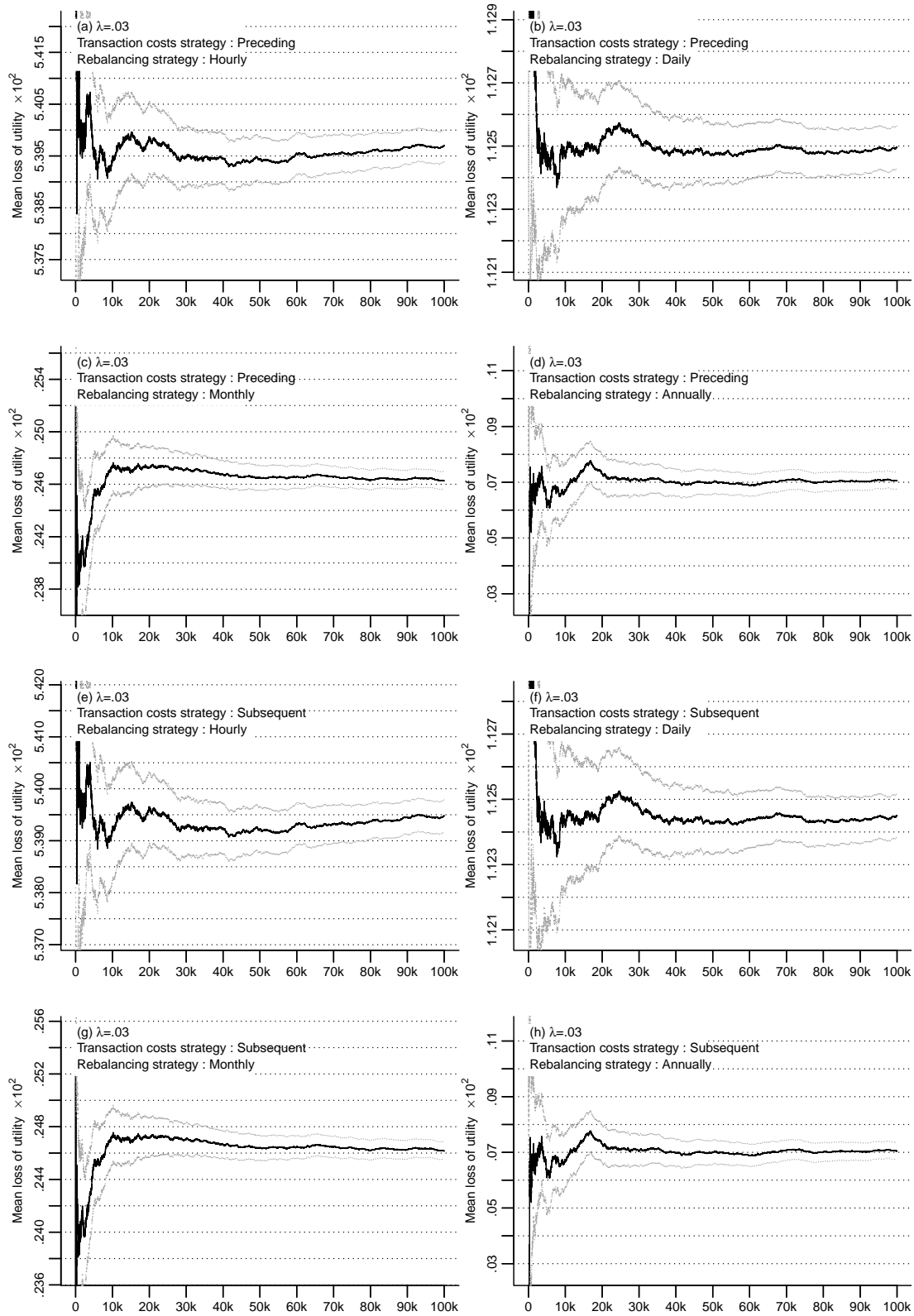
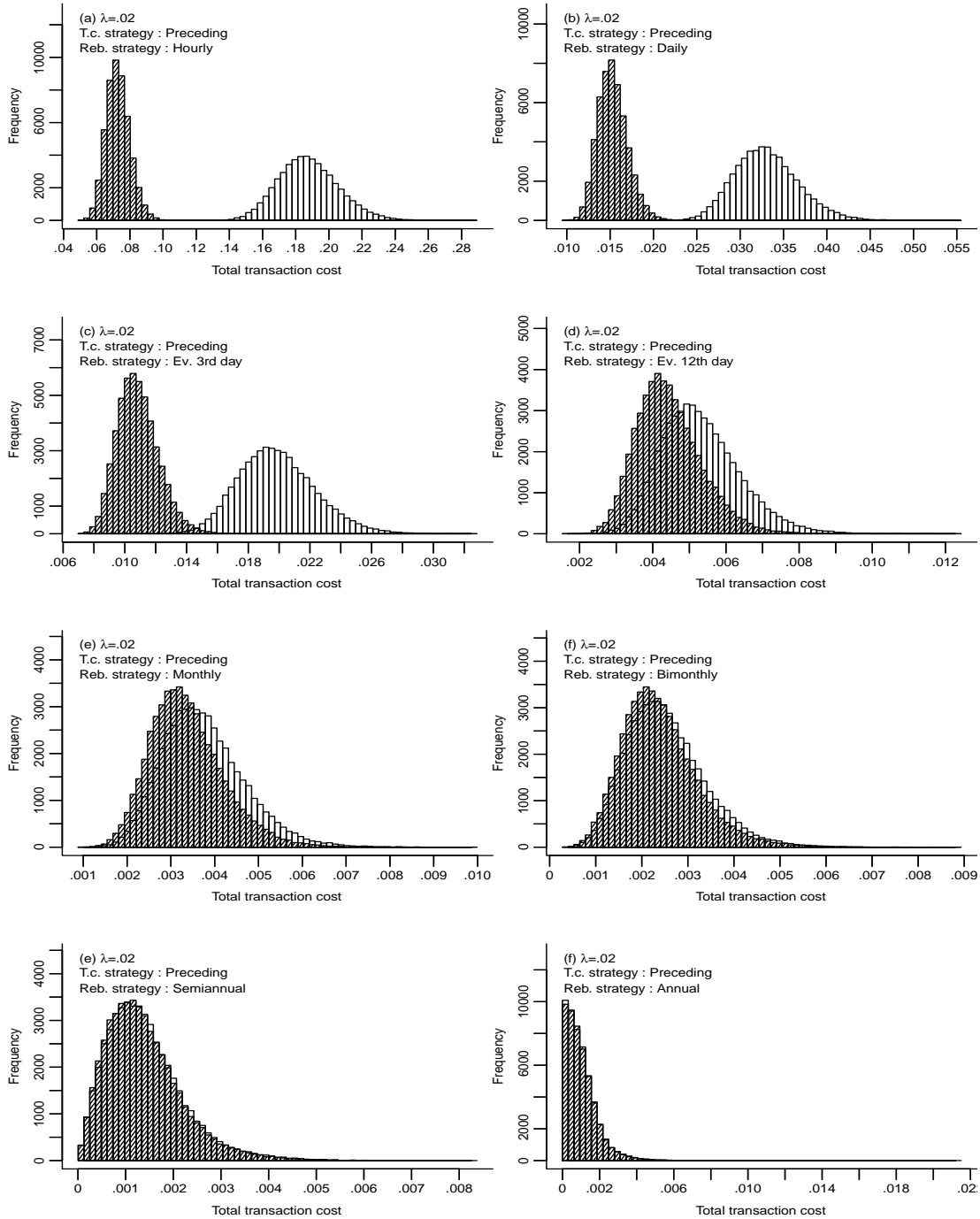


Figure A.4: The mean losses of utility with transaction cost proportion  $\lambda = .03$ . (a)-(d) preceding transaction costs and (e)-(h) subsequent transaction costs.

## A.2 Simulation model IV



**Figure A.5:** Distributions of total transaction costs of stochastic volatility portfolios and constant volatility portfolios (shaded),  $\lambda = .02$ .

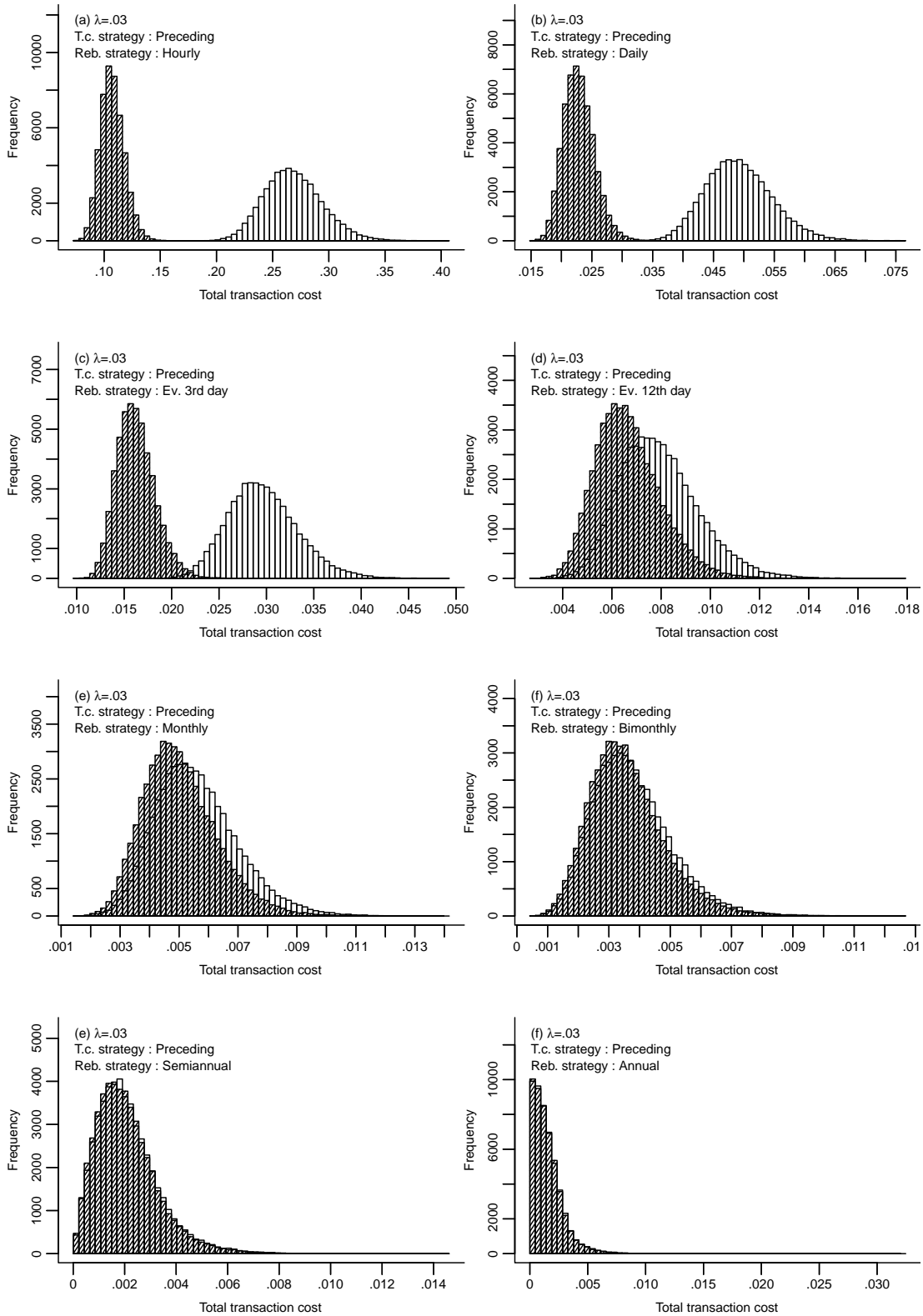


Figure A.6: Distributions of total transaction costs of stochastic volatility portfolios and constant volatility portfolios (shaded),  $\lambda = .03$ .



# Appendix B

## R source code

### B.1 Support functions

```
1 ##
2 # Master thesis
3 # Support functions
4 #
5
6 printex = function(table) {
7   #
8   # Convertes R tables to Latex table output.
9   #
10  rowNames = F
11  if (!is.null(rownames(table))) { rowNames = T } else { rowNames = F }
12  nRow = length(table[,1])
13  nCol = length(table[1,])
14  temp = ""
15  for (i in 1:nRow) {
16    if (rowNames) temp = paste(temp, rownames(table)[i], " & ", sep="")
17    for (j in 1:nCol) {
18      temp = paste(temp, table[i,j], sep="")
19      if (j < nCol) temp = paste(temp, " & ", sep="")
20    }
21    temp = paste(temp, " \\", "\\ ", sep="")
22    cat(temp, "\n", sep="")
23    temp = ""
24  }
25 }
26
27 is.zero = function(x) {
28   #
29   # Checks if elements of vector x == 0.
30   #
31   return(x == 0)
32 }
33
34 trimLast = function(x) {
35   #
36   # Removes last element of vector x.
37   #
38   n = length(x)
39   return(x[-n])
}
```

```

40 }
41
42 strictlyIncreasing = function(x) {
43   #
44   # Checks if the elements of vector x is strictly increasing.
45   #
46   strictlyInc = T
47   for (i in 2:length(x)) { strictlyInc = strictlyInc * ((x[i]/x[i-1])>1) }
48   return(strictlyInc)
49 }
50
51 strictlyDecreasing = function(x) {
52   #
53   # Checks if the elements of vector x is strictly decreasing.
54   #
55   strictlyDec = T
56   for (i in 2:length(x)) { strictlyDec = strictlyDec * ((x[i]/x[i-1])<1) }
57   return(strictlyDec)
58 }
59
60 merge.list = function(x) {
61   #
62   # Merges list elements of list x.
63   #
64   n.x = length(x)
65   merged = NULL
66   for (k in 1:n.x) merged = c(merged,x[[k]])
67   return(merged)
68 }
69
70 listDiff = function(listA , listB) {
71   #
72   # Computes the difference between lists.
73   #
74   listNames = names(listA)
75   returnList = vector("list",length(listNames))
76   names(returnList) = listNames
77   for (k in 1:length(listNames)) { returnList[[k]] = listA[[k]] - listB[[k]] }
78   return(returnList)
79 }
80
81 subsample = function(x,nSub=10000) {
82   #
83   # Downsamples vector x to length nSub.
84   #
85   inc = length(x) / nSub
86   subsamples = 1:nSub*NA
87   for (k in seq(0,nSub-2,2)) {
88     actSubsample = x[(k*inc+1):((k+2)*inc)]
89     minSubsample = min(actSubsample)
90     maxSubsample = max(actSubsample)
91     minIndex = match(minSubsample,actSubsample)
92     maxIndex = match(maxSubsample,actSubsample)
93     if (minIndex < maxIndex) { subsamples[k+1] = minSubsample; subsamples[k+2] =
94       maxSubsample }
95     else { subsamples[k+1] = maxSubsample; subsamples[k+2] = minSubsample }
96   }
97   indexList = seq(inc,length(x),inc)
98   return(list(index=indexList,subsamples=subsamples))
99 }
100 niceplot = function(x,y,xTicks,yTicks,xLabels,yLabels,xTitle,yTitle,figsPerPage
101   =4,caption=F,y.superscript=F,y.addCustom=0,nCol=1,multiPlot=F,newDev=T,
  plotHist=F,horizLines=F,downsample=F,nSub=10000,breaks,...) {
  #

```

```

102 # Secures nice plots in Latex.
103 #
104 if (missing(y)) y = NULL
105 if (caption) { yLength = c(20.18,6.33,4.05,2.48,1.88) }
106 else { yLength = c(20.18,6.33,4.05,2.83,2.20) }
107 if (newDev) {
108   windows(11.9, yLength[figsPerPage])
109   par(mfrow=c(1, nCol), cex.axis=.7, oma=c(0,0,0,0), mar=c(1.3,1.15,.55,0), mgp=c
      (2,.5,0), las=0, bty="l", lab=c(10,7,7))
110   y.adj = y.addCustom
111   if (y.superscript) y.adj = y.adj + .17
112   if (!missing("xTitle") && missing("yTitle")) par(cex.lab=.7, mar=c
      (2.4,1.15,.55,0), mgp=c(1,.5,0))
113   if (missing("xTitle") && !missing("yTitle")) par(cex.lab=.7, mar=c(1.3,2.15+y.
      adj,.55,0), mgp=c(1,.5,0))
114   if (!missing("xTitle") && !missing("yTitle")) par(cex.lab=.7, mar=c(2.4,2.15+y
      .adj,.55,0), mgp=c(1,.5,0))
115 }
116
117 if (downsample) {
118   if (is.null(y)) subsampleObject = subsample(x, nSub)
119   else subsampleObject = subsample(y, nSub)
120   x = subsampleObject$index
121   y = subsampleObject$subsamples
122 }
123
124 if (!newDev && !multiPlot) lines(x,y,...)
125 else {
126   if (plotHist) {
127     histObject = hist(x, breaks=breaks, freq=T, main="", axes=F, ann=F,...)
128     box(bty="l")
129   }
130   else plot(x,y, type="l", xaxt="n", yaxt="n", ann=F,...)
131
132   if (missing(xTicks)) xTicks = axis(1, labels=F)
133   if (missing(xLabels)) { xLabels = sub("0[.]",".", format(xTicks, scientific=F)
      ); xLabels = gsub(" ","", xLabels) }
134   if (!any(xTicks==0) && min(xTicks)<=0 && max(xTicks)>=0) { xLabels = sort(c
      (0, xLabels)); xTicks = sort(c(0, xTicks)) }
135   options(warn=-1)
136   xLabels[as.numeric(xLabels)==0] = 0
137   axis(1, xTicks, xLabels, padj=-.5)
138
139   if (missing(yTicks)) yTicks = axis(2, labels=F)
140   if (missing(yLabels)) yLabels = sub("0[.]",".", format(yTicks, scientific=F))
141   if (!any(yTicks==0) && min(yTicks)<=0 && max(yTicks)>=0) { yLabels = sort(c
      (0, yLabels)); yTicks = sort(c(0, yTicks)) }
142   yLabels[as.numeric(yLabels)==0] = 0
143   axis(2, yTicks, yLabels, padj=-.1)
144   if (horizLines) abline(h=yTicks, lty=3)
145
146   if (!missing("xTitle")) title(xlab=xTitle, line=1.3)
147   if (!missing("yTitle")) title(ylab=yTitle, line=1.55)
148
149   if (plotHist) invisible(histObject)
150 }
151 }
152
153 niceLines = function(x,y,...) { niceplot(x,y,newDev=F,...) }
154
155 niceHist = function(x,y, horizLines=F, breaks,...) {
156   #
157   # Secures nice histograms in Latex.
158   #
159   if (missing(breaks)) breaks = 10

```

```

160 niceplot(x, plotHist=T, breaks=breaks, ...)
161 }
162
163 addHist = function(x, ...) {
164   #
165   # Superimposes a histogram on active plotting device.
166   #
167   histObject = hist(x, plot=F, ...)
168   xLeft = trimLast(histObject$breaks)
169   delta = xLeft[2] - xLeft[1]
170   yBottom = trimLast(rep(0, length(xLeft)))
171   xRight = trimLast(xLeft + delta)
172   yTop = histObject$counts
173   rect(xLeft, yBottom, xRight, yTop, ...)
174   invisible(histObject)
175 }
176
177 nicelegend = function(...) {
178   #
179   # Makes nice plot legends.
180   #
181   legendObject = legend(..., plot=F)
182   x.tune = 1/10
183   x.left = legendObject$text$x - (legendObject$text$x - legendObject$rect$left)*x.
184         tune
185   y.bottom = legendObject$rect$top - legendObject$rect$h*.9
186   x.right = x.left + legendObject$rect$w
187   y.top = (legendObject$text$y + legendObject$rect$top) / 2
188   rect(x.left, y.bottom, x.right, y.top, col="white", border="white")
189   invisible(legend(...))
190 }
191
192 cumMean = function(x) {
193   #
194   # Calculates the cumulative mean along a vector.
195   #
196   cumulativeMean = cumsum(x) / 1:length(x)
197   return(cumulativeMean)
198 }
199
200 cumSd = function(x) {
201   #
202   # Calculates the cumulative standard deviation along a vector.
203   #
204   nn = 1:length(x)
205   cumulativeSd = sqrt((1/(nn-1)) * (cumsum(x^2) - cumsum(x)^2/nn))
206 }
207
208 colRange = function(x) {
209   #
210   # Calculates the ranges of the columns of matrix x.
211   #
212   n.col = ncol(x)
213   ranges = matrix(NA, 2, n.col)
214   for (k in 1:n.col) { ranges[,k] = range(x[,k]) }
215   return(ranges)
216 }
217
218 colSds = function(X) {
219   #
220   # Computes the standard deviations along the columns of matrix X.
221   #
222   nCol = ncol(X)
223   sds = 1:nCol*NA
224   for (k in 1:nCol) { sds[k] = sd(X[,k]) }

```



```

224     return(sds)
225 }
226
227 rowSds = function(X) {
228     #
229     # Computes the standard deviations along the rows of matrix X.
230     #
231     nRow = nrow(X)
232     sds = 1:nRow*NA
233     for (k in 1:nRow) { sds[k] = sd(X[k,]) }
234     return(sds)
235 }
236
237 colCorrs = function(X,Y) {
238     #
239     # Computes the correlations between the columns of matrices X and Y,
240     # respectively.
241     #
242     nCol = ncol(X)
243     corrs = 1:nCol*NA
244     for (k in 1:nCol) { corrs[k] = cor(X[,k],Y[,k]) }
245     return(corrs)
246 }
247
248 colCumsums = function(X) {
249     #
250     # Calculates cumulative sums along columns of matrix X.
251     #
252     nRow = nrow(X)
253     nCol = ncol(X)
254     cumsums = matrix(NA,nRow,nCol)
255     for (k in 1:nCol) { cumsums[,k] = cumsum(X[,k]) }
256     return(cumsums)
257 }

```

## B.2 Initialization and estimation

```

1  ##
2  # Master Thesis
3  # Estimation of parameters
4  #
5
6  source("R/supportFunctions.R")
7  source("R/machinery_general.R")
8
9  graphics.off()
10
11 #
12 # Function declarations
13 #
14
15 logReturn = function(x) {
16     #
17     # Computes the log returns of a time series x.
18     #
19     n = length(x)
20     x.up = x[2:n]
21     x.low = x[1:(n-1)]
22     logReturns = log(x.up/x.low)
23     return(logReturns)

```

```

24 }
25
26 optimalControl = function(drift , volatility , rent , riskAversion) {
27   #
28   # Computes the optimal control following a power-type utility function.
29   #
30   control = pmax(pmin((drift - rent)/((1 - riskAversion)*volatility^2), 1), 0)
31   return(control)
32 }
33
34 #
35 # Loading OBX price and treasury bill data
36 #
37
38 obx = read.table("Datasett/OBX-finalSet.txt")
39 tbill = read.table("Datasett/tbill-finalSet.txt")
40
41 niceplot(obx[,2], yTitle="Price")
42 abline(v=3188, lty=3)
43 text(3188, min(obx[,2]), "Lehman brothers", adj=c(.05, -.4), cex=.7, srt=90)
44 savePlot("images/obx", type="eps")
45
46 #
47 # Estimation of the annual drift , volatility and rate of return
48 #
49
50 nTradingDays = 252
51 nTimePoints = 6048
52 obxLogReturns = logReturn(obx$price)
53 drift = nTradingDays * mean(obxLogReturns)
54 volatility = sqrt(nTradingDays) * sd(obxLogReturns)
55 tbillLogReturns = (1/252)*log(1+tbill$rent)
56 rent = 252*mean(tbillLogReturns)
57
58 niceplot(obxLogReturns, yTitle="Log return")
59 abline(h=0, lty=3)
60 abline(v=3187, lty=3)
61 text(3187, min(obxLogReturns), "Lehman brothers", adj=c(.05, -.4), cex=.7, srt=90)
62 savePlot("images/obxLogReturns", type="eps")
63
64 #
65 # Estimation of risk aversion
66 #
67
68 alpha = .01
69 wRisky = .5
70 wSure = 1 - wRisky
71 VaR = -(wRisky*quantile(obxLogReturns, alpha)+wSure*quantile(tbillLogReturns,
72   alpha))
72 delta = 1/252
73 riskAve = riskAversion(drift , volatility , rent , VaR, delta , alpha)
74
75 #
76 # Estimation of optimal control
77 #
78
79 uStar = optimalControl(drift , volatility , rent , riskAve)
80
81 #
82 # Estimation of Heston parameters
83 #
84
85 shortTermVar = function(x, windowLength, delta=1) {
86   #
87   # Calculates volatility of short term window.

```

```

88 | #
89 | n = length(x)
90 | shortTermVar = 1:(n-windowLength+1) * NA
91 | for (k in 1:(n-windowLength+1)) { shortTermVar[k] = (1/delta) * var(x[k:(k+
    |   windowLength-1])) }
92 | return(shortTermVar)
93 | }
94 |
95 | var.2 = shortTermVar(obsLogReturns,2,1/nTradingDays)
96 | var.2.mean = mean(var.2)
97 | var.2.sd = sd(var.2)
98 | var.3 = shortTermVar(obsLogReturns,3,1/nTradingDays)
99 | var.3.mean = mean(var.3)
100 | var.3.sd = sd(var.3)
101 | var.4 = shortTermVar(obsLogReturns,4,1/nTradingDays)
102 | var.4.mean = mean(var.4)
103 | var.4.sd = sd(var.4)
104 | var.5 = shortTermVar(obsLogReturns,5,1/nTradingDays)
105 | var.5.mean = mean(var.5)
106 | var.5.sd = sd(var.5)
107 | var.6 = shortTermVar(obsLogReturns,6,1/nTradingDays)
108 | var.6.mean = mean(var.6)
109 | var.6.sd = sd(var.6)
110 | var.7 = shortTermVar(obsLogReturns,7,1/nTradingDays)
111 | var.7.mean = mean(var.7)
112 | var.7.sd = sd(var.7)
113 |
114 | delta = 1 / nTimePoints
115 |
116 | # Window length : 2
117 | n = length(var.2)
118 | var.2.up = var.2[2:n]
119 | var.2.down = var.2[1:(n-1)]
120 | y.2 = (var.2.up - var.2.down) / sqrt(var.2.down)
121 | x1.2 = 1 / sqrt(var.2.down)
122 | x2.2 = sqrt(var.2.down)
123 | linreg.2 = lm(y.2 ~ x1.2 + x2.2 - 1)
124 | summary(linreg.2)
125 | beta1 = linreg.2$coeff[1]
126 | beta2 = linreg.2$coeff[2]
127 | var.long.2 = -beta1 / beta2
128 | reversionRate.2 = -beta2 / delta
129 | var.init.2 = var.long.2
130 | var.inc.2 = diff(var.2)
131 | volOfVol.2 = sd(var.inc.2)
132 | correlation.2 = cor(obsLogReturns[1:length(var.inc.2)], var.inc.2)
133 |
134 | # Window length : 3
135 | n = length(var.3)
136 | var.3.up = var.3[2:n]
137 | var.3.down = var.3[1:(n-1)]
138 | y.3 = (var.3.up - var.3.down) / sqrt(var.3.down)
139 | x1.3 = 1 / sqrt(var.3.down)
140 | x2.3 = sqrt(var.3.down)
141 | linreg.3 = lm(y.3 ~ x1.3 + x2.3 - 1)
142 | beta1 = linreg.3$coeff[1]
143 | beta2 = linreg.3$coeff[2]
144 | var.long.3 = -beta1 / beta2
145 | reversionRate.3 = -beta2 / delta
146 | var.init.3 = var.long.3
147 | var.inc.3 = diff(var.3)
148 | volOfVol.3 = sd(var.inc.3)
149 | correlation.3 = cor(obsLogReturns[1:length(var.inc.3)], var.inc.3)
150 |
151 | # Window length : 4

```

```

152 n = length(var.4)
153 var.4.up = var.4[2:n]
154 var.4.down = var.4[1:(n-1)]
155 y.4 = (var.4.up - var.4.down) / sqrt(var.4.down)
156 x1.4 = 1 / sqrt(var.4.down)
157 x2.4 = sqrt(var.4.down)
158 linreg.4 = lm(y.4 ~ x1.4 + x2.4 - 1)
159 beta1 = linreg.4$coeff[1]
160 beta2 = linreg.4$coeff[2]
161 var.long.4 = -beta1 / beta2
162 reversionRate.4 = -beta2 / delta
163 var.init.4 = var.long.4
164 var.inc.4 = diff(var.4)
165 volOfVol.4 = sd(var.inc.4)
166 correlation.4 = cor(obxLogReturns[1:length(var.inc.4)], var.inc.4)
167
168 # Window length : 5
169 n = length(var.5)
170 var.5.up = var.5[2:n]
171 var.5.down = var.5[1:(n-1)]
172 y.5 = (var.5.up - var.5.down) / sqrt(var.5.down)
173 x1.5 = 1 / sqrt(var.5.down)
174 x2.5 = sqrt(var.5.down)
175 linreg.5 = lm(y.5 ~ x1.5 + x2.5 - 1)
176 beta1 = linreg.5$coeff[1]
177 beta2 = linreg.5$coeff[2]
178 var.long.5 = -beta1 / beta2
179 reversionRate.5 = -beta2 / delta
180 var.init.5 = var.long.5
181 var.inc.5 = diff(var.5)
182 volOfVol.5 = sd(var.inc.5)
183 correlation.5 = cor(obxLogReturns[1:length(var.inc.5)], var.inc.5)
184
185 # Window length : 6
186 n = length(var.6)
187 var.6.up = var.6[2:n]
188 var.6.down = var.6[1:(n-1)]
189 y.6 = (var.6.up - var.6.down) / sqrt(var.6.down)
190 x1.6 = 1 / sqrt(var.6.down)
191 x2.6 = sqrt(var.6.down)
192 linreg.6 = lm(y.6 ~ x1.6 + x2.6 - 1)
193 beta1 = linreg.6$coeff[1]
194 beta2 = linreg.6$coeff[2]
195 var.long.6 = -beta1 / beta2
196 reversionRate.6 = -beta2 / delta
197 var.init.6 = var.long.6
198 var.inc.6 = diff(var.6)
199 volOfVol.6 = sd(var.inc.6)
200 correlation.6 = cor(obxLogReturns[1:length(var.inc.6)], var.inc.6)
201
202 # Window length : 7
203 n = length(var.7)
204 var.7.up = var.7[2:n]
205 var.7.down = var.7[1:(n-1)]
206 y.7 = (var.7.up - var.7.down) / sqrt(var.7.down)
207 x1.7 = 1 / sqrt(var.7.down)
208 x2.7 = sqrt(var.7.down)
209 linreg.7 = lm(y.7 ~ x1.7 + x2.7 - 1)
210 beta1 = linreg.7$coeff[1]
211 beta2 = linreg.7$coeff[2]
212 var.long.7 = -beta1 / beta2
213 reversionRate.7 = -beta2 / delta
214 var.init.7 = var.long.7
215 var.inc.7 = diff(var.7)
216 volOfVol.7 = sd(var.inc.7)

```

```

217 correlation.7 = cor(obsLogReturns[1:length(var.inc.7)], var.inc.7)
218
219 # Plotting and saving
220 y.range = range(var.2)
221 y.ticks = c(0,1,2,3,4)
222 niceplot(var.2, yTicks=y.ticks, yTitle="Volatility", figsPerPage=5, ylim=y.range)
223 nicelegend("topleft", "(a) Window length = 2", bty="n", bg="white", cex=.7)
224 savePlot("images/volatility_winLength2", type="eps")
225 niceplot(var.7, yTicks=y.ticks, yTitle="Volatility", figsPerPage=5, ylim=y.range)
226 nicelegend("topleft", "(b) Window length = 7", bty="n", bg="white", cex=.7)
227 savePlot("images/volatility_winLength7", type="eps")
228
229 # Construction of output table
230 tab = matrix(NA, 6, 6)
231
232 tab[1,] = c(2, var.init.2, reversionRate.2, var.long.2, volOfVol.2, correlation.2)
233 tab[2,] = c(3, var.init.3, reversionRate.3, var.long.3, volOfVol.3, correlation.3)
234 tab[3,] = c(4, var.init.4, reversionRate.4, var.long.4, volOfVol.4, correlation.4)
235 tab[4,] = c(5, var.init.5, reversionRate.5, var.long.5, volOfVol.5, correlation.5)
236 tab[5,] = c(6, var.init.6, reversionRate.6, var.long.6, volOfVol.6, correlation.6)
237 tab[6,] = c(7, var.init.7, reversionRate.7, var.long.7, volOfVol.7, correlation.7)
238
239 colNames = c("1", "var.init", "revRate", "var.long", "volOfVol", "Correlation")
240 colnames(tab) = colNames
241 transformation = cbind(rep(1,6), rep(1e2,6), rep(1,6), rep(1e2,6), rep(1,6), rep(1e2
,6))
242 tab = round(tab*transformation, 4)
243 as.data.frame(tab)
244 for (k in 1:6) {
245   tab[k,2] = paste(tab[k,2], "\\e{\\text{-2}}", sep="")
246   tab[k,4] = paste(tab[k,4], "\\e{\\text{-2}}", sep="")
247   tab[k,6] = paste(tab[k,6], "\\e{\\text{-2}}", sep="")
248 }
249 printex(tab)
250
251 # Plotting differences of 5-day variances
252 niceplot(diff(sqrt(var.5)), yTitle="Change of volatility")
253 savePlot("images/5dayVol_diff", type="eps")

```

```

1 ##
2 # Master Thesis
3 # Initialization of parameters
4 #
5
6 # Basic parameters
7 initWealth = 1
8 nTradingDays = 252
9 nDailyIncrements = 24 # Hourly updates of portfolio value
10 nDailyRebs = 12/252 # Monthly-rebalancing strategy
11 drift = .0657
12 volatility = .2537
13 rent = .0449
14 riskAversion = .5255
15 uStar = optimalControl(drift, volatility, rent, riskAversion)
16
17 # Additional transaction cost parameters
18 costProp = .03
19
20 # Additional stochastic volatility parameters
21 var.init = 6.7456e-2
22 reversionRate = 320.1192
23 var.long = 6.7456e-2
24 volOfVol = .0590
25 correlation = 2.6706e-2

```

```

26 | uStar.constVol = optimalControl(drift , sqrt(var.long) , rent , riskAversion)
27 |
28 | # Setting up simulation model I input parameters
29 | paramSet = c(initWealth , nTradingDays , nDailyIncrements , nDailyRebs , drift ,
              | volatility , rent , riskAversion , uStar)
30 | names(paramSet) = c("initWealth" , "nTradingDays" , "nDailyIncrements" , "nDailyRebs"
              | " , " drift" , " volatility" , " rent" , " riskAversion" , " uStar")
31 |
32 | # Setting up simulation model II and III input parameters
33 | paramSet.transCost = c(paramSet , costProp)
34 | names(paramSet.transCost) = c("initWealth" , "nTradingDays" , "nDailyIncrements" , "
              | nDailyRebs" , " drift" , " volatility" , " rent" , " riskAversion" , " uStar" , " costProp")
35 |
36 | # Setting up simulation model IV input parameters
37 | paramSet.constVol = c(initWealth , nTradingDays , nDailyIncrements , nDailyRebs , drift ,
              | sqrt(var.long) , rent , riskAversion , uStar.constVol , costProp)
38 | paramSet.stochVol = c(initWealth , nTradingDays , nDailyIncrements , nDailyRebs , drift ,
              | rent , riskAversion , costProp , var.init , reversionRate , var.long , volOfVol ,
              | correlation)
39 | nParam.stochVol = length(paramSet.stochVol)
40 | names(paramSet.stochVol) = c("initWealth" , "nTradingDays" , "nDailyIncrements" , "
              | nDailyRebs" , " drift" , " rent" , " riskAversion" , " costProp" , " var.init" , "
              | reversionRate" , " var.long" , " volOfVol" , " correlation")
41 |
42 | # Calculating number of time points and equidistant time increment delta
43 | nTimePoints = nTradingDays * nDailyIncrements
44 | delta = 1 / nTimePoints

```

## B.3 General simulation machinery

```

1 | ##
2 | # Master Thesis
3 | # General machinery
4 | #
5 |
6 | logReturn = function(x) {
7 |   #
8 |   # Computes the log returns of a time series x.
9 |   #
10 |   n = length(x)
11 |   xUp = x[2:n]
12 |   xLow = x[1:(n-1)]
13 |   logReturns = log(xUp/xLow)
14 |   return(logReturns)
15 | }
16 |
17 | riskAversion = function(drift , volatility , rent , VaR , delta , alpha) {
18 |   #
19 |   # Computes the risk aversion parameter of a power-type utility function
20 |   # through Value at Risk.
21 |   #
22 |   qAlpha = qnorm(alpha)
23 |   lengthVol = length(volatility)
24 |   if (lengthVol==1) solution = 1:2*NA
25 |   else solution = matrix(NA , lengthVol , 2)
26 |   a = drift - rent + qAlpha*volatility/sqrt(delta)
27 |   b = 2*volatility^2*(VaR/delta+rent)
28 |   if (lengthVol==1) {
29 |     solution[1] = 1 + (drift-rent)*(a+sqrt(a^2+b))/b
30 |     solution[2] = 1 + (drift-rent)*(a-sqrt(a^2+b))/b

```

```

31     }
32     else {
33         solution[,1] = 1 + (drift-rent)*(a+sqrt(a^2+b))/b
34         solution[,2] = 1 + (drift-rent)*(a-sqrt(a^2+b))/b
35     }
36     return(solution)
37 }
38
39 expectedWealth = function(initWealth, drift, rent, uStar, tp) {
40     #
41     # Computes the expected wealth.
42     #
43     return(initWealth*exp((drift*uStar + rent*(1-uStar))*tp))
44 }
45
46 stDevWealth = function(initWealth, drift, volatility, rent, uStar, tp) {
47     #
48     # Computes the expected standard deviation of the wealth.
49     #
50     expecWealth = expectedWealth(initWealth, drift, rent, uStar, tp)
51     return(sqrt(expecWealth^2 * (exp(volatility^2*uStar^2*tp) - 1)))
52 }
53
54 expectedLogReturn = function(drift, volatility, rent, uStar, delta) {
55     #
56     # Computes the expected log return.
57     #
58     return((drift*uStar + rent*(1-uStar) - .5*volatility^2*uStar^2)*delta)
59 }
60
61 stDevLogReturn = function(volatility, uStar, delta) {
62     #
63     # Computes the expected standard deviation of the log returns.
64     #
65     return(sqrt(volatility^2*uStar^2*delta))
66 }
67
68 simRiskyAsset = function(initValue, drift, volatility, BM) {
69     #
70     # Calculates risky asset values according to Brownian motion BM. Uses
71     # Euler-Maruyama approximation.
72     #
73     nTimePoints = length(BM)
74     delta = 1 / nTimePoints
75     inc = c(0, diff(BM))
76     simValue = initWealth * (1 + drift*delta + volatility*inc[1])
77     for (i in 2:nTimePoints) { simValue[i] = simValue[i-1] * (1 + drift*delta +
78         volatility*inc[i]) }
79     return(simValue)
80 }
81
82 riskFreeAsset = function(initValue, rent, nTimePoints) {
83     #
84     # Calculates risk-free asset values using Euler approximation.
85     #
86     delta = 1 / nTimePoints
87     value = initValue * (1 + rent*delta)
88     for (i in 2:nTimePoints) { value[i] = value[i-1] * (1 + rent*delta) }
89     return(value)
90 }
91
92 expectedLogReturn = function(drift, volatility, rent, uStar, tp) {
93     #
94     # Computes the expected log return from time 0 to time tp.
95     #

```

```

95 |   return((drift*uStar + rent*(1-uStar) - .5*volatility^2*uStar^2)*tp)
96 | }
97 |
98 | stDevLogReturn = function(volatility ,uStar ,tp) {
99 |   #
100 |   # Computes the ex ante standard deviation of the log returns from time 0
101 |   # to time tp.
102 |   #
103 |   return(volatility*uStar*sqrt(tp))
104 | }
105 |
106 | exAnteSharpeRatio = function(drift , volatility ,rent ,uStar ,tp) {
107 |   #
108 |   # Computes the ex ante, that is the expected Sharpe ratio of the
109 |   # portfolio.
110 |   #
111 |   expecLogReturn = expectedLogReturn(drift , volatility ,rent ,uStar ,tp)
112 |   sdLogReturn = stDevLogReturn(volatility ,uStar ,tp)
113 |   return((expecLogReturn - rent) / sdLogReturn)
114 | }
115 |
116 | sharpeRatio = function(terminalWealth ,rent ,sdLogReturn ,nTimePoints) {
117 |   #
118 |   # Computes the ex post Sharpe ratio given the terminal wealth of a time
119 |   # series of wealths, the benchmark risk free rate of return and the
120 |   # standard deviation of the log returns of the wealth series.
121 |   #
122 |   return((log(terminalWealth) - rent) / (nTimePoints*sdLogReturn))
123 | }
124 |
125 | kill = function() {
126 |   #
127 |   # Removes redundant doSMP workers.
128 |   #
129 |   rmSessions(all.names=T)
130 | }
131 |
132 | optimalControl = function(drift , volatility ,rent ,riskAversion) {
133 |   #
134 |   # Computes the optimal control following a power-type utility function.
135 |   #
136 |   control = pmax(pmin((drift-rent)/((1-riskAversion)*volatility^2),1),0)
137 |   return(control)
138 | }
139 |
140 | utility = function(x,param,type="power") {
141 |   #
142 |   # Computes the power-type utility of a wealth x.
143 |   #
144 |   if (type=="power") utility = x^param
145 |   return(utility)
146 | }
147 |
148 | brownianIncrement = function(nSims ,nTimePoints ,thread=1) {
149 |   #
150 |   # Generates nSims rows of nTimePoints Brownian increments each increment
151 |   # with variance 1 / nTimePoints.
152 |   #
153 |   delta = 1 / nTimePoints
154 |   N = nSims * nTimePoints
155 |   brownianMatrix = t(matrix(rnorm(N,0 ,sqrt(delta)),nTimePoints ,nSims))
156 |   return(brownianMatrix)
157 | }
158 |
159 | multiSim = function(nSims ,nCores ,func ,paramSet) {

```



```

160 #
161 # Splits nSims simulations into nSims/nCores simulations which are
162 # simulated on nSims/nCores processor cores. The nSims/nCores subsets
163 # are then put together and returned.
164 #
165 if (nSims/nCores != round(nSims/nCores)) stop("Number of simulations is not a
      multiple of number of cores.")
166 cat(" Doing", format(nSims, scientific=F), " simulation runs on", nCores, " core(s)
      ... \n")
167 flush.console()
168 cat(" Parameter set :", paramSet, "\n")
169 flush.console()
170 timeStart = proc.time()[3][[1]]
171 workers = startWorkers(nCores)
172 registerDoSMP(workers)
173 multiSimObject = foreach(j=1:nCores) %dopar% func(nSims/nCores, paramSet)
174 stopWorkers(workers)
175 timeElapsed = proc.time()[3][[1]] - timeStart
176 cat(format(nSims, scientific=F), " simulation runs completed in", timeElapsed, "
      seconds.\n")
177 if (is.list(multiSimObject[[1]])) {
178   multiSimNames = names(multiSimObject[[1]])
179   returnObject = vector("list", length(multiSimNames))
180   names(returnObject) = multiSimNames
181   for (k in 1:length(multiSimNames)) { returnObject[[k]] = c(sapply(
      multiSimObject, get, x=multiSimNames[k])) }
182 }
183 else if (is.matrix(multiSimObject[[1]])) { returnObject = abind(multiSimObject
      , along=1) }
184 return(returnObject)
185 }
186
187 distribute = function(nSims, nCores, func, paramSets) {
188   #
189   # Apply-style wrapper function for simulating multiple parameter sets.
190   #
191   nParamSets = nrow(paramSets)
192   cat(" Simulating", nParamSets, " parameter sets... \n")
193   flush.console()
194   returnList = list()
195   timeStart = proc.time()[3][[1]]
196   for (k in 1:nParamSets) { returnList[[k]] = multiSim(nSims, nCores, func,
      paramSets[k,]) }
197   timeElapsed = proc.time()[3][[1]] - timeStart
198   cat(nParamSets, " parameter sets completed in", timeElapsed, " seconds.\n")
199   return(returnList)
200 }

```

## B.4 Simulation model I

### B.4.1 Simulation machinery

```

1 ##
2 # Master Thesis
3 # Simulation model I
4 # Simulation algorithm
5 #
6

```

```

7 | simPortfolio = function(nSims, paramSet, brownianDataSet=NULL) {
8 |   #
9 |   # Simulates nSims portfolios following the 9 parameter values of paramSet
10 |   # and returns terminal utilities of theoretical and simulated wealth.
11 |   #
12 |
13 |   logReturn = function(x) {
14 |     #
15 |     # Computes the log returns of a time series x.
16 |     #
17 |     n = length(x)
18 |     xUp = x[2:n]
19 |     xLow = x[1:(n-1)]
20 |     logReturns = log(xUp/xLow)
21 |     return(logReturns)
22 |   }
23 |
24 |   brownianIncrement = function(n, delta) {
25 |     #
26 |     # Simulates random series of n brownian increments with variance delta.
27 |     #
28 |     return(rnorm(n,0,sqrt(delta)))
29 |   }
30 |
31 |   #
32 |   # Assigning variables.
33 |   #
34 |   nParams = length(paramSet)
35 |   if (nParams != 9) stop(paste("Number of input parameters equals ",nParams,".
36 |     Must equal 9.",sep=""))
37 |   varNames = c("initWealth", "nTradingDays", "nDailyIncrements", "nDailyRebs", "
38 |     drift", "volatility", "rent", "riskAversion", "uStar")
39 |   for (j in 1:9) { assign(varNames[j], paramSet[j]) }
40 |
41 |   #
42 |   # Initializing the simulation structure.
43 |   #
44 |   simIndex = 1:nSims
45 |   nTimePoints = nTradingDays * nDailyIncrements
46 |   lastIndex = nTimePoints
47 |   delta = 1 / nTimePoints
48 |   timePoints = seq(delta,1,delta)
49 |   nRebDelay = nDailyIncrements / nDailyRebs
50 |   rebIndex = seq(nRebDelay, nTimePoints, nRebDelay)
51 |   rebIndex.length = length(rebIndex)
52 |   days = seq(delta*nTradingDays, nTradingDays, delta*nTradingDays)
53 |   rebDays = days[rebIndex]
54 |   ones = rep(1,nRebDelay)
55 |
56 |   # Common structure
57 |   simWealth = 1:nTimePoints * NA
58 |
59 |   # Setting start time
60 |   timeStart = proc.time()[3][[1]]
61 |
62 |   #
63 |   # If nSims = 1, the full simulation scheme is applied. If nSims > 1, to
64 |   # gain speed, the compact form will be applied.
65 |   #
66 |   if (nSims == 1) {
67 |     # Initialization of time series
68 |     simWealth.risky = 1:nTimePoints * NA
69 |     simWealth.riskfree = 1:nTimePoints * NA

```

```

70   propInRisky = 1:nTimePoints * NA
71   propInRiskfree = 1:nTimePoints * NA
72
73   # Brownian increments and motion
74   if (!is.null(brownianDataSet)) load(brownianDataSet)
75   else inc = brownianIncrement(nTimePoints, delta)
76   BM = cumsum(inc)
77
78   # Calculation of theoretical wealth and relevant statistics
79   thWealth = initWealth*exp((drift*uStar+rent*(1-uStar)-.5*volatility^2*uStar
80     ^2)*timePoints+volatility*uStar*BM)
81   thTermWealth = tail(thWealth,1)
82   sdThWealth = sd(thWealth)
83   thLogReturn = logReturn(c(initWealth, thWealth))
84   sdThLogReturn = sd(thLogReturn)
85
86   # Initial time points to be simulated
87   activeIndices = 1:nRebDelay
88   rebPoint = tail(activeIndices,1)
89
90   # Initial simulations
91   simWealth.risky[activeIndices] = uStar*initWealth*cumprod(1+drift*delta+
92     volatility*inc[activeIndices])
93   simWealth.riskfree[activeIndices] = (1-uStar)*initWealth*cumprod((1+rent*
94     delta)*ones)
95   simWealth.risky.prime = simWealth.risky[rebPoint]
96   simWealth.riskfree.prime = simWealth.riskfree[rebPoint]
97   transQuantity = (1-uStar)*simWealth.risky.prime - uStar*simWealth.riskfree.
98     prime
99   simWealth.risky[rebPoint] = simWealth.risky.prime - transQuantity
100  simWealth.riskfree[rebPoint] = simWealth.riskfree.prime + transQuantity
101  simWealth[activeIndices] = simWealth.risky[activeIndices] + simWealth.
102    riskfree[activeIndices]
103  propInRisky[activeIndices] = simWealth.risky[activeIndices] / simWealth[
104    activeIndices]
105  propInRiskfree[activeIndices] = simWealth.riskfree[activeIndices] /
106    simWealth[activeIndices]
107
108  # Remainder of simulations
109  for (j in rebIndex[-rebIndex.length] + 1) {
110
111    activeIndices = j:(j+nRebDelay-1)
112    rebPoint = tail(activeIndices,1)
113
114    simWealth.risky[activeIndices] = uStar*simWealth[j-1]*cumprod(1+drift*
115      delta+volatility*inc[activeIndices])
116    simWealth.riskfree[activeIndices] = (1-uStar)*simWealth[j-1]*cumprod((1+
117      rent*delta)*ones)
118    simWealth.risky.prime = simWealth.risky[rebPoint]
119    simWealth.riskfree.prime = simWealth.riskfree[rebPoint]
120    transQuantity = (1-uStar)*simWealth.risky.prime - uStar*simWealth.riskfree
121      .prime
122    simWealth.risky[rebPoint] = simWealth.risky.prime - transQuantity
123    simWealth.riskfree[rebPoint] = simWealth.riskfree.prime + transQuantity
124    simWealth[activeIndices] = simWealth.risky[activeIndices] + simWealth.
125      riskfree[activeIndices]
126    propInRisky[activeIndices] = simWealth.risky[activeIndices] / simWealth[
127      activeIndices]
128    propInRiskfree[activeIndices] = simWealth.riskfree[activeIndices] /
129      simWealth[activeIndices]
130  }
131
132  # Calculation of relevant statistics
133  sdSimWealth = sd(simWealth)
134  simTermWealth = tail(simWealth,1)

```

```

122     simLogReturn = logReturn(c(initWealth, simWealth))
123     sdSimLogReturn = sd(simLogReturn)
124   }
125
126   # nSims > 1 : Compact (rapid) simulation scheme
127   else {
128
129     # Initialization of vectors of relevant statistics
130     sdThWealth = simIndex * NA
131     thTermWealth = simIndex * NA
132     sdThLogReturn = simIndex * NA
133     sdSimWealth = simIndex * NA
134     simTermWealth = simIndex * NA
135     sdSimLogReturn = simIndex * NA
136
137     # Doing nSims simulation runs
138     for (k in 1:nSims) {
139
140       # Brownian increments and motion
141       inc = brownianIncrement(nTimePoints, delta)
142       BM = cumsum(inc)
143
144       # Calculation of theoretical wealth and relevant statistics
145       thWealth = initWealth*exp((drift*uStar+rent*(1-uStar)-.5*volatility^2*
146         uStar^2)*timePoints+volatility*uStar*BM)
147       thTermWealth[k] = tail(thWealth,1)
148       sdThWealth[k] = sd(thWealth)
149       thLogReturn = logReturn(c(initWealth, thWealth))
150       sdThLogReturn[k] = sd(thLogReturn)
151
152       # Initial simulation time points
153       activeIndices = 1:nRebDelay
154       rebPoint = tail(activeIndices,1)
155
156       # Initial simulations of wealth
157       simWealth[activeIndices] = uStar*initWealth*cumprod(1+drift*delta+
158         volatility*inc[activeIndices]) + (1-uStar)*initWealth*cumprod((1+rent*
159         delta)*ones)
160
161       # The rest of the wealthy simulations
162       for (j in rebIndex[-rebIndex.length] + 1) {
163
164         activeIndices = j:(j+nRebDelay-1)
165         rebPoint = tail(activeIndices,1)
166
167         simWealth[activeIndices] = uStar*simWealth[j-1]*cumprod(1+drift*delta+
168           volatility*inc[activeIndices]) + (1-uStar)*simWealth[j-1]*cumprod
169           ((1+rent*delta)*ones)
170       }
171
172     }
173   }
174
175   # Calculation of relevant statistics of last simulation run
176   sdSimWealth[k] = sd(simWealth)
177   simTermWealth[k] = simWealth[lastIndex]
178   simLogReturn = logReturn(c(initWealth, simWealth))
179   sdSimLogReturn[k] = sd(simLogReturn)
180 }
181
182 # Calculation of total simulation time
183 timeElapsed = proc.time()[3][[1]] - timeStart
184 cat(nSims," simulation(s) completed in",timeElapsed," seconds.\n")
185 flush.console()
186
187 # Construction of the list of data to be returned from the function.
188 if (nSims == 1) {

```

```

182     returnList = list(days, rebDays, inc, BM, propInRisky, thWealth, sdThWealth,
183                     thTermWealth, thLogReturn, sdThLogReturn, simWealth, sdSimWealth,
184                     simTermWealth, simLogReturn, sdSimLogReturn)
185     names(returnList) = c("days", "rebDays", "brownianIncrements", "brownianMotion",
186                          "propInRisky", "thWealth", "sdThWealth", "thTermWealth", "thLogReturn",
187                          "sdThLogReturn", "simWealth", "sdSimWealth", "simTermWealth", "simLogReturn",
188                          "sdSimLogReturn")
189   }
190   else {
191     returnList = list(thTermWealth, sdThWealth, sdThLogReturn, simTermWealth,
192                      sdSimWealth, sdSimLogReturn)
193     names(returnList) = c("thTermWealth", "sdThWealth", "sdThLogReturn",
194                          "simTermWealth", "sdSimWealth", "sdSimLogReturn")
195   }
196 }
197 return(returnList)
198 }

```

## B.4.2 Execution

```

1  ##
2  # Master Thesis
3  # Simulation model I
4  # Simulation
5  #
6
7  require(doSMP)
8  source("R/supportFunctions.R")
9  source("R/listArithmetic.R")
10 source("R/machinery_general.R")
11 source("R/machinery_basic.R")
12 source("R/initParameters.R")
13
14 delta = 1 / (nTradingDays*nDailyIncrements)
15 alpha = .05 # Significance level
16 qAlphaHalf = qnorm(1-alpha/2) # 1-alpha/2 percentile of std. norm. dist.
17
18 #
19 # Plot and analysis of one simulation test run (nSims=1)
20 #
21
22 nSims = 1
23 simObject = simPortfolio(nSims, paramSet)
24 save(simObject, file="Datasett/testRun.RData")
25
26 propInRisky = simObject$proportion.in.risky
27 days = simObject$days
28 rebDays = c(simObject$rebDays, 252)
29 thWealth = simObject$thWealth
30 simWealth = simObject$simWealth
31 utilityThWealth = utility(thWealth, riskAversion)
32 utilitySimWealth = utility(simWealth, riskAversion)
33 diffUtility = utilityThWealth - utilitySimWealth
34
35 # Plotting test run
36 xTicks = c(0, rebDays)
37 xTitle = "Trading days"
38 yTitle = "Utility"
39 niceplot(c(0, days), c(initWealth, utilityThWealth), xTicks, xTitle=xTitle, yTitle=
40           yTitle, horizLines=T, col="red")
41 lines(c(0, days), c(initWealth, utilitySimWealth), col="dodgerblue")

```

```

41 abline(v=rebDays, lty=3)
42 savePlot("images/testrun", type="eps")
43 niceplot(c(0, days), c(0, diffUtility), xTicks, xTitle=xTitle, yTitle=yTitle,
44         horizLines=F)
45 abline(v=rebDays, lty=3)
46 savePlot("images/testrun_diff", type="eps")
47 # Plotting proportion in risky
48 yTitle = "Proportion of wealth in risky asset"
49 niceplot(c(0, days), c(uStar, propInRisky), xTicks, xTitle=xTitle, yTitle=yTitle)
50 abline(h=uStar, lty=3)
51 abline(v=rebDays, lty=3)
52 text(0, uStar, "u* = .6811", adj=c(.4, 1.2), offset=.1, cex=.7)
53 savePlot("images/testrunPropInRisky", type="eps")
54
55 # Plotting risky asset, risk-free asset and portfolio values
56 BM = simObject$brownianMotion
57 simRisky = c(1, simRiskyAsset(1, drift, volatility, BM))
58 riskFree = riskFreeAsset(1, rent, 6048)
59 yTitle = "Value"
60 niceplot(c(0, days), simRisky, xTicks, xTitle=xTitle, yTitle=yTitle, horizLines=F, col
61         ="red")
62 abline(v=rebDays, lty=3)
63 nicelines(c(0, days), c(1, riskFree), col="dodgerblue")
64 nicelines(c(0, days), c(1, simWealth))
65 savePlot("images/testrun_wealth", type="eps")
66
67 #
68 # Simulating rebalancing strategy vs loss of utility, 1 mill. runs using
69 # 32 processor cores
70 #
71 # Simulating
72 nSims = 1000000
73 nCores = 32
74 nDailyRebs = c(24, 6, 1, 1/2, 1/12, 1/21, 1/42, 1/126, 1/252)
75 strategyNames = c("Hourly", "Every 4th hour", "Daily", "Every 3rd day", "Every 12th
76                 day", "Monthly", "Bimonthly", "Semiannually", "Annually")
77 paramSets = cbind(initWealth, nTradingDays, nDailyIncrements, nDailyRebs, drift,
78                 volatility, rent, riskAversion, uStar)
79 rebStrategy = distribute(nSims, nCores, simPortfolio, paramSets)
80 names(rebStrategy) = strategyNames
81 save(rebStrategy, file="Datasett/rebStrategy.RData")
82
83 # Calculating mean loss of utility
84 termWealth.th = lapply(rebStrategy, get, x="thTermWealth")
85 termUtility.th = lapply(termWealth.th, utility, param=riskAversion)
86 meanTermUtility.th = sapply(termUtility.th, mean)
87 termWealth.sim = lapply(rebStrategy, get, x="simTermWealth")
88 termUtility.sim = lapply(termWealth.sim, utility, param=riskAversion)
89 meanTermUtility.sim = sapply(termUtility.sim, mean)
90 LOU = listDiff(termUtility.th, termUtility.sim)
91 meansLOU = sapply(LOU, mean)
92 sdsLOU = sapply(LOU, sd)
93 sdMeansLOU = sdsLOU / sqrt(nSims)
94 lowerCL = meansLOU - qAlphaHalf*sdMeansLOU
95 upperCL = meansLOU + qAlphaHalf*sdMeansLOU
96
97 # Plotting mean losses of utility and 95% confidence intervals
98 yMin = min(lowerCL)
99 yMax = max(upperCL)
100 xTicks = 1:9
101 xTitle = "Rebalancing strategy"
102 yTitle = "Mean loss of utility"

```

```

101 niceplot(xTicks, xTitle=xTitle, yTitle=yTitle, meansLOU, xTicks, xLabels=
      strategyNames, ylim=c(yMin, yMax))
102 nicelines(xTicks, lowerCL, lty=3)
103 nicelines(xTicks, upperCL, lty=3)
104 abline(h=0, lty=3)
105 savePlot("images/reb.LOU", type="eps")
106
107 #
108 # Plotting and making histograms
109 #
110
111 # Plotting the losses of utility of the monthly-strategy
112 LOUmonthly = LOU$Monthly
113 save(LOUmonthly, file="Datasett/LOUmonthly.RData")
114 yTitle = "Loss of utility"
115 millionLabels = c(0, "100k", "200k", "300k", "400k", "500k", "600k", "700k", "800k", "900
      k", "1M")
116 xTicks = seq(0, 1000000, 100000)
117 niceplot(LOUmonthly, xTicks=xTicks, xLabels=millionLabels, yTitle=yTitle, downsample
      =T)
118 abline(h=0, lty=3)
119 savePlot("images/LOUmonthly", type="eps")
120 yTitle = "Frequency"
121 xTitle = "Loss of utility"
122 nicehist(LOUmonthly, xTitle=xTitle, yTitle=yTitle, breaks=100, figsPerPage=4)
123 meanLOUmonthly = mean(LOUmonthly)
124 abline(v=meanLOUmonthly, lty=3)
125 savePlot("images/LOUfreqMonthly", type="eps")
126
127 # Histogram of the lower 1 percent of the monthly losses of utility
128 alpha = .05
129 qAlphaLOUmonthly = quantile(LOUmonthly, alpha)
130 nicehist(LOUmonthly[LOUmonthly<=qAlphaLOUmonthly], xTitle=xTitle, yTitle=yTitle,
      breaks=100, figsPerPage=4)
131 savePlot("images/LOUfreqLowerAlphaMonthly", type="eps")
132
133 # Cumulative mean of the losses of utility of the monthly strategy
134 cumMeanLOUmonthly = cumMean(LOUmonthly)
135 yTitle = "Mean loss of utility"
136 niceplot(cumMeanLOUmonthly, xLabels=millionLabels, yTitle=yTitle, figsPerPage=3,
      downsample=T, ylim=c(-.000025, .000025))
137 abline(h=0, lty=3)
138 cumSdMeanLOUmonthly = cumSd(LOUmonthly) / sqrt(nn)
139 qAlphaHalf = qnorm(1-alpha/2)
140 lowerCL = cumMeanLOUmonthly - qAlphaHalf*cumSdMeanLOUmonthly
141 upperCL = cumMeanLOUmonthly + qAlphaHalf*cumSdMeanLOUmonthly
142 nicelines(lowerCL, downsample=T, col="gray", lty=3)
143 nicelines(upperCL, downsample=T, col="gray", lty=3)
144 savePlot("images/meanLOUmonthly", type="eps")
145
146 # Cumulative mean of the losses of utility of the hourly strategy
147 transformation = 1e6
148 x = LOU$Hourly * transformation
149 cum.mean = cumMean(x)
150 nn = 1:length(x)
151 y.title = expression(paste("Mean loss of utility" %s% 10^-6))
152 niceplot(cum.mean, xLabels=millionLabels, yTitle=y.title, y.superscript=T, nCol=2,
      downsample=T, ylim=c(-1, 1))
153 abline(h=0, lty=3)
154 cumMean.sd = cumSd(x) / sqrt(nn)
155 lowerCL = cum.mean - qAlphaHalf*cumMean.sd
156 upperCL = cum.mean + qAlphaHalf*cumMean.sd
157 nicelines(lowerCL, downsample=T, col="gray", lty=3)
158 nicelines(upperCL, downsample=T, col="gray", lty=3)
159 legendText = "(a) Rebalancing strategy: Hourly"

```

```

160 nicelegend("topright", legendText, bty="n", cex=.7, inset=c(.15, 0))
161
162 # Cumulative mean of the losses of utility of the daily strategy
163 transformation = 1e6
164 x = LOU$Daily * transformation
165 cum.mean = cumMean(x)
166 nn = 1:length(x)
167 y.title = expression(paste("Mean loss of utility" %*% 10^-6))
168 niceplot(cum.mean, xLabels=millionLabels, yTitle=y.title, multiPlot=T, newDev=F,
169         downsample=T, ylim=c(-3, 3))
169 abline(h=0, lty=3)
170 cumMean.sd = cumSd(x) / sqrt(nn)
171 lowerCL = cum.mean - qAlphaHalf*cumMean.sd
172 upperCL = cum.mean + qAlphaHalf*cumMean.sd
173 nicelines(lowerCL, downsample=T, col="gray", lty=3)
174 nicelines(upperCL, downsample=T, col="gray", lty=3)
175 legendText = "(b) Rebalancing strategy: Daily"
176 nicelegend("topright", legendText, bty="n", cex=.7, inset=c(.15, 0))
177 savePlot("images/cumMeanLOU.HourlyDaily", type="eps")
178
179 # Cumulative mean of the losses of utility of the 'every 3rd day' strategy
180 transformation = 1e6
181 x = LOU$"Every 3rd day" * transformation
182 cum.mean = cumMean(x)
183 nn = 1:length(x)
184 y.title = expression(paste("Mean loss of utility" %*% 10^-6))
185 niceplot(cum.mean, xLabels=millionLabels, yTitle=y.title, y.superscript=T, nCol=2,
186         downsample=T, ylim=c(-3, 3))
186 abline(h=0, lty=3)
187 cumMean.sd = cumSd(x) / sqrt(nn)
188 lowerCL = cum.mean - qAlphaHalf*cumMean.sd
189 upperCL = cum.mean + qAlphaHalf*cumMean.sd
190 nicelines(lowerCL, downsample=T, col="gray", lty=3)
191 nicelines(upperCL, downsample=T, col="gray", lty=3)
192 legendText = "(c) Rebalancing strategy: Every 3rd day"
193 nicelegend("topright", legendText, bty="n", cex=.7, inset=c(.17, 0))
194
195 # Cumulative mean of the losses of utility of the 'every 12th day' strategy
196 transformation = 1e5
197 x = LOU$"Every 12th day" * transformation
198 cum.mean = cumMean(x)
199 nn = 1:length(x)
200 y.title = expression(paste("Mean loss of utility" %*% 10^-5))
201 niceplot(cum.mean, xLabels=millionLabels, yTitle=y.title, multiPlot=T, newDev=F,
202         downsample=T, ylim=c(-.7, .7))
202 abline(h=0, lty=3)
203 cumMean.sd = cumSd(x) / sqrt(nn)
204 lowerCL = cum.mean - qAlphaHalf*cumMean.sd
205 upperCL = cum.mean + qAlphaHalf*cumMean.sd
206 nicelines(lowerCL, downsample=T, col="gray", lty=3)
207 nicelines(upperCL, downsample=T, col="gray", lty=3)
208 legendText = "(d) Rebalancing strategy: Every 12th day"
209 nicelegend("topright", legendText, bty="n", cex=.7, inset=c(.18, 0))
210 savePlot("images/cumMeanLOU.3rd12th", type="eps")
211
212 # Cumulative mean of the losses of utility of the monthly strategy
213 transformation = 1e5
214 x = LOU$Monthly * transformation
215 cum.mean = cumMean(x)
216 nn = 1:length(x)
217 y.title = expression(paste("Mean loss of utility" %*% 10^-5))
218 niceplot(cum.mean, xLabels=millionLabels, yTitle=y.title, y.superscript=T, nCol=2,
219         downsample=T, ylim=c(-1, 1))
219 abline(h=0, lty=3)
220 cumMean.sd = cumSd(x) / sqrt(nn)

```



```

221 lowerCL = cum.mean - qAlphaHalf*cumMean.sd
222 upperCL = cum.mean + qAlphaHalf*cumMean.sd
223 nicelines(lowerCL,downsample=T,col="gray",lty=3)
224 nicelines(upperCL,downsample=T,col="gray",lty=3)
225 legendText = "(e) Rebalancing strategy: Monthly"
226 nicelegend("topright",legendText,bty="n",cex=.7,inset=c(.15,0))
227
228 # Cumulative mean of the losses of utility of the bimonthly strategy
229 transformation = 1e5
230 x = LOU$Bimonthly * transformation
231 cum.mean = cumMean(x)
232 nn = 1:length(x)
233 y.title = expression(paste("Mean loss of utility" %*% 10^-5))
234 niceplot(cum.mean,xLabels=millionLabels,yTitle=y.title,multiPlot=T,newDev=F,
235         downsample=T,ylim=c(-1.2,1.2))
236 abline(h=0,lty=3)
237 cumMean.sd = cumSd(x) / sqrt(nn)
238 lowerCL = cum.mean - qAlphaHalf*cumMean.sd
239 upperCL = cum.mean + qAlphaHalf*cumMean.sd
240 nicelines(lowerCL,downsample=T,col="gray",lty=3)
241 nicelines(upperCL,downsample=T,col="gray",lty=3)
242 legendText = "(f) Rebalancing strategy: Bimonthly"
243 nicelegend("topright",legendText,bty="n",cex=.7,inset=c(.16,0))
244 savePlot("images/cumMeanLOU_MonthlyBi",type="eps")
245
246 # Cumulative mean of the losses of utility of the semiannual strategy
247 transformation = 1e5
248 x = LOU$"Half-yearly" * transformation
249 cum.mean = cumMean(x)
250 nn = 1:length(x)
251 y.title = expression(paste("Mean loss of utility" %*% 10^-5))
252 niceplot(cum.mean,xLabels=millionLabels,yTitle=y.title,y.superscript=T,nCol=2,
253         downsample=T,ylim=c(-3,3))
254 abline(h=0,lty=3)
255 cumMean.sd = cumSd(x) / sqrt(nn)
256 lowerCL = cum.mean - qAlphaHalf*cumMean.sd
257 upperCL = cum.mean + qAlphaHalf*cumMean.sd
258 nicelines(lowerCL,downsample=T,col="gray",lty=3)
259 nicelines(upperCL,downsample=T,col="gray",lty=3)
260 legendText = "(g) Rebalancing strategy: Semiannually"
261 nicelegend("topright",legendText,bty="n",cex=.7,inset=c(.18,0))
262
263 # Cumulative mean of the losses of utility of the annual strategy
264 transformation = 1e5
265 x = LOU$Yearly * transformation
266 cum.mean = cumMean(x)
267 nn = 1:length(x)
268 y.title = expression(paste("Mean loss of utility" %*% 10^-5))
269 niceplot(cum.mean,xLabels=millionLabels,yTitle=y.title,multiPlot=T,newDev=F,
270         downsample=T,ylim=c(-5,5))
271 abline(h=0,lty=3)
272 cumMean.sd = cumSd(x) / sqrt(nn)
273 lowerCL = cum.mean - qAlphaHalf*cumMean.sd
274 upperCL = cum.mean + qAlphaHalf*cumMean.sd
275 nicelines(lowerCL,downsample=T,col="gray",lty=3)
276 nicelines(upperCL,downsample=T,col="gray",lty=3)
277 legendText = "(h) Rebalancing strategy: Annually"
278 nicelegend("topright",legendText,bty="n",cex=.7,inset=c(.15,0))
279 savePlot("images/cumMeanLOU_AnnuallySemi",type="eps")
280
281 #
282 # Rebalancing strategy vs Sharpe ratio
283 #
284
285 nTimePoints = 6048

```

```

283 expecLogReturn = expectedLogReturn(drift , volatility , rent , uStar ,1)
284 sdLogReturn = stDevLogReturn( volatility , uStar ,1)
285 exAnteSR = exAnteSharpeRatio(drift , volatility , rent , uStar ,1)
286 logAdjustedSR = logAdjustedSharpeRatio(drift , volatility , rent , uStar ,1)
287
288 # Calculating Sharpe ratios of theoretical portfolios
289 strategy_termWealth.th = sapply(rebStrategy , get , x="thTermWealth")
290 strategy_meanTermWealth.th = colMeans(strategy_termWealth.th)
291 strategy_logReturn.th = log(strategy_termWealth.th)
292 strategy_meanLogReturn.th = colMeans(strategy_logReturn.th)
293 strategy_excessReturn.th = strategy_logReturn.th - rent
294 strategy_meanExcessReturn.th = colMeans(strategy_excessReturn.th)
295 strategy_sdLogReturn.th = sapply(rebStrategy , get , x="sdThLogReturn")
296 strategy_meanSdLogReturn.th = colMeans(strategy_sdLogReturn.th)
297 strategy_annualizedSdLogReturn.th = strategy_sdLogReturn.th * sqrt(nTimePoints)
298 strategy_meanAnnualizedSdLogReturn.th = colMeans(strategy_annualizedSdLogReturn.
    th)
299 strategy_volOfVol.th = colSds(strategy_annualizedSdLogReturn.th)
300 strategy_correlation.th = colCorrs(strategy_logReturn.th , strategy_sdLogReturn.th
    )
301 strategy_SR.th = strategy_excessReturn.th / (sqrt(nTimePoints)*
    strategy_sdLogReturn.th)
302 save(strategy_SR.th , file="Datasett/strategy_SR.th.RData")
303 strategy_meanSR.th = colMeans(strategy_SR.th)
304 strategy_sdSR.th = colSds(strategy_SR.th)
305 strategy_sdMeanSR.th = strategy_sdSR.th / sqrt(nrow(strategy_SR.th))
306
307 # Calculating Sharpe ratios of simulated portfolios
308 strategy_termWealth.sim = sapply(rebStrategy , get , x="simTermWealth")
309 strategy_meanTermWealth.sim = colMeans(strategy_termWealth.sim)
310 strategy_lossOfWealth.sim = strategy_termWealth.th - strategy_termWealth.sim
311 strategy_meanLossOfWealth.sim = colMeans(strategy_lossOfWealth.sim)
312 strategy_logReturn.sim = log(strategy_termWealth.sim)
313 strategy_meanLogReturn.sim = colMeans(strategy_logReturn.sim)
314 strategy_sdTermLogReturn.sim = colSds(strategy_logReturn.sim)
315 strategy_excessReturn.sim = strategy_logReturn.sim - rent
316 strategy_meanExcessReturn.sim = colMeans(strategy_excessReturn.sim)
317 strategy_sdLogReturn.sim = sapply(rebStrategy , get , x="sdSimLogReturn")
318 strategy_meanSdLogReturn.sim = colMeans(strategy_sdLogReturn.sim)
319 strategy_annualizedSdLogReturn.sim = strategy_sdLogReturn.sim * sqrt(nTimePoints
    )
320 strategy_meanAnnualizedSdLogReturn.sim = colMeans(strategy_annualizedSdLogReturn
    .sim)
321 strategy_volOfVol.sim = colSds(strategy_annualizedSdLogReturn.sim)
322 strategy_correlation.sim = colCorrs(strategy_logReturn.sim , strategy_sdLogReturn.
    sim)
323 strategy_SR.sim = strategy_excessReturn.sim / (strategy_annualizedSdLogReturn.
    sim)
324 save(strategy_SR.sim , file="Datasett/strategy_SR.sim.RData")
325 strategy_meanSR.sim = colMeans(strategy_SR.sim)
326 strategy_ranking.sim = c(1,3,2,4,5,6,7,8,9)
327 strategy_sdSR.sim = colSds(strategy_SR.sim)
328 strategy_sdMeanSR.sim = strategy_sdSR.sim / sqrt(nrow(strategy_SR.sim))
329 strategy_testStat.sim = (strategy_meanSR.sim - exAnteSR) / strategy_sdMeanSR.sim
330 strategy_pValue.sim = 2*pnorm(-abs(strategy_testStat.sim))
331
332 # Calculating confidence intervals of the mean Sharpe ratios
333 strategy_lowerCL.sim = strategy_meanSR.sim - qAlphaHalf*strategy_sdMeanSR.sim
334 strategy_upperCL.sim = strategy_meanSR.sim + qAlphaHalf*strategy_sdMeanSR.sim
335
336 # Creating tables for printout
337 tab1 = matrix(NA,18,5)
338 for (i in 1:9) {
339   tab1[2*i-1,] = c(strategy_meanTermWealth.th[i] , 0 , meanTermUtility.th[i] , 0 , 0)

```

```

340     tab1[2*i,] = c(strategy_meanTermWealth.sim[i], strategy_meanLossOfWealth.sim[i]
      ], meanTermUtility.sim[i], meansLOU[i], sdsLOU[i])
341 }
342 tab1[,1] = round(tab1[,1],4)
343 tab1[,2] = round(tab1[,2]*1e5,4)
344 tab1[,3] = round(tab1[,3],4)
345 tab1[,4] = round(tab1[,4]*1e5,4)
346 tab1[,5] = round(tab1[,5]*1e3,4)
347
348 for (k in 1:18) {
349     tab1[k,2] = paste(tab1[k,2], "\\e{\\text{-5}}", sep="")
350     tab1[k,4] = paste(tab1[k,4], "\\e{\\text{-5}}", sep="")
351     tab1[k,5] = paste(tab1[k,5], "\\e{\\text{-3}}", sep="")
352 }
353
354 printex(tab1)
355
356 tab2 = matrix(NA,18,5)
357 for (i in 1:9) {
358     tab2[2*i-1,] = c(strategy_meanLogReturn.th[i],
      strategy_meanAnnualizedSdLogReturn.th[i], strategy_meanSR.th[i],
      strategy_volOfVol.th[i], strategy_correlation.th[i])
359     tab2[2*i,] = c(strategy_meanLogReturn.sim[i],
      strategy_meanAnnualizedSdLogReturn.sim[i], strategy_meanSR.sim[i],
      strategy_volOfVol.sim[i], strategy_correlation.sim[i])
360 }
361 colnames(tab2) = c("meanLogRet", "annMeanSdLogRet", "meanSR", "volOfVol", "corr")
362 tab2[,1] = round(tab2[,1]*1e2,4)
363 tab2[,2] = round(tab2[,2],4)
364 tab2[,3] = round(tab2[,3]*1e2,4)
365 tab2[,4] = round(tab2[,4]*1e2,4)
366 tab2[,5] = round(tab2[,5],4)
367
368 for (k in 1:18) {
369     tab2[k,1] = paste(tab2[k,1], "\\e{\\text{-2}}", sep="")
370     tab2[k,3] = paste(tab2[k,3], "\\e{\\text{-2}}", sep="")
371     tab2[k,4] = paste(tab2[k,4], "\\e{\\text{-2}}", sep="")
372 }
373
374 printex(tab2)
375
376 # Plotting mean Sharpe ratios vs rebalancing strategies
377 xTicks = 1:9
378 xTitle = "Rebalancing strategy"
379 yTitle = "Sharpe ratio"
380 yMin = min(c(0, strategy_lowerCL.sim))
381 yMax = max(strategy_upperCL.sim)
382 niceplot(xTicks, xTitle=xTitle, yTitle=yTitle, strategy_meanSR.sim, xTicks, xLabels=
      strategyNames, ylim=c(yMin, yMax))
383 abline(v=xTicks, col="gray", lty=3)
384 nicelines(strategy_lowerCL.sim, lty=2)
385 nicelines(strategy_upperCL.sim, lty=2)
386 abline(h=exAnteSR, lty=3)
387 text(8.39, exAnteSR, paste("ex ante Sharpe ratio =", round(exAnteSR,4)), pos=1,
      offset=.2, cex=.7)
388 savePlot("images/exPostSharpeRatio", type="eps")
389
390 # Histogram of the losses of utility of the hourly-strategy superimposed
391 # on a histogram of the losses of utility of the annual-strategy.
392 histObject = hist(strategy_annualizedSdLogReturn.sim[,9], breaks=100, plot=F)
393 breakPoints = histObject$breaks
394 histObject = hist(strategy_annualizedSdLogReturn.sim[,1], breaks=breakPoints, plot
      =F)
395 yTitle = "Frequency"
396 xTitle = "Standard deviation"

```

```

397 yMin = 0
398 yMax = max(histObject$counts)
399 yRange = c(yMin,yMax)
400 nicehist(strategy_annualizedSdLogReturn.sim[,9], xTitle=xTitle, yTitle=yTitle,
           breaks=breakPoints, ylim=yRange)
401 addHist(strategy_annualizedSdLogReturn.sim[,1], density=30)
402 savePlot("images/sdHourlyVsAnnually", type="eps")
403
404 strategy_sdAnnualizedSdLogReturn.sim = colSds(strategy_annualizedSdLogReturn.sim
         )
405 strategy_corrLogReturnSdLogReturn.sim = colCorrs(strategy_logReturn.sim,
           strategy_annualizedSdLogReturn.sim)
406 tab2 = cbind(strategy_meanAnnualizedSdLogReturn.sim,
           strategy_sdAnnualizedSdLogReturn.sim, strategy_corrLogReturnSdLogReturn.sim)
407 tab2[,2] = tab2[,2] * 1e2
408 rownames(tab2) = strategyNames
409 printex(trimLeadingZero(tab2))
410
411 # Histograms of losses of utility of the hourly and the daily strategy
412 x.title = "Loss of utility"
413 y.title = "Frequency"
414 breaksLength = 70
415 res = seq(min(LOU$Hourly), max(LOU$Hourly), length=breaksLength)
416 histObject = hist(LOU$Hourly, breaks=res, plot=F)
417 y.lim = range(histObject$counts) * 1.1
418 nicehist(LOU$Hourly, xTitle=x.title, yTitle=y.title, nCol=2, ylim=y.lim, breaks=res)
419 legendText = c("(a) Rebalancing strategy: Hourly", expression(paste("Mean =
           -.0300" %*% 10^-5)), expression(paste("StDev = .1471" %*% 10^-3)))
420 nicelegend("topleft", legendText, bty="n", cex=.7)
421 res = seq(min(LOU$Daily), max(LOU$Daily), length=breaksLength)
422 histObject = hist(LOU$Daily, breaks=res, plot=F)
423 y.lim = range(histObject$counts) * 1.1
424 nicehist(LOU$Daily, xTitle=x.title, yTitle=y.title, multiPlot=T, newDev=F, ylim=y.lim
           , breaks=res)
425 legendText = c("(b) Rebalancing strategy: Daily", expression(paste("Mean = .0670"
           %*% 10^-5)), expression(paste("StDev = .3623" %*% 10^-3)))
426 nicelegend("topleft", legendText, bty="n", cex=.7)
427 savePlot("images/histLouHourlyDaily", type="eps")
428
429 # Histograms of losses of utility of the 'every 3rd day'-strategy and the
430 # 'every 12th day'-strategy
431 res = seq(min(LOU$"Every 3rd day"), max(LOU$"Every 3rd day"), length=breaksLength)
432 histObject = hist(LOU$"Every 3rd day", breaks=res, plot=F)
433 y.lim = range(histObject$counts) * 1.1
434 nicehist(LOU$"Every 3rd day", xTitle=x.title, yTitle=y.title, nCol=2, ylim=y.lim,
           breaks=res)
435 legendText = c("(c) Rebalancing strategy: Every 3rd day", expression(paste("Mean
           = .0414" %*% 10^-5)), expression(paste("StDev = .4947" %*% 10^-3)))
436 nicelegend("topleft", legendText, bty="n", cex=.7)
437 res = seq(min(LOU$"Every 12th day"), max(LOU$"Every 12th day"), length=
           breaksLength)
438 histObject = hist(LOU$"Every 12th day", breaks=res, plot=F)
439 y.lim = range(histObject$counts) * 1.1
440 nicehist(LOU$"Every 12th day", xTitle=x.title, yTitle=y.title, multiPlot=T, newDev=F
           , ylim=y.lim, breaks=res)
441 legendText = c("(d) Rebalancing strategy: Every 12th day", expression(paste("Mean
           = .1424" %*% 10^-5)), expression(paste("StDev = 1.1764" %*% 10^-3)))
442 nicelegend("topleft", legendText, bty="n", cex=.7)
443 savePlot("images/histLou3rd12th", type="eps")
444
445 # Histograms of losses of utility of the monthly and the bimonthly strategy
446 res = seq(min(LOU$Monthly), max(LOU$Monthly), length=breaksLength)
447 histObject = hist(LOU$Monthly, breaks=res, plot=F)
448 y.lim = range(histObject$counts) * 1.1
449 nicehist(LOU$Monthly, xTitle=x.title, yTitle=y.title, nCol=2, ylim=y.lim, breaks=res)

```

```

450 | legendText = c("(e) Rebalancing strategy: Monthly", expression(paste("Mean =
      | -.1769" %*% 10^-5)), expression(paste("StDev = 1.5515" %*% 10^-3)))
451 | nicelegend("topleft", legendText, bty="n", cex=.7)
452 | res = seq(min(LOU$Bimonthly), max(LOU$Bimonthly), length=breaksLength)
453 | histObject = hist(LOU$Bimonthly, breaks=res, plot=F)
454 | y.lim = range(histObject$counts) * 1.1
455 | nicehist(LOU$Bimonthly, xTitle=x.title, yTitle=y.title, multiPlot=T, newDev=F, ylim=y
      | .lim, breaks=res)
456 | legendText = c("(f) Rebalancing strategy: Bimonthly", expression(paste("Mean =
      | .2887" %*% 10^-5)), expression(paste("StDev = 2.1899" %*% 10^-3)))
457 | nicelegend("topleft", legendText, bty="n", cex=.7)
458 | savePlot("images/histLouMonthlyBi", type="eps")
459 |
460 | # Histograms of losses of utility of the semi-annual and the annual strategy
461 | res = seq(min(LOU$Half-yearly), max(LOU$Half-yearly), length=breaksLength)
462 | histObject = hist(LOU$Half-yearly, breaks=res, plot=F)
463 | y.lim = range(histObject$counts) * 1.1
464 | nicehist(LOU$Half-yearly, xTitle=x.title, yTitle=y.title, nCol=2, ylim=y.lim,
      | breaks=res)
465 | legendText = c("(g) Rebalancing strategy: Semiannually", expression(paste("Mean =
      | 1.2969" %*% 10^-5)), expression(paste("StDev = 3.7678" %*% 10^-3)))
466 | nicelegend("topleft", legendText, bty="n", cex=.7)
467 | res = seq(min(LOU$Yearly), max(LOU$Yearly), length=breaksLength)
468 | histObject = hist(LOU$Yearly, breaks=res, plot=F)
469 | y.lim = range(histObject$counts) * 1.1
470 | nicehist(LOU$Yearly, xTitle=x.title, yTitle=y.title, multiPlot=T, newDev=F, ylim=y.
      | lim, breaks=res)
471 | legendText = c("(h) Rebalancing strategy: Annually", expression(paste("Mean =
      | 2.0172" %*% 10^-5)), expression(paste("StDev = 5.3237" %*% 10^-3)))
472 | nicelegend("topleft", legendText, bty="n", cex=.7)
473 | savePlot("images/histLouHalfYearly", type="eps")

```

## B.5 Simulation model II and III

### B.5.1 Simulation machinery

```

1 | ##
2 | # Master Thesis
3 | # Simulation model II and III
4 | # Simulation algorithm
5 | #
6 |
7 | simPortfolio.transCost = function(nSims, paramSet, brownianFileName=NULL) {
8 |   #
9 |   # Simulates nSims portfolios following the 9 parameter values of paramSet
10 |  # and returns terminal utilities of theoretical and simulated wealth and
11 |  # the loss of utility. Includes transaction costs!
12 |  #
13 |  logReturn = function(x) {
14 |    #
15 |    # Computes the log returns of a time series x.
16 |    #
17 |    n = length(x)
18 |    xUp = x[2:n]
19 |    xLow = x[1:(n-1)]
20 |    logReturns = log(xUp/xLow)
21 |    return(logReturns)
22 |  }

```

```

23 |
24 | brownianIncrement = function(n, delta) {
25 |   #
26 |   # Simulates random series of n brownian increments with variance delta.
27 |   #
28 |   return(rnorm(n,0,sqrt(delta)))
29 | }
30 |
31 | #
32 | # Assigning variables.
33 | #
34 | nParams = length(paramSet)
35 | if (nParams != 10) stop(paste("Number of input parameters equals ",nParams,".
      |   Must equal 10.",sep=""))
36 | varNames = c("initWealth","nTradingDays","nDailyIncrements","nDailyRebs","
      |   drift","volatility","rent","riskAversion","uStar","costProp")
37 | for (j in 1:10) { assign(varNames[j],paramSet[j]) }
38 |
39 | #
40 | # Initializing the simulation structure.
41 | #
42 |
43 | simIndex = 1:nSims
44 | nTimePoints = nTradingDays * nDailyIncrements
45 | lastIndex = nTimePoints
46 | delta = 1 / nTimePoints
47 | timePoints = seq(delta,1,delta)
48 | nRebDelay = nDailyIncrements / nDailyRebs
49 | rebIndex = seq(nRebDelay,nTimePoints,nRebDelay)
50 | #rebIndex = rebIndex[-length(rebIndex)]
51 | days = seq(delta*nTradingDays,nTradingDays,delta*nTradingDays)
52 | rebDays = days[rebIndex]
53 | ones = rep(1,nRebDelay)
54 |
55 | # Start of simulation time
56 | timeStart = proc.time()[3][[1]]
57 |
58 | # Common structure
59 | simWealth = NA
60 | simWealth.pre = NA
61 | simWealth.sub = NA
62 |
63 | #
64 | # Using full simulation scheme if nSims = 1
65 | #
66 |
67 | if (nSims == 1) {
68 |
69 |   # Intializing other statistics
70 |   riskyReturn = 1:nTimePoints * NA
71 |   riskfreeReturn = 1:nTimePoints * NA
72 |
73 |   # Intializing simulated wealth without transaction costs
74 |   simWealth.risky = NA
75 |   simWealth.riskfree = NA
76 |   transQuantity = 1:nTimePoints * 0
77 |   propInRisky = NA
78 |   propInRiskfree = NA
79 |
80 |   # Intializing simulated wealth with preceding transaction costs
81 |   simWealth.risky.pre = NA
82 |   simWealth.riskfree.pre = NA
83 |   transQuantity.pre = 1:nTimePoints * 0
84 |   transCost.pre = 1:nTimePoints * 0
85 |   propInRisky.pre = NA

```

```

86   propInRiskfree.pre = NA
87
88   # Intializing simulated wealth with subsequent transaction costs
89   simWealth.risky.sub = NA
90   simWealth.riskfree.sub = NA
91   transQuantity.sub = 1:nTimePoints * 0
92   transCost.sub = 1:nTimePoints * 0
93   propInRisky.sub = NA
94   propInRiskfree.sub = NA
95
96   # Generation of Brownian motion
97   if (!is.null(brownianFileName) && file.exists(brownianFileName, sep="")) {
98     cat(" Loading brownian increments...\n"); load(brownianFileName) }
99   else { inc = brownianIncrement(nTimePoints, delta) }
100  if (!is.null(brownianFileName) && !file.exists(brownianFileName)) { cat("
101    Saving brownian increments...\n"); save(inc, file=brownianFileName) }
102  if (exists("dualInc")) { inc = dualInc[,1] }
103  BM = cumsum(inc)
104
105  # Initialization and calculation of theoretical wealth
106  initRiskyPrice = 1
107  riskyPrice = initRiskyPrice*exp((drift - .5*volatility^2)*timePoints+
108    volatility*BM)
109  initRiskfreePrice = 1
110  riskfreePrice = initRiskfreePrice*exp(rent*timePoints)
111  thWealth = initWealth*exp((drift*uStar+rent*(1-uStar) - .5*volatility^2*uStar
112    ^2)*timePoints+volatility*uStar*BM)
113
114  #
115  # First part of the simulations
116  #
117
118  # Time points to be simulated
119  activeIndices = 1:nRebDelay
120  rebPoint = tail(activeIndices, 1)
121
122  # Calculating risky and risk free returns
123  riskyReturn[activeIndices] = cumprod(1+drift*delta+volatility*inc[
124    activeIndices]) - 1
125  riskfreeReturn[activeIndices] = cumprod((1+rent*delta)*ones) - 1
126
127  # Without transaction costs
128  simWealth.risky = uStar*initWealth*cumprod(1+drift*delta+volatility*inc[
129    activeIndices])
130  simWealth.riskfree = (1-uStar)*initWealth*cumprod((1+rent*delta)*ones)
131  simWealth = simWealth.risky + simWealth.riskfree
132  simWealth.risky.prime = simWealth.risky[rebPoint]
133  simWealth.riskfree.prime = simWealth.riskfree[rebPoint]
134  transQuantity[rebPoint] = ((1-uStar)*simWealth.risky.prime - uStar*simWealth
135    .riskfree.prime)
136  simWealth.risky[rebPoint] = simWealth.risky.prime - transQuantity[rebPoint]
137  simWealth.riskfree[rebPoint] = simWealth.riskfree.prime + transQuantity[
138    rebPoint]
139  simWealth[rebPoint] = simWealth.risky[rebPoint] + simWealth.riskfree[
140    rebPoint]
141  propInRisky[activeIndices] = simWealth.risky / simWealth
142  propInRiskfree[activeIndices] = simWealth.riskfree / simWealth
143
144  # Preceding transaction costs
145  simWealth.risky.pre = uStar*initWealth*cumprod(1+drift*delta+volatility*inc[
146    activeIndices])
147  simWealth.riskfree.pre = (1-uStar)*initWealth*cumprod((1+rent*delta)*ones)
148  simWealth.pre = simWealth.risky.pre + simWealth.riskfree.pre
149  simWealth.risky.pre.prime = simWealth.risky.pre[rebPoint]
150  simWealth.riskfree.pre.prime = simWealth.riskfree.pre[rebPoint]

```

```

141   signDiffReturn.pre = sign(prod(1+drift*delta+volatility*inc[activeIndices])
142     - prod((1+rent*delta)*ones))
143   transQuantity.pre[rebPoint] = ((1-uStar)*simWealth.risky.pre.prime - uStar*
144     simWealth.riskfree.pre.prime) / (1-signDiffReturn.pre*costProp*uStar)
145   transCost.pre[rebPoint] = abs(costProp*transQuantity.pre[rebPoint])
146   simWealth.risky.pre[rebPoint] = simWealth.risky.pre.prime - transQuantity.
147     pre[rebPoint]
148   simWealth.riskfree.pre[rebPoint] = simWealth.riskfree.pre.prime +
149     transQuantity.pre[rebPoint] - costProp*abs(transQuantity.pre[rebPoint])
150   simWealth.pre[rebPoint] = simWealth.risky.pre[rebPoint] + simWealth.riskfree
151     .pre[rebPoint]
152   propInRisky.pre[activeIndices] = simWealth.risky.pre / simWealth.pre
153   propInRiskfree.pre[activeIndices] = simWealth.riskfree.pre / simWealth.pre
154
155   # Subsequent transaction costs
156   simWealth.risky.sub = simWealth.risky.pre
157   simWealth.riskfree.sub = simWealth.riskfree.pre
158   simWealth.sub = simWealth.pre
159   simWealth.risky.sub.prime = simWealth.risky.pre.prime
160   simWealth.riskfree.sub.prime = simWealth.riskfree.pre.prime
161   transQuantity.sub[rebPoint] = (1-uStar)*simWealth.risky.sub.prime - uStar*
162     simWealth.riskfree.sub.prime
163   transCost.sub[rebPoint] = abs(costProp*transQuantity.sub[rebPoint])
164   simWealth.risky.sub[rebPoint] = simWealth.risky.sub.prime - transQuantity.
165     sub[rebPoint]
166   simWealth.riskfree.sub[rebPoint] = simWealth.riskfree.sub.prime +
167     transQuantity.sub[rebPoint] - costProp*abs(transQuantity.sub[rebPoint])
168   simWealth.sub[rebPoint] = simWealth.risky.sub[rebPoint] + simWealth.riskfree
169     .sub[rebPoint]
170   propInRisky.sub[activeIndices] = simWealth.risky.sub / simWealth.sub
171   propInRiskfree.sub[activeIndices] = simWealth.riskfree.sub / simWealth.sub
172
173   for (j in rebIndex[-length(rebIndex)] + 1) {
174     activeIndices = j:(j+nRebDelay-1)
175     rebPoint = tail(activeIndices,1)
176
177     # Calculating risky and risk free returns
178     riskyReturn[activeIndices] = cumprod(1+drift*delta+volatility*inc[
179       activeIndices]) - 1
180     riskfreeReturn[activeIndices] = cumprod((1+rent*delta)*ones) - 1
181
182     # Without transaction costs
183     simWealth.risky[activeIndices] = uStar*simWealth[j-1]*cumprod(1+drift*
184       delta+volatility*inc[activeIndices])
185     simWealth.riskfree[activeIndices] = (1-uStar)*simWealth[j-1]*cumprod((1+
186       rent*delta)*ones)
187     simWealth[activeIndices] = simWealth.risky[activeIndices] + simWealth.
188       riskfree[activeIndices]
189     simWealth.risky.prime = simWealth.risky[rebPoint]
190     simWealth.riskfree.prime = simWealth.riskfree[rebPoint]
191     transQuantity[rebPoint] = ((1-uStar)*simWealth.risky.prime - uStar*
192       simWealth.riskfree.prime)
193     simWealth.risky[rebPoint] = simWealth.risky.prime - transQuantity[rebPoint
194       ]
195     simWealth.riskfree[rebPoint] = simWealth.riskfree.prime + transQuantity[
196       rebPoint]
197     simWealth[rebPoint] = simWealth.risky[rebPoint] + simWealth.riskfree[
198       rebPoint]
199     propInRisky[activeIndices] = simWealth.risky[activeIndices] / simWealth[
200       activeIndices]
201     propInRiskfree[activeIndices] = simWealth.riskfree[activeIndices] /
202       simWealth[activeIndices]
203
204     # Preceding transaction costs

```



```

186     simWealth.risky.pre[activeIndices] = uStar*simWealth.pre[j-1]*cumprod(1+
187       drift*delta+volatility*inc[activeIndices])
188     simWealth.riskfree.pre[activeIndices] = (1-uStar)*simWealth.pre[j-1]*
189       cumprod((1+rent*delta)*ones)
190     simWealth.pre[activeIndices] = simWealth.risky.pre[activeIndices] +
191       simWealth.riskfree.pre[activeIndices]
192     simWealth.risky.pre.prime = simWealth.risky.pre[rebPoint]
193     simWealth.riskfree.pre.prime = simWealth.riskfree.pre[rebPoint]
194     signDiffReturn.pre = sign(prod(1+drift*delta+volatility*inc[activeIndices]
195       )) - prod((1+rent*delta)*ones))
196     transQuantity.pre[rebPoint] = ((1-uStar)*simWealth.risky.pre.prime - uStar
197       *simWealth.riskfree.pre.prime) / (1-signDiffReturn.pre*costProp*uStar)
198     transCost.pre[rebPoint] = abs(costProp*transQuantity.pre[rebPoint])
199     simWealth.risky.pre[rebPoint] = simWealth.risky.pre.prime - transQuantity.
200       pre[rebPoint]
201     simWealth.riskfree.pre[rebPoint] = simWealth.riskfree.pre.prime +
202       transQuantity.pre[rebPoint] - costProp*abs(transQuantity.pre[rebPoint]
203       ])
204     simWealth.pre[rebPoint] = simWealth.risky.pre[rebPoint] + simWealth.
205       riskfree.pre[rebPoint]
206     propInRisky.pre[activeIndices] = simWealth.risky.pre[activeIndices] /
207       simWealth.pre[activeIndices]
208     propInRiskfree.pre[activeIndices] = simWealth.riskfree.pre[activeIndices]
209       / simWealth.pre[activeIndices]
210
211     # Subsequent transaction costs
212     simWealth.risky.sub[activeIndices] = uStar*simWealth.sub[j-1]*cumprod(1+
213       drift*delta+volatility*inc[activeIndices])
214     simWealth.riskfree.sub[activeIndices] = (1-uStar)*simWealth.sub[j-1]*
215       cumprod((1+rent*delta)*ones)
216     simWealth.sub[activeIndices] = simWealth.risky.sub[activeIndices] +
217       simWealth.riskfree.sub[activeIndices]
218     simWealth.risky.sub.prime = simWealth.risky.sub[rebPoint]
219     simWealth.riskfree.sub.prime = simWealth.riskfree.sub[rebPoint]
220     transQuantity.sub[rebPoint] = (1-uStar)*simWealth.risky.sub.prime - uStar*
221       simWealth.riskfree.sub.prime
222     transCost.sub[rebPoint] = abs(costProp*transQuantity.sub[rebPoint])
223     simWealth.risky.sub[rebPoint] = simWealth.risky.sub.prime - transQuantity.
224       sub[rebPoint]
225     simWealth.riskfree.sub[rebPoint] = simWealth.riskfree.sub.prime +
226       transQuantity.sub[rebPoint] - costProp*abs(transQuantity.sub[rebPoint]
227       ])
228     simWealth.sub[rebPoint] = simWealth.risky.sub[rebPoint] + simWealth.
229       riskfree.sub[rebPoint]
230     propInRisky.sub[activeIndices] = simWealth.risky.sub[activeIndices] /
231       simWealth.sub[activeIndices]
232     propInRiskfree.sub[activeIndices] = simWealth.riskfree.sub[activeIndices]
233       / simWealth.sub[activeIndices]
234   }
235 }
236
237 #
238 # Using compact form of simulation scheme if nSims > 1
239 #
240
241 else {
242   thWealth.sd = simIndex * NA
243   thWealth.terminal = simIndex * NA
244   thWealth.logReturn.sd = simIndex * NA
245
246   simWealth.sd = simIndex * NA
247   simWealth.terminal = simIndex * NA
248   simWealth.logReturn.sd = simIndex * NA
249 }

```

```

230   simWealth.pre.sd = simIndex * NA
231   simWealth.pre.terminal = simIndex * NA
232   simWealth.pre.logReturn.sd = simIndex * NA
233   totalTransCost.pre = simIndex * 0
234
235   simWealth.sub.sd = simIndex * NA
236   simWealth.sub.terminal = simIndex * NA
237   simWealth.sub.logReturn.sd = simIndex * NA
238   totalTransCost.sub = simIndex * 0
239
240   for (k in 1:nSims) {
241
242     # Generation of Brownian motion
243     inc = brownianIncrement(nTimePoints, delta)
244     BM = cumsum(inc)
245
246     # Initialization and calculation of theoretical wealth
247     thWealth = initWealth*exp((drift*uStar+rent*(1-uStar)-.5*volatility^2*
248       uStar^2)*timePoints+volatility*uStar*BM)
249
250     #
251     # Simulated wealths until first rebalancing time point
252     #
253     # Common quantities
254     activeIndices = 1:nRebDelay
255     rebPoint = tail(activeIndices,1)
256     return.risky = prod(1+drift*delta+volatility*inc[activeIndices])
257     return.riskfree = prod((1+rent*delta)*ones)
258     diffReturn = return.risky - return.riskfree
259
260     # No transaction costs
261     simWealth[activeIndices] = uStar*initWealth*cumprod(1+drift*delta+
262       volatility*inc[activeIndices]) + (1-uStar)*initWealth*cumprod((1+rent*
263       delta)*ones)
264
265     # Preceding transaction costs
266     simWealth.pre[activeIndices] = uStar*initWealth*cumprod(1+drift*delta+
267       volatility*inc[activeIndices]) + (1-uStar)*initWealth*cumprod((1+rent*
268       delta)*ones)
269     signDiffReturn = sign(diffReturn)
270     transCost.pre = costProp * abs((uStar*(1-uStar)*initWealth*diffReturn) /
271       (1-signDiffReturn*costProp*uStar))
272     totalTransCost.pre[k] = totalTransCost.pre[k] + transCost.pre
273     simWealth.pre[rebPoint] = uStar*initWealth*return.risky + (1-uStar)*
274       initWealth*return.riskfree - transCost.pre
275
276     # Subsequent transaction costs
277     simWealth.sub[activeIndices] = uStar*initWealth*cumprod(1+drift*delta+
278       volatility*inc[activeIndices]) + (1-uStar)*initWealth*cumprod((1+rent*
279       delta)*ones)
280     transCost.sub = costProp * abs(uStar*(1-uStar)*initWealth*diffReturn)
281     totalTransCost.sub[k] = totalTransCost.sub[k] + transCost.sub
282     simWealth.sub[rebPoint] = uStar*initWealth*return.risky + (1-uStar)*
283       initWealth*return.riskfree - transCost.sub
284
285     #
286     # The rest of the simulated wealths
287     #
288     for (j in rebIndex[-length(rebIndex)] + 1) {
289
290       # Common quantities
291       activeIndices = j:(j+nRebDelay-1)
292       rebPoint = tail(activeIndices,1)

```

```

285         return.risky = prod(1+drift*delta+volatility*inc[activeIndices])
286         return.riskfree = prod((1+rent*delta)*ones)
287         diffReturn = return.risky - return.riskfree
288
289     # No transaction costs
290     simWealth[activeIndices] = uStar*simWealth[j-1]*cumprod(1+drift*delta+
        volatility*inc[activeIndices]) + (1-uStar)*simWealth[j-1]*cumprod
        ((1+rent*delta)*ones)
291
292     # Preceding transaction costs
293     simWealth.pre[activeIndices] = uStar*simWealth.pre[j-1]*cumprod(1+drift*
        delta+volatility*inc[activeIndices]) + (1-uStar)*simWealth.pre[j-1]*
        cumprod((1+rent*delta)*ones)
294     signDiffReturn = sign(diffReturn)
295     transCost.pre = costProp * abs((uStar*(1-uStar)*simWealth.pre[j-1]*
        diffReturn) / (1-signDiffReturn*costProp*uStar))
296     totalTransCost.pre[k] = totalTransCost.pre[k] + transCost.pre
297     simWealth.pre[rebPoint] = uStar*simWealth.pre[j-1]*return.risky + (1-
        uStar)*simWealth.pre[j-1]*return.riskfree - transCost.pre
298
299     # Subsequent transaction costs
300     simWealth.sub[activeIndices] = uStar*simWealth.sub[j-1]*cumprod(1+drift*
        delta+volatility*inc[activeIndices]) + (1-uStar)*simWealth.sub[j-1]*
        cumprod((1+rent*delta)*ones)
301     transCost.sub = costProp * abs(uStar*(1-uStar)*simWealth.sub[j-1]*
        diffReturn)
302     totalTransCost.sub[k] = totalTransCost.sub[k] + transCost.sub
303     simWealth.sub[rebPoint] = uStar*simWealth.sub[j-1]*return.risky + (1-
        uStar)*simWealth.sub[j-1]*return.riskfree - transCost.sub
304     }
305
306     thWealth.sd[k] = sd(thWealth)
307     thWealth.terminal[k] = thWealth[lastIndex]
308     thWealth.logReturn = logReturn(c(initWealth, thWealth))
309     thWealth.logReturn.sd[k] = sd(thWealth.logReturn)
310
311     simWealth.sd[k] = sd(simWealth)
312     simWealth.terminal[k] = simWealth[lastIndex]
313     simWealth.logReturn = logReturn(c(initWealth, simWealth))
314     simWealth.logReturn.sd[k] = sd(simWealth.logReturn)
315
316     simWealth.pre.sd[k] = sd(simWealth.pre)
317     simWealth.pre.terminal[k] = simWealth.pre[lastIndex]
318     simWealth.pre.logReturn = logReturn(c(initWealth, simWealth.pre))
319     simWealth.pre.logReturn.sd[k] = sd(simWealth.pre.logReturn)
320
321     simWealth.sub.sd[k] = sd(simWealth.sub)
322     simWealth.sub.terminal[k] = simWealth.sub[lastIndex]
323     simWealth.sub.logReturn = logReturn(c(initWealth, simWealth.sub))
324     simWealth.sub.logReturn.sd[k] = sd(simWealth.sub.logReturn)
325 }
326 }
327
328 # Calculation of total simulation time
329 timeElapsed = proc.time()[3][[1]] - timeStart
330 cat(nSims," simulation(s) completed in",timeElapsed," seconds.\n")
331 flush.console()
332
333 # Construction of the list of data to be returned from the function.
334 if (nSims == 1) {
335     stdNames = c("simWealth.risky", "simWealth.riskfree", "simWealth", "
        transQuantity", "transCost", "propInRisky", "propInRiskfree")
336     returnList.without = list(simWealth.risky, simWealth.riskfree, simWealth,
        transQuantity, propInRisky, propInRiskfree)
337     names(returnList.without) = stdNames[-5]

```

```

338     returnList.pre = list(simWealth.risky.pre, simWealth.riskfree.pre, simWealth.
339       pre, transQuantity.pre, transCost.pre, propInRisky.pre, propInRiskfree.pre)
340     names(returnList.pre) = stdNames
341     returnList.sub = list(simWealth.risky.sub, simWealth.riskfree.sub, simWealth.
342       sub, transQuantity.sub, transCost.sub, propInRisky.sub, propInRiskfree.sub)
343     names(returnList.sub) = stdNames
344     returnList = list(days, rebDays, rebIndex, inc, BM, riskyPrice, riskfreePrice,
345       thWealth, riskyReturn, riskfreeReturn, returnList.without, returnList.pre,
346       returnList.sub)
347     names(returnList) = c("days", "rebDays", "rebIndex", "BM.increments", "BM",
348       "riskyPrice", "riskfreePrice", "thWealth", "riskyReturn", "riskfreeReturn",
349       "withoutTransCost", "precedingTransCost", "subsequentTransCost")
350   }
351   else {
352     stdNames = c("simWealth.terminal", "simWealth.sd", "simWealth.logReturn.sd",
353       "totalTransCost")
354     returnList.th = list(thWealth.terminal, thWealth.sd, thWealth.logReturn.sd)
355     names(returnList.th) = c("thWealth.terminal", "thWealth.sd", "thWealth.
356       logReturn.sd")
357     returnList.without = list(simWealth.terminal, simWealth.sd, simWealth.
358       logReturn.sd)
359     names(returnList.without) = stdNames[-4]
360     returnList.pre = list(simWealth.pre.terminal, simWealth.pre.sd, simWealth.pre.
361       logReturn.sd, totalTransCost.pre)
362     names(returnList.pre) = stdNames
363     returnList.sub = list(simWealth.sub.terminal, simWealth.sub.sd, simWealth.sub.
364       logReturn.sd, totalTransCost.sub)
365     names(returnList.sub) = stdNames
366     returnList = list(returnList.th, returnList.without, returnList.pre, returnList
367       .sub)
368     names(returnList) = c("theoretical", "noTransCost", "precedingTransCost",
369       "subsequentTransCost")
370   }
371 }
372 return(returnList)
373 }

```

## B.5.2 Execution

```

1  ##
2  # Master thesis
3  # Simulation with transaction costs
4  # Two single runs
5  #
6
7  source("R/supportFunctions.R")
8  source("R/machinery_general.R")
9  source("R/initParameters.R")
10 source("R/machinery_transCost.R")
11
12 alpha = .05
13 qAlpha.half = qnorm(1-alpha/2)
14 graphics.off()
15
16 #
17 # Looking at difference in transaction cost
18 #
19
20 transCostDiffConstant.posDiff = function(lambda, uStar) { (lambda^2*uStar^2*(1-
    uStar)) / (1-lambda*uStar) }

```

```

21 | transCostDiffConstant.negDiff = function(lambda, uStar) { (lambda^2*uStar^2*(1-
    |   uStar)) / (1+lambda*uStar) }
22 |
23 | lambdaSeries = 0:350 / 10000
24 |
25 | xTitle = expression(paste(" Transaction cost proportion " * lambda))
26 | yTitle = expression(italic(f(lambda * "|" * u * "*" *))) %%% 10^-4
27 | constant = transCostDiffConstant.posDiff(lambdaSeries, uStar) * 1e4
28 | y.range = range(constant)
29 | niceplot(lambdaSeries, constant, xTitle=xTitle, yTitle=yTitle, figsPerPage=3,y.
    |   superscript=T, nCol=2)
30 | lambda = .01
31 | lines(c(lambda, lambda), c(-1, constant [lambda*10000]), lty=3)
32 | lines(c(-1, lambda), c(constant [lambda*10000], constant [lambda*10000]), lty=3)
33 | text(0, constant [lambda*10000], substitute(paste(number %%% 10^-4, list(number=
    |   round(constant [lambda*10000], 4), costProp=lambda)), adj=c(0, -.2), cex=.7)
34 | lambda = .02
35 | lines(c(lambda, lambda), c(-1, constant [lambda*10000]), lty=3)
36 | lines(c(-1, lambda), c(constant [lambda*10000], constant [lambda*10000]), lty=3)
37 | text(0, constant [lambda*10000], substitute(paste(number %%% 10^-4, list(number=
    |   round(constant [lambda*10000], 4), costProp=lambda)), adj=c(0, -.2), cex=.7)
38 | lambda = .03
39 | lines(c(lambda, lambda), c(-1, constant [lambda*10000]), lty=3)
40 | lines(c(-1, lambda), c(constant [lambda*10000], constant [lambda*10000]), lty=3)
41 | text(0, constant [lambda*10000], substitute(paste(number %%% 10^-4, list(number=
    |   round(constant [lambda*10000], 4), costProp=lambda)), adj=c(0, -.2), cex=.7)
42 | legendText = expression(paste(" (a)", ~D[k]>=0))
43 | legend(" topleft", legendText, bty="n", cex=.7)
44 |
45 | constant = transCostDiffConstant.negDiff(lambdaSeries, uStar) * 1e4
46 | niceplot(lambdaSeries, constant, xTitle=xTitle, yTitle=yTitle, y.superscript=T,
    |   multiPlot=T, newDev=F, ylim=y.range)
47 | lambda = .01
48 | lines(c(lambda, lambda), c(-1, constant [lambda*10000]), lty=3)
49 | lines(c(-1, lambda), c(constant [lambda*10000], constant [lambda*10000]), lty=3)
50 | text(0, constant [lambda*10000], substitute(paste(number %%% 10^-4, list(number=
    |   round(constant [lambda*10000], 4), costProp=lambda)), adj=c(0, -.2), cex=.7)
51 | lambda = .02
52 | lines(c(lambda, lambda), c(-1, constant [lambda*10000]), lty=3)
53 | lines(c(-1, lambda), c(constant [lambda*10000], constant [lambda*10000]), lty=3)
54 | text(0, constant [lambda*10000], substitute(paste(number %%% 10^-4, list(number=
    |   round(constant [lambda*10000], 4), costProp=lambda)), adj=c(0, -.2), cex=.7)
55 | lambda = .03
56 | lines(c(lambda, lambda), c(-1, constant [lambda*10000]), lty=3)
57 | lines(c(-1, lambda), c(constant [lambda*10000], constant [lambda*10000]), lty=3)
58 | text(0, constant [lambda*10000], substitute(paste(number %%% 10^-4, list(number=
    |   round(constant [lambda*10000], 4), costProp=lambda)), adj=c(0, -.2), cex=.7)
59 | legendText = expression(paste(" (b)", ~D[k]<0))
60 | legend(" topleft", legendText, bty="n", cex=.7)
61 |
62 | savePlot(" images/transCostConstant", type="eps")
63 |
64 | #
65 | # One simulation run: strong risky asset development
66 | #
67 |
68 | nSims = 1
69 |
70 | if (file.exists(" Datasett/singleRun_transCost_01.RData")) {
71 |   cat(" Loading simulated portfolios ... \n")
72 |   load(" Datasett/singleRun_transCost_01.RData")
73 | } else {
74 |   cat(" Simulating portfolios ... \n")
75 |   simObject.01 = simPortfolio.transCost(nSims, paramSet, costProp=.01, loadBrownian
    |     =T)

```

```

76 | save(simObject.01, file="Datsett/singleRun_transCost.01.RData")
77 | }
78 |
79 | if (file.exists("Datsett/singleRun_transCost.02.RData")) {
80 |   cat(" Loading simulated portfolios ... \n")
81 |   load("Datsett/singleRun_transCost.02.RData")
82 | } else {
83 |   cat(" Simulating portfolios ... \n")
84 |   simObject.02 = simPortfolio.transCost(nSims, paramSet, costProp=.02, loadBrownian
      =T)
85 |   save(simObject.02, file="Datsett/singleRun_transCost.02.RData")
86 | }
87 |
88 | if (file.exists("Datsett/singleRun_transCost.03.RData")) {
89 |   cat(" Loading simulated portfolios ... \n")
90 |   load("Datsett/singleRun_transCost.03.RData")
91 | } else {
92 |   cat(" Simulating portfolios ... \n")
93 |   simObject.03 = simPortfolio.transCost(nSims, paramSet, costProp=.03, loadBrownian
      =T)
94 |   save(simObject.03, file="Datsett/singleRun_transCost.03.RData")
95 | }
96 |
97 | days = c(0, simObject.01$days)
98 | rebDays = simObject.01$rebDays
99 |
100 | # Plotting risky and risk free asset prices as benchmark
101 | xTicks = c(0, rebDays)
102 | xTitle = "Trading days"
103 | yTitle = "Asset price"
104 | riskyPrice = c(1, simObject.01$riskyPrice)
105 | riskfreePrice = c(1, simObject.01$riskfreePrice)
106 | niceplot(days, riskyPrice, xTicks, xTitle=xTitle, yTitle=yTitle, figsPerPage=5, y.
      superscript=T, horizLines=T, col="red")
107 | nicelines(days, riskfreePrice, col="blue")
108 | abline(v=xTicks, lty=3)
109 | legendText = "(a)"
110 | legend("topleft", legendText, bty="n", cex=.7)
111 | savePlot("images/riskyPrice_risklessPrice", type="eps")
112 |
113 | # Plotting risky and risk free asset period returns
114 | rebIndex = simObject.01$rebIndex
115 | yTitle = "Asset return during period"
116 | y.max = max(c(0, simObject.01$riskyReturn[rebIndex]))
117 | niceplot(xTicks, c(0, simObject.01$riskyReturn[rebIndex]), xTicks, xTitle=xTitle,
      yTitle=yTitle, figsPerPage=5, y. superscript=T, horizLines=T, col="red")
118 | nicelines(xTicks, c(0, simObject.01$riskfreeReturn[rebIndex]), col="blue")
119 | abline(v=xTicks, lty=3)
120 | legendText = "(b)"
121 | legend("topleft", legendText, bty="n", cex=.7)
122 | savePlot("images/riskyReturn_risklessReturn", type="eps")
123 |
124 | # Transaction cost differences, lambda = .01
125 | costProp = .03
126 | transCost.03.pre = abs(costProp * simObject.03$precedingTransCost$transQuantity[
      rebIndex])
127 | transCost.03.sub = abs(costProp * simObject.03$subsequentTransCost$transQuantity
      [rebIndex])
128 | transCost.03.diff = transCost.03.pre - transCost.03.sub
129 | y.range = range(transCost.03.diff*1e5)
130 | yTitle = expression(paste("Trans. cost difference", phantom(0) %*% 10^5))
131 | niceplot(xTicks, c(0, transCost.03.diff*1e5), xTicks, xTitle=xTitle, yTitle=yTitle,
      figsPerPage=5, y. superscript=T)
132 | abline(v=xTicks, lty=3)
133 | abline(h=0, lty=3)

```

```

134 legendText = expression(paste("(e) ", lambda*"=.03"))
135 legend("topleft", legendText, bty="n", cex=.7)
136 savePlot("images/pre_sub_diff_03", type="eps")
137
138 # Transaction cost differences, lambda = .02
139 costProp = .01
140 transCost.01.pre = abs(costProp * simObject.01$precedingTransCost$transQuantity[
    rebIndex])
141 transCost.01.sub = abs(costProp * simObject.01$subsequentTransCost$transQuantity
    [rebIndex])
142 transCost.01.diff = transCost.01.pre - transCost.01.sub
143 niceplot(xTicks, c(0, transCost.01.diff*1e5), xTicks, xTitle=xTitle, yTitle=yTitle,
    figsPerPage=5, y.superscript=Γ, ylim=y.range)
144 abline(v=xTicks, lty=3)
145 abline(h=0, lty=3)
146 legendText = expression(paste("(c) ", lambda*"=.01"))
147 legend("topleft", legendText, bty="n", cex=.7)
148 savePlot("images/pre_sub_diff_01", type="eps")
149
150 # Transaction cost differences, lambda = .03
151 costProp = .02
152 transCost.02.pre = abs(costProp * simObject.02$precedingTransCost$transQuantity[
    rebIndex]) * 1e5
153 transCost.02.sub = abs(costProp * simObject.02$subsequentTransCost$transQuantity
    [rebIndex]) * 1e5
154 niceplot(xTicks, c(0, transCost.02.pre-transCost.02.sub), xTicks, xTitle=xTitle,
    yTitle=yTitle, figsPerPage=5, y.superscript=Γ, ylim=y.range)
155 abline(v=xTicks, lty=3)
156 abline(h=0, lty=3)
157 legendText = expression(paste("(d) ", lambda*"=.02"))
158 legend("topleft", legendText, bty="n", cex=.7)
159 savePlot("images/pre_sub_diff_02", type="eps")
160
161 # Transaction cost different ratio
162 transCost.03.01.diff.ratio = transCost.03.diff / transCost.01.diff
163 print(transCost.03.01.diff.ratio)
164 yTitle = expression(paste("Trans. cost ratio ", lambda=="03," vs ", lambda=="01"))
165 niceplot(xTicks[-1], transCost.03.01.diff.ratio, xTicks, xTitle=xTitle, yTitle=
    yTitle)
166 abline(v=xTicks[-1], lty=3)
167 savePlot("images/transCost_diff_ratio", type="eps")
168
169 # Creating summarizing table
170 withoutTransCost = simObject.01$withoutTransCost
171 precedingTransCost.01 = simObject.01$precedingTransCost
172 subsequentTransCost.01 = simObject.01$subsequentTransCost
173
174 precedingTransCost.02 = simObject.02$precedingTransCost
175 subsequentTransCost.02 = simObject.02$subsequentTransCost
176
177 precedingTransCost.03 = simObject.03$precedingTransCost
178 subsequentTransCost.03 = simObject.03$subsequentTransCost
179
180 terminalWealth.th = last(simObject.01$thWealth)
181 terminalUtility.th = utility(terminalWealth.th, riskAversion)
182
183 terminalWealth.without = last(withoutTransCost$simWealth)
184 lossOfWealth.without = terminalWealth.th - terminalWealth.without
185 terminalUtility.without = utility(terminalWealth.without, riskAversion)
186 lossOfUtility.without = terminalUtility.th - terminalUtility.without
187
188 costProp = .01
189 terminalWealth.pre.01 = last(precedingTransCost.01$simWealth)
190 lossOfWealth.pre.01 = terminalWealth.th - terminalWealth.pre.01
191 terminalUtility.pre.01 = utility(terminalWealth.pre.01, riskAversion)

```

```

192 | lossOfUtility.pre.01 = terminalUtility.th - terminalUtility.pre.01
193 | totalTransCost.pre.01 = sum(precedingTransCost.01$transCost)
194 | terminalWealth.sub.01 = last(subsequentTransCost.01$simWealth)
195 | lossOfWealth.sub.01 = terminalWealth.th - terminalWealth.sub.01
196 | terminalUtility.sub.01 = utility(terminalWealth.sub.01, riskAversion)
197 | lossOfUtility.sub.01 = terminalUtility.th - terminalUtility.sub.01
198 | totalTransCost.sub.01 = sum(subsequentTransCost.01$transCost)
199 |
200 | costProp = .02
201 | terminalWealth.pre.02 = last(precedingTransCost.02$simWealth)
202 | lossOfWealth.pre.02 = terminalWealth.th - terminalWealth.pre.02
203 | terminalUtility.pre.02 = utility(terminalWealth.pre.02, riskAversion)
204 | lossOfUtility.pre.02 = terminalUtility.th - terminalUtility.pre.02
205 | totalTransCost.pre.02 = sum(precedingTransCost.02$transCost)
206 | terminalWealth.sub.02 = last(subsequentTransCost.02$simWealth)
207 | lossOfWealth.sub.02 = terminalWealth.th - terminalWealth.sub.02
208 | terminalUtility.sub.02 = utility(terminalWealth.sub.02, riskAversion)
209 | lossOfUtility.sub.02 = terminalUtility.th - terminalUtility.sub.02
210 | totalTransCost.sub.02 = sum(subsequentTransCost.02$transCost)
211 |
212 | costProp = .03
213 | terminalWealth.pre.03 = last(precedingTransCost.03$simWealth)
214 | lossOfWealth.pre.03 = terminalWealth.th - terminalWealth.pre.03
215 | terminalUtility.pre.03 = utility(terminalWealth.pre.03, riskAversion)
216 | lossOfUtility.pre.03 = terminalUtility.th - terminalUtility.pre.03
217 | totalTransCost.pre.03 = sum(precedingTransCost.03$transCost)
218 | terminalWealth.sub.03 = last(subsequentTransCost.03$simWealth)
219 | lossOfWealth.sub.03 = terminalWealth.th - terminalWealth.sub.03
220 | terminalUtility.sub.03 = utility(terminalWealth.sub.03, riskAversion)
221 | lossOfUtility.sub.03 = terminalUtility.th - terminalUtility.sub.03
222 | totalTransCost.sub.03 = sum(subsequentTransCost.03$transCost)
223 |
224 | tab = matrix(NA, 8, 5)
225 | tab[1,] = c(terminalWealth.th, 0, terminalUtility.th, 0, 0)
226 | tab[2,] = c(terminalWealth.without, lossOfWealth.without, terminalUtility.without,
227 |           lossOfUtility.without, 0)
228 | tab[3,] = c(terminalWealth.pre.01, lossOfWealth.pre.01, terminalUtility.pre.01,
229 |           lossOfUtility.pre.01, totalTransCost.pre.01)
230 | tab[4,] = c(terminalWealth.pre.02, lossOfWealth.pre.02, terminalUtility.pre.02,
231 |           lossOfUtility.pre.02, totalTransCost.pre.02)
232 | tab[5,] = c(terminalWealth.pre.03, lossOfWealth.pre.03, terminalUtility.pre.03,
233 |           lossOfUtility.pre.03, totalTransCost.pre.03)
234 | tab[6,] = c(terminalWealth.sub.01, lossOfWealth.sub.01, terminalUtility.sub.01,
235 |           lossOfUtility.sub.01, totalTransCost.sub.01)
236 | tab[7,] = c(terminalWealth.sub.02, lossOfWealth.sub.02, terminalUtility.sub.02,
237 |           lossOfUtility.sub.02, totalTransCost.sub.02)
238 | tab[8,] = c(terminalWealth.sub.03, lossOfWealth.sub.03, terminalUtility.sub.03,
239 |           lossOfUtility.sub.03, totalTransCost.sub.03)
240 |
241 | tab.orig = tab
242 | print(tab)
243 | tab[,2] = tab[,2] * 1e3
244 | tab[,4:5] = tab[,4:5] * 1e3
245 | tab = round(tab, 4)
246 | as.data.frame(tab)
247 |
248 | for (k in 1:8) {
249 |   tab[k,2] = paste(tab[k,2], "\\e{\\text{-3}}", sep="")
250 |   tab[k,4] = paste(tab[k,4], "\\e{\\text{-3}}", sep="")
251 |   tab[k,5] = paste(tab[k,5], "\\e{\\text{-3}}", sep="")
252 | }
253 | printex(tab)
254 |
255 | #
256 | # Second simulation run: weak risky asset development

```



```

250 #
251
252 if (file.exists("Datasett/singleRun2_transCost_01.RData")) {
253   cat("Loading simulated portfolios...\n")
254   load("Datasett/singleRun2_transCost_01.RData")
255 } else {
256   cat("Simulating portfolios...\n")
257   simObject.01 = simPortfolio.transCost(nSims, paramSet, costProp=.01,
258     brownianFileName="Datasett/brownianIncrements2.RData")
259   while (last(simObject.01$riskyPrice) > .75) {
260     file.remove("Datasett/brownianIncrements2.RData")
261     simObject.01 = simPortfolio.transCost(nSims, paramSet, costProp=.01,
262       brownianFileName="Datasett/brownianIncrements2.RData")
263   }
264   save(simObject.01, file="Datasett/singleRun2_transCost_01.RData")
265 }
266
267 if (file.exists("Datasett/singleRun2_transCost_02.RData")) {
268   cat("Loading simulated portfolios...\n")
269   load("Datasett/singleRun2_transCost_02.RData")
270 } else {
271   cat("Simulating portfolios...\n")
272   simObject.02 = simPortfolio.transCost(nSims, paramSet, costProp=.02,
273     brownianFileName="Datasett/brownianIncrements2.RData")
274   save(simObject.02, file="Datasett/singleRun2_transCost_02.RData")
275 }
276
277 if (file.exists("Datasett/singleRun2_transCost_03.RData")) {
278   cat("Loading simulated portfolios...\n")
279   load("Datasett/singleRun2_transCost_03.RData")
280 } else {
281   cat("Simulating portfolios...\n")
282   simObject.03 = simPortfolio.transCost(nSims, paramSet, costProp=.03,
283     brownianFileName="Datasett/brownianIncrements2.RData")
284   save(simObject.03, file="Datasett/singleRun2_transCost_03.RData")
285 }
286
287 rebIndex = simObject.01$rebIndex
288 days = c(0, simObject.01$days)
289 rebDays = simObject.01$rebDays
290 xTicks = c(0, rebDays)
291 xTitle = "Trading days"
292
293 # Plotting risky and risk free asset prices as benchmark
294 yTitle = "Asset price"
295 riskyPrice = c(1, simObject.01$riskyPrice)
296 riskfreePrice = c(1, simObject.01$riskfreePrice)
297 niceplot(days, riskyPrice, xTicks, xTitle=xTitle, yTitle=yTitle, figsPerPage=5, y.
298   superscript=T, horizLines=T, col="red")
299 nicelines(days, riskfreePrice, col="blue")
300 abline(v=xTicks, lty=3)
301 legendText = "(a)"
302 legend("topleft", legendText, bty="n", cex=.7)
303 savePlot("images/riskyPrice_risklessPrice2", type="eps")
304
305 # Plotting risky and risk free asset period returns
306 yTitle = "Asset return during period"
307 niceplot(xTicks, c(0, simObject.01$riskyReturn[rebIndex]), xTicks, xTitle=xTitle,
308   yTitle=yTitle, figsPerPage=5, y. superscript=T, horizLines=T, col="red")
309 nicelines(xTicks, c(0, simObject.01$riskfreeReturn[rebIndex]), col="blue")
310 abline(v=xTicks, lty=3)
311 legendText = "(b)"
312 legend("topleft", legendText, bty="n", cex=.7)
313 savePlot("images/riskyReturn_risklessReturn2", type="eps")
314
315

```

```

309 # Creating summarizing table
310 withoutTransCost = simObject.01$withoutTransCost
311 precedingTransCost.01 = simObject.01$precedingTransCost
312 subsequentTransCost.01 = simObject.01$subsequentTransCost
313
314 precedingTransCost.02 = simObject.02$precedingTransCost
315 subsequentTransCost.02 = simObject.02$subsequentTransCost
316
317 precedingTransCost.03 = simObject.03$precedingTransCost
318 subsequentTransCost.03 = simObject.03$subsequentTransCost
319
320 terminalWealth.th = last(simObject.01$thWealth)
321 terminalUtility.th = utility(terminalWealth.th, riskAversion)
322
323 terminalWealth.without = last(withoutTransCost$simWealth)
324 lossOfWealth.without = terminalWealth.th - terminalWealth.without
325 terminalUtility.without = utility(terminalWealth.without, riskAversion)
326 lossOfUtility.without = terminalUtility.th - terminalUtility.without
327
328 costProp = .01
329 terminalWealth.pre.01 = last(precedingTransCost.01$simWealth)
330 lossOfWealth.pre.01 = terminalWealth.th - terminalWealth.pre.01
331 terminalUtility.pre.01 = utility(terminalWealth.pre.01, riskAversion)
332 lossOfUtility.pre.01 = terminalUtility.th - terminalUtility.pre.01
333 totalTransCost.pre.01 = sum(precedingTransCost.01$transCost)
334 terminalWealth.sub.01 = last(subsequentTransCost.01$simWealth)
335 lossOfWealth.sub.01 = terminalWealth.th - terminalWealth.sub.01
336 terminalUtility.sub.01 = utility(terminalWealth.sub.01, riskAversion)
337 lossOfUtility.sub.01 = terminalUtility.th - terminalUtility.sub.01
338 totalTransCost.sub.01 = sum(subsequentTransCost.01$transCost)
339
340 costProp = .02
341 terminalWealth.pre.02 = last(precedingTransCost.02$simWealth)
342 lossOfWealth.pre.02 = terminalWealth.th - terminalWealth.pre.02
343 terminalUtility.pre.02 = utility(terminalWealth.pre.02, riskAversion)
344 lossOfUtility.pre.02 = terminalUtility.th - terminalUtility.pre.02
345 totalTransCost.pre.02 = sum(precedingTransCost.02$transCost)
346 terminalWealth.sub.02 = last(subsequentTransCost.02$simWealth)
347 lossOfWealth.sub.02 = terminalWealth.th - terminalWealth.sub.02
348 terminalUtility.sub.02 = utility(terminalWealth.sub.02, riskAversion)
349 lossOfUtility.sub.02 = terminalUtility.th - terminalUtility.sub.02
350 totalTransCost.sub.02 = sum(subsequentTransCost.02$transCost)
351
352 costProp = .03
353 terminalWealth.pre.03 = last(precedingTransCost.03$simWealth)
354 lossOfWealth.pre.03 = terminalWealth.th - terminalWealth.pre.03
355 terminalUtility.pre.03 = utility(terminalWealth.pre.03, riskAversion)
356 lossOfUtility.pre.03 = terminalUtility.th - terminalUtility.pre.03
357 totalTransCost.pre.03 = sum(precedingTransCost.03$transCost)
358 terminalWealth.sub.03 = last(subsequentTransCost.03$simWealth)
359 lossOfWealth.sub.03 = terminalWealth.th - terminalWealth.sub.03
360 terminalUtility.sub.03 = utility(terminalWealth.sub.03, riskAversion)
361 lossOfUtility.sub.03 = terminalUtility.th - terminalUtility.sub.03
362 totalTransCost.sub.03 = sum(subsequentTransCost.03$transCost)
363
364 tab = matrix(NA, 8, 5)
365 tab[1,] = c(terminalWealth.th, 0, terminalUtility.th, 0, 0)
366 tab[2,] = c(terminalWealth.without, lossOfWealth.without, terminalUtility.without,
367           lossOfUtility.without, 0)
368 tab[3,] = c(terminalWealth.pre.01, lossOfWealth.pre.01, terminalUtility.pre.01,
369           lossOfUtility.pre.01, totalTransCost.pre.01)
370 tab[4,] = c(terminalWealth.pre.02, lossOfWealth.pre.02, terminalUtility.pre.02,
371           lossOfUtility.pre.02, totalTransCost.pre.02)
372 tab[5,] = c(terminalWealth.pre.03, lossOfWealth.pre.03, terminalUtility.pre.03,
373           lossOfUtility.pre.03, totalTransCost.pre.03)

```

```

370 tab[6,] = c(terminalWealth.sub.01,lossOfWealth.sub.01,terminalUtility.sub.01,
             lossOfUtility.sub.01,totalTransCost.sub.01)
371 tab[7,] = c(terminalWealth.sub.02,lossOfWealth.sub.02,terminalUtility.sub.02,
             lossOfUtility.sub.02,totalTransCost.sub.02)
372 tab[8,] = c(terminalWealth.sub.03,lossOfWealth.sub.03,terminalUtility.sub.03,
             lossOfUtility.sub.03,totalTransCost.sub.03)
373
374 tab.orig = tab
375 print(tab)
376 tab[,2] = tab[,2] * 1e3
377 tab[,4:5] = tab[,4:5] * 1e3
378 tab = round(tab,4)
379 as.data.frame(tab)
380
381 for (k in 1:8) {
382   tab[k,2] = paste(tab[k,2],"\e{\text-3}",sep="")
383   tab[k,4] = paste(tab[k,4],"\e{\text-3}",sep="")
384   tab[k,5] = paste(tab[k,5],"\e{\text-3}",sep="")
385 }
386 printex(tab)

```

```

1  ##
2  # Master thesis
3  # Simulations with transaction costs
4  # Loss of utility and Sharpe ratio
5  #
6
7  #
8  # Initialization
9  #
10
11 require(doSMP)
12 source("R/supportFunctions.R")
13 source("R/machinery_general.R")
14 source("R/initParameters.R")
15 source("R/machinery_transCost.R")
16
17 alpha = .05
18 qAlpha.half = qnorm(1-alpha/2)
19 graphics.off()
20
21 ##
22 # Rebalancing strategy vs loss of utility
23 #
24
25 # Common parameter settings
26 nSims = 100000
27 nCores = 25
28 nDailyRebs = c(24,6,1,1/2,1/12,1/21,1/42,1/126,1/252)
29 strategyNames = c("Hourly","Every 4th hour","Daily","Every 3rd day","Every 12th
                 day","Monthly","Bimonthly","Semiannually","Annually")
30
31 #
32 # Performing simulations, transaction cost proportion = .01
33 #
34
35 costProp = .01
36 paramSets.transCost = cbind(initWealth,nTradingDays,nDailyIncrements,nDailyRebs,
                             drift,volatility,rent,riskAversion,uStar,costProp)
37 rebStrategy.transCost.01 = distribute(nSims,nCores,simPortfolio.transCost,
                                     paramSets.transCost)
38 names(rebStrategy.transCost.01) = strategyNames
39
40 # Organizing returned data

```

```

41 n.entries = length(rebStrategy.transCost.01)
42 for (k in 1:n.entries) {
43
44   th = rebStrategy.transCost.01[[k]]$theoretical
45   rebStrategy.transCost.01[[k]]$theoretical = list(merge.list(th[seq(1,3*nCores
46     -2,3)]), merge.list(th[seq(2,3*nCores-1,3)]), merge.list(th[seq(3,3*nCores
47     ,3)]))
48   names(rebStrategy.transCost.01[[k]]$theoretical) = c("thWealth.terminal",
49     "thWealth.sd", "thWealth.logReturn.sd")
50
51   no = rebStrategy.transCost.01[[k]]$noTransCost
52   rebStrategy.transCost.01[[k]]$noTransCost = list(merge.list(no[seq(1,3*nCores
53     -2,3)]), merge.list(no[seq(2,3*nCores-1,3)]), merge.list(no[seq(3,3*nCores
54     ,3)]))
55   names(rebStrategy.transCost.01[[k]]$noTransCost) = c("simWealth.terminal",
56     "simWealth.sd", "simWealth.logReturn.sd")
57
58   pre = rebStrategy.transCost.01[[k]]$precedingTransCost
59   rebStrategy.transCost.01[[k]]$precedingTransCost = list(merge.list(pre[seq
60     (1,4*nCores-3,4)]), merge.list(pre[seq(2,4*nCores-2,4)]), merge.list(pre[
61     seq(3,4*nCores-1,4)]), merge.list(pre[seq(4,4*nCores,4)]))
62   names(rebStrategy.transCost.01[[k]]$precedingTransCost) = c("simWealth.
63     terminal", "simWealth.sd", "simWealth.logReturn.sd", "totalTransCost")
64
65   sub = rebStrategy.transCost.01[[k]]$subsequentTransCost
66   rebStrategy.transCost.01[[k]]$subsequentTransCost = list(merge.list(sub[seq
67     (1,4*nCores-3,4)]), merge.list(sub[seq(2,4*nCores-2,4)]), merge.list(sub[
68     seq(3,4*nCores-1,4)]), merge.list(sub[seq(4,4*nCores,4)]))
69   names(rebStrategy.transCost.01[[k]]$subsequentTransCost) = c("simWealth.
70     terminal", "simWealth.sd", "simWealth.logReturn.sd", "totalTransCost")
71 }
72
73 save(rebStrategy.transCost.01, file="Datasett/rebStrategy_transCost_01.RData")
74
75 #
76 # Performing simulations, transaction cost proportion = .02
77 #
78
79 costProp = .02
80 paramSets.transCost = cbind(initWealth, nTradingDays, nDailyIncrements, nDailyRebs,
81   drift, volatility, rent, riskAversion, uStar, costProp)
82 rebStrategy.transCost.02 = distribute(nSims, nCores, simPortfolio.transCost,
83   paramSets.transCost)
84 names(rebStrategy.transCost.02) = strategyNames
85
86 n.entries = length(rebStrategy.transCost.02)
87 for (k in 1:n.entries) {
88
89   th = rebStrategy.transCost.02[[k]]$theoretical
90   rebStrategy.transCost.02[[k]]$theoretical = list(merge.list(th[seq(1,3*nCores
91     -2,3)]), merge.list(th[seq(2,3*nCores-1,3)]), merge.list(th[seq(3,3*nCores
92     ,3)]))
93   names(rebStrategy.transCost.02[[k]]$theoretical) = c("thWealth.terminal",
94     "thWealth.sd", "thWealth.logReturn.sd")
95
96   no = rebStrategy.transCost.02[[k]]$noTransCost
97   rebStrategy.transCost.02[[k]]$noTransCost = list(merge.list(no[seq(1,3*nCores
98     -2,3)]), merge.list(no[seq(2,3*nCores-1,3)]), merge.list(no[seq(3,3*nCores
99     ,3)]))
100   names(rebStrategy.transCost.02[[k]]$noTransCost) = c("simWealth.terminal",
101     "simWealth.sd", "simWealth.logReturn.sd")
102
103   pre = rebStrategy.transCost.02[[k]]$precedingTransCost
104   rebStrategy.transCost.02[[k]]$precedingTransCost = list(merge.list(pre[seq
105     (1,4*nCores-3,4)]), merge.list(pre[seq(2,4*nCores-2,4)]), merge.list(pre[

```

```

85     seq(3,4*nCores-1,4)), merge.list(pre[seq(4,4*nCores,4)])
names(rebStrategy.transCost.02[[k]]$precedingTransCost) = c("simWealth.
    terminal","simWealth.sd","simWealth.logReturn.sd","totalTransCost")
86
87 sub = rebStrategy.transCost.02[[k]]$subsequentTransCost
88 rebStrategy.transCost.02[[k]]$subsequentTransCost = list(merge.list(sub[seq
    (1,4*nCores-3,4)], merge.list(sub[seq(2,4*nCores-2,4)]), merge.list(sub[
    seq(3,4*nCores-1,4)]), merge.list(sub[seq(4,4*nCores,4)]))
89 names(rebStrategy.transCost.02[[k]]$subsequentTransCost) = c("simWealth.
    terminal","simWealth.sd","simWealth.logReturn.sd","totalTransCost")
90 }
91
92 save(rebStrategy.transCost.02, file="Datasett/rebStrategy_transCost_02.RData")
93
94 #
95 # Performing simulations, transaction cost proportion = .03
96 #
97
98 costProp = .03
99 paramSets.transCost = cbind(initWealth, nTradingDays, nDailyIncrements, nDailyRebs,
    drift, volatility, rent, riskAversion, uStar, costProp)
100 rebStrategy.transCost.03 = distribute(nSims, nCores, simPortfolio.transCost,
    paramSets.transCost)
101 names(rebStrategy.transCost.03) = strategyNames
102 load("Datasett/rebStrategy_transCost_03.RData")
103
104 n.entries = length(rebStrategy.transCost.03)
105 for (k in 1:n.entries) {
106
107 th = rebStrategy.transCost.03[[k]]$theoretical
108 rebStrategy.transCost.03[[k]]$theoretical = list(merge.list(th[seq(1,3*nCores
    -2,3)], merge.list(th[seq(2,3*nCores-1,3)]), merge.list(th[seq(3,3*nCores
    ,3)]))
109 names(rebStrategy.transCost.03[[k]]$theoretical) = c("thWealth.terminal","
    thWealth.sd","thWealth.logReturn.sd")
110
111 no = rebStrategy.transCost.03[[k]]$noTransCost
112 rebStrategy.transCost.03[[k]]$noTransCost = list(merge.list(no[seq(1,3*nCores
    -2,3)], merge.list(no[seq(2,3*nCores-1,3)]), merge.list(no[seq(3,3*nCores
    ,3)]))
113 names(rebStrategy.transCost.03[[k]]$noTransCost) = c("simWealth.terminal","
    simWealth.sd","simWealth.logReturn.sd")
114
115 pre = rebStrategy.transCost.03[[k]]$precedingTransCost
116 rebStrategy.transCost.03[[k]]$precedingTransCost = list(merge.list(pre[seq
    (1,4*nCores-3,4)], merge.list(pre[seq(2,4*nCores-2,4)]), merge.list(pre[
    seq(3,4*nCores-1,4)]), merge.list(pre[seq(4,4*nCores,4)]))
117 names(rebStrategy.transCost.03[[k]]$precedingTransCost) = c("simWealth.
    terminal","simWealth.sd","simWealth.logReturn.sd","totalTransCost")
118
119 sub = rebStrategy.transCost.03[[k]]$subsequentTransCost
120 rebStrategy.transCost.03[[k]]$subsequentTransCost = list(merge.list(sub[seq
    (1,4*nCores-3,4)], merge.list(sub[seq(2,4*nCores-2,4)]), merge.list(sub[
    seq(3,4*nCores-1,4)]), merge.list(sub[seq(4,4*nCores,4)]))
121 names(rebStrategy.transCost.03[[k]]$subsequentTransCost) = c("simWealth.
    terminal","simWealth.sd","simWealth.logReturn.sd","totalTransCost")
122 }
123
124 save(rebStrategy.transCost.03, file="Datasett/rebStrategy_transCost_03.RData")
125
126 #
127 # Calculating relevant statistics and plotting
128 # Transaction cost proportion = .01
129 # Preceding transaction costs
130 #

```

```

131
132 x.labels = c(0,"10k","20k","30k","40k","50k","60k","70k","80k","90k","100k")
133 nn = 1:nSims
134 sel4 = c(1,3,6,9)
135
136 # Theoretical
137 terminalWealth.th.01 = matrix(NA,nSims,n.entries)
138 sdWealth.th.01 = matrix(NA,nSims,n.entries)
139 sdLogReturn.th.01 = matrix(NA,nSims,n.entries)
140 for (k in 1:n.entries) {
141   terminalWealth.th.01[,k] = rebStrategy.transCost.01[[c(k,1)]]$thWealth.
      terminal
142   sdWealth.th.01[,k] = rebStrategy.transCost.01[[c(k,1)]]$thWealth.sd
143   sdLogReturn.th.01[,k] = rebStrategy.transCost.01[[c(k,1)]]$thWealth.logReturn.
      sd
144 }
145 colnames(terminalWealth.th.01) = strategyNames
146 terminalWealth.th.01.mean = colMeans(terminalWealth.th.01)
147 terminalWealth.th.01.mean.sel4 = terminalWealth.th.01.mean[sel4]
148 sdWealth.th.01.mean = colMeans(sdWealth.th.01)
149 sdWealth.th.01.mean.sel4 = sdWealth.th.01.mean[sel4]
150 terminalWealth.th.01.sd = colSds(terminalWealth.th.01)
151 terminalWealth.th.01.sd.sel4 = terminalWealth.th.01.sd[sel4]
152 terminalUtility.th.01 = utility(terminalWealth.th.01,riskAversion)
153 terminalUtility.th.01.mean = colMeans(terminalUtility.th.01)
154 terminalUtility.th.01.mean.sel4 = terminalUtility.th.01.mean[sel4]
155 terminalLogReturn.th.01 = log(terminalWealth.th.01)
156 terminalLogReturn.th.01.mean = colMeans(terminalLogReturn.th.01)
157 terminalLogReturn.th.01.mean.sel4 = terminalLogReturn.th.01.mean[sel4]
158 sdLogReturn.th.01.mean = colMeans(sdLogReturn.th.01)
159 sdLogReturn.th.01.mean.sel4 = sdLogReturn.th.01.mean[sel4]
160 annualizedSdLogReturn.th.01 = sdLogReturn.th.01 * sqrt(nTimePoints)
161 annualizedSdLogReturn.th.01.mean = colMeans(annualizedSdLogReturn.th.01)
162 terminalLogReturn.th.01.sd = colSds(terminalLogReturn.th.01)
163 terminalLogReturn.th.01.sd.sel4 = terminalLogReturn.th.01.sd[sel4]
164 excessReturn.th.01 = terminalLogReturn.th.01 - rent
165 sharpeRatio.th.01 = excessReturn.th.01 / (sqrt(nTimePoints)*sdLogReturn.th.01)
166 sharpeRatio.th.01.mean = colMeans(sharpeRatio.th.01)
167 sharpeRatio.th.01.mean.sel4 = sharpeRatio.th.01.mean[sel4]
168 volOfVol.th.01 = colSds(annualizedSdLogReturn.th.01)
169 correlation.th.01 = colCorrs(terminalLogReturn.th.01,annualizedSdLogReturn.th
      .01)
170
171 # Simulated, no transaction costs
172 terminalWealth.none.01 = matrix(NA,nSims,n.entries)
173 sdWealth.none.01 = matrix(NA,nSims,n.entries)
174 sdLogReturn.none.01 = matrix(NA,nSims,n.entries)
175 for (k in 1:n.entries) {
176   terminalWealth.none.01[,k] = rebStrategy.transCost.01[[c(k,2)]]$simWealth.
      terminal
177   sdWealth.none.01[,k] = rebStrategy.transCost.01[[c(k,2)]]$simWealth.sd
178   sdLogReturn.none.01[,k] = rebStrategy.transCost.01[[c(k,2)]]$simWealth.
      logReturn.sd
179 }
180 colnames(terminalWealth.none.01) = strategyNames
181 terminalWealth.none.01.mean = colMeans(terminalWealth.none.01)
182 terminalWealth.none.01.mean.sel4 = terminalWealth.none.01.mean[sel4]
183 sdWealth.none.01.mean = colMeans(sdWealth.none.01)
184 sdWealth.none.01.mean.sel4 = sdWealth.none.01.mean[sel4]
185 terminalWealth.none.01.sd = colSds(terminalWealth.none.01)
186 terminalWealth.none.01.sd.sel4 = terminalWealth.none.01.sd[sel4]
187 lossOfWealth.none.01 = terminalWealth.th.01 - terminalWealth.none.01
188 lossOfWealth.none.01.mean = colMeans(lossOfWealth.none.01)
189 lossOfWealth.none.01.mean.sel4 = lossOfWealth.none.01.mean[sel4]
190 terminalUtility.none.01 = utility(terminalWealth.none.01,riskAversion)

```

```

191 terminalUtility.none.01.mean = colMeans(terminalUtility.none.01)
192 terminalUtility.none.01.mean.sel4 = terminalUtility.none.01.mean[sel4]
193 lossOfUtility.none.01 = terminalUtility.th.01 - terminalUtility.none.01
194 lossOfUtility.none.01.mean = colMeans(lossOfUtility.none.01)
195 lossOfUtility.none.01.mean.sel4 = lossOfUtility.none.01.mean[sel4]
196 lossOfUtility.none.01.sd = colSds(lossOfUtility.none.01)
197 terminalLogReturn.none.01 = log(terminalWealth.none.01)
198 terminalLogReturn.none.01.mean = colMeans(terminalLogReturn.none.01)
199 terminalLogReturn.none.01.mean.sel4 = terminalLogReturn.none.01.mean[sel4]
200 sdLogReturn.none.01.mean = colMeans(sdLogReturn.none.01)
201 sdLogReturn.none.01.mean.sel4 = sdLogReturn.none.01.mean[sel4]
202 annualizedSdLogReturn.none.01 = sdLogReturn.none.01 * sqrt(nTimePoints)
203 annualizedSdLogReturn.none.01.mean = colMeans(annualizedSdLogReturn.none.01)
204 terminalLogReturn.none.01.sd = colSds(terminalLogReturn.none.01)
205 terminalLogReturn.none.01.sd.sel4 = terminalLogReturn.none.01.sd[sel4]
206 excessReturn.none.01 = terminalLogReturn.none.01 - rent
207 sharpeRatio.none.01 = excessReturn.none.01 / (sqrt(nTimePoints)*sdLogReturn.none
    .01)
208 sharpeRatio.none.01.mean = colMeans(sharpeRatio.none.01)
209 sharpeRatio.none.01.mean.sel4 = sharpeRatio.none.01.mean[sel4]
210 volOfVol.none.01 = colSds(annualizedSdLogReturn.none.01)
211 correlation.none.01 = colCorrs(terminalLogReturn.none.01, annualizedSdLogReturn.
    none.01)
212
213 # Simulated, preceding transaction costs
214 terminalWealth.pre.01 = matrix(NA, nSims, n.entries)
215 sdWealth.pre.01 = matrix(NA, nSims, n.entries)
216 sdLogReturn.pre.01 = matrix(NA, nSims, n.entries)
217 totalTransCost.pre.01 = matrix(NA, nSims, n.entries)
218 for (k in 1:n.entries) {
219   terminalWealth.pre.01[,k] = rebStrategy.transCost.01[[c(k,3)]]$simWealth.
    terminal
220   sdWealth.pre.01[,k] = rebStrategy.transCost.01[[c(k,3)]]$simWealth.sd
221   sdLogReturn.pre.01[,k] = rebStrategy.transCost.01[[c(k,3)]]$simWealth.
    logReturn.sd
222   totalTransCost.pre.01[,k] = rebStrategy.transCost.01[[c(k,3)]]$totalTransCost
223 }
224 colnames(terminalWealth.pre.01) = strategyNames
225 terminalWealth.pre.01.mean = colMeans(terminalWealth.pre.01)
226 terminalWealth.pre.01.mean.sel4 = terminalWealth.pre.01.mean[sel4]
227 sdWealth.pre.01.mean = colMeans(sdWealth.pre.01)
228 sdWealth.pre.01.mean.sel4 = sdWealth.pre.01.mean[sel4]
229 terminalWealth.pre.01.sd = colSds(terminalWealth.pre.01)
230 terminalWealth.pre.01.sd.sel4 = terminalWealth.pre.01.sd[sel4]
231 lossOfWealth.pre.01 = terminalWealth.th.01 - terminalWealth.pre.01
232 lossOfWealth.pre.01.mean = colMeans(lossOfWealth.pre.01)
233 lossOfWealth.pre.01.mean.sel4 = lossOfWealth.pre.01.mean[sel4]
234 terminalUtility.pre.01 = utility(terminalWealth.pre.01, riskAversion)
235 terminalUtility.pre.01.mean = colMeans(terminalUtility.pre.01)
236 terminalUtility.pre.01.mean.sel4 = terminalUtility.pre.01.mean[sel4]
237 lossOfUtility.pre.01 = terminalUtility.th.01 - terminalUtility.pre.01
238 lossOfUtility.pre.01.sel4 = lossOfUtility.pre.01[,sel4]
239 lossOfUtility.pre.01.mean = colMeans(lossOfUtility.pre.01)
240 lossOfUtility.pre.01.mean.sel4 = lossOfUtility.pre.01.mean[sel4]
241 lossOfUtility.pre.01.sd = colSds(lossOfUtility.pre.01)
242 lossOfUtility.pre.01.cumMean = apply(lossOfUtility.pre.01, 2, cumMean)
243 lossOfUtility.pre.01.cumMean.sel4 = lossOfUtility.pre.01.cumMean[,sel4]
244 lossOfUtility.pre.01.cumSd = apply(lossOfUtility.pre.01, 2, cumSd)
245 lossOfUtility.pre.01.cumSd.sel4 = lossOfUtility.pre.01.cumSd[,sel4]
246 lossOfUtility.pre.01.sdCumMean = apply(lossOfUtility.pre.01.cumSd, 2, function(x)
    {x/sqrt(nn)})
247 lossOfUtility.pre.01.sdCumMean.sel4 = lossOfUtility.pre.01.sdCumMean[,sel4]
248 lossOfUtility.pre.01.cumMean.lowerCL = lossOfUtility.pre.01.cumMean - qAlpha.
    half * lossOfUtility.pre.01.sdCumMean
249 lossOfUtility.pre.01.cumMean.upperCL = lossOfUtility.pre.01.cumMean + qAlpha.

```

```

    half * lossOfUtility.pre.01.sdCumMean
250 lossOfUtility.pre.01.cumMean.lowerCL.sel4 = lossOfUtility.pre.01.cumMean.lowerCL
    [,sel4]
251 lossOfUtility.pre.01.cumMean.upperCL.sel4 = lossOfUtility.pre.01.cumMean.upperCL
    [,sel4]
252 terminalLogReturn.pre.01 = log(terminalWealth.pre.01)
253 terminalLogReturn.pre.01.mean = colMeans(terminalLogReturn.pre.01)
254 terminalLogReturn.pre.01.mean.sel4 = terminalLogReturn.pre.01.mean[sel4]
255 sdLogReturn.pre.01.mean = colMeans(sdLogReturn.pre.01)
256 sdLogReturn.pre.01.mean.sel4 = sdLogReturn.pre.01.mean[sel4]
257 annualizedSdLogReturn.pre.01 = sdLogReturn.pre.01 * sqrt(nTimePoints)
258 annualizedSdLogReturn.pre.01.mean = colMeans(annualizedSdLogReturn.pre.01)
259 terminalLogReturn.pre.01.sd = colSds(terminalLogReturn.pre.01)
260 terminalLogReturn.pre.01.sd.sel4 = terminalLogReturn.pre.01.sd[sel4]
261 excessReturn.pre.01 = terminalLogReturn.pre.01 - rent
262 sharpeRatio.pre.01 = excessReturn.pre.01 / (sqrt(nTimePoints)*sdLogReturn.pre
    .01)
263 sharpeRatio.pre.01.mean = colMeans(sharpeRatio.pre.01)
264 sharpeRatio.pre.01.mean.sel4 = sharpeRatio.pre.01.mean[sel4]
265 volOfVol.pre.01 = colSds(annualizedSdLogReturn.pre.01)
266 correlation.pre.01 = colCorrs(terminalLogReturn.pre.01, annualizedSdLogReturn.pre
    .01)
267 totalTransCost.pre.01.mean = colMeans(totalTransCost.pre.01)
268 totalTransCost.pre.01.mean.sel4 = totalTransCost.pre.01.mean[sel4]
269
270 y.rangeDiff.pre.01.sel4 = colRange(lossOfUtility.pre.01.cumMean.sel4)[2,] -
    colRange(lossOfUtility.pre.01.cumMean.sel4)[1,]
271 y.lim.pre.01.sel4 = rbind(lossOfUtility.pre.01.mean.sel4 - y.rangeDiff.pre.01.
    sel4/25,lossOfUtility.pre.01.mean.sel4 + y.rangeDiff.pre.01.sel4/25)
272
273 transformation = 1e2
274 y.title = expression(paste("Mean loss of utility",phantom(0) %%% 10^2))
275 niceplot(lossOfUtility.pre.01.cumMean.sel4[,1]*transformation, xLabels=x.labels,
    yTitle=y.title, figsPerPage=4, y.addCustom=.2, nCol=2, horizLines=T, downsample=T
    , ylim=y.lim.pre.01.sel4[,1]*transformation)
276 nicelines(lossOfUtility.pre.01.cumMean.lowerCL.sel4[,1]*transformation,
    downsample=T, col="darkgray", lty=3)
277 nicelines(lossOfUtility.pre.01.cumMean.upperCL.sel4[,1]*transformation,
    downsample=T, col="darkgray", lty=3)
278 legendText = c(expression(paste("(a)", lambda*=".01")), "Transaction costs
    strategy : Preceding", "Rebalancing strategy : Hourly"))
279 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
280
281 niceplot(lossOfUtility.pre.01.cumMean.sel4[,2]*transformation, xLabels=x.labels,
    yTitle=y.title, figsPerPage=4, y.addCustom=.2, multiPlot=T, newDev=F, horizLines=
    T, downsample=T, ylim=y.lim.pre.01.sel4[,2]*transformation*c(1,1.0001))
282 nicelines(lossOfUtility.pre.01.cumMean.lowerCL.sel4[,2]*transformation,
    downsample=T, col="darkgray", lty=3)
283 nicelines(lossOfUtility.pre.01.cumMean.upperCL.sel4[,2]*transformation,
    downsample=T, col="darkgray", lty=3)
284 legendText = c(expression(paste("(b)", lambda*=".01")), "Transaction costs
    strategy : Preceding", "Rebalancing strategy : Daily"))
285 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
286 savePlot("images/lossOfUtility_01_pre_Hourly_Daily", type="eps")
287
288 niceplot(lossOfUtility.pre.01.cumMean.sel4[,3]*transformation, xLabels=x.labels,
    yTitle=y.title, figsPerPage=4, y.addCustom=.2, nCol=2, horizLines=T, downsample=T
    , ylim=y.lim.pre.01.sel4[,3]*transformation)
289 nicelines(lossOfUtility.pre.01.cumMean.lowerCL.sel4[,3]*transformation,
    downsample=T, col="darkgray", lty=3)
290 nicelines(lossOfUtility.pre.01.cumMean.upperCL.sel4[,3]*transformation,
    downsample=T, col="darkgray", lty=3)
291 legendText = c(expression(paste("(c)", lambda*=".01")), "Transaction costs
    strategy : Preceding", "Rebalancing strategy : Monthly"))
292 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)

```



```

293 niceplot(lossOfUtility.pre.01.cumMean.sel4[,4]*transformation,xLabels=x.labels,
294           yTitle=y.title,figsPerPage=4,y.addCustom=.2,multiPlot=T,newDev=F,horizLines=
           T,downsample=T,ylim=y.lim.pre.01.sel4[,4]*transformation)
295 nicelines(lossOfUtility.pre.01.cumMean.lowerCL.sel4[,4]*transformation,
           downsample=T,col="darkgray",lty=3)
296 nicelines(lossOfUtility.pre.01.cumMean.upperCL.sel4[,4]*transformation,
           downsample=T,col="darkgray",lty=3)
297 legendText = c(expression(paste("(d)",lambda*=".01")),
           "Transaction costs
           strategy : Preceding","Rebalancing strategy : Annually"))
298 legendObject = nicelegend("topleft",legendText,bty="n",bg="white",cex=.7)
299 savePlot("images/lossOfUtility_01_pre_Monthly_Annually",type="eps")
300
301 x.ticks = 1:9
302 x.title = "Rebalancing strategy"
303 niceplot(x.ticks,lossOfUtility.pre.01.mean*transformation,xLabels=strategyNames,
           xTitle=x.title,yTitle=y.title,y.addCustom=.2)
304 abline(v=x.ticks,lty=3)
305 legendText = expression(paste("(a)",lambda*=".01"))
306 nicelegend("left",legendText,horiz=T,bty="n",bg="white",cex=.7)
307 savePlot("images/rebStrategy_v_lossOfUtility_transCost_01",type="eps")
308
309 y.title = "Sharpe ratio"
310 niceplot(x.ticks,sharpeRatio.pre.01.mean,xLabels=strategyNames,xTitle=x.title,
           yTitle=y.title)
311 abline(v=x.ticks,lty=3)
312 nicelegend("topleft",legendText,bty="n",bg="white",cex=.7)
313 savePlot("images/rebStrategy_v_sharpeRatio_transCost_01",type="eps")
314
315 # Simulated, subsequent transaction costs
316 terminalWealth.sub.01 = matrix(NA,nSims,n.entries)
317 sdWealth.sub.01 = matrix(NA,nSims,n.entries)
318 sdLogReturn.sub.01 = matrix(NA,nSims,n.entries)
319 totalTransCost.sub.01 = matrix(NA,nSims,n.entries)
320 for(k in 1:n.entries){
321   terminalWealth.sub.01[k] = rebStrategy.transCost.01[[c(k,4)]]$simWealth.
           terminal
322   sdWealth.sub.01[k] = rebStrategy.transCost.01[[c(k,4)]]$simWealth.sd
323   sdLogReturn.sub.01[k] = rebStrategy.transCost.01[[c(k,4)]]$simWealth.
           logReturn.sd
324   totalTransCost.sub.01[k] = rebStrategy.transCost.01[[c(k,4)]]$totalTransCost
325 }
326 colnames(terminalWealth.sub.01) = strategyNames
327 terminalWealth.sub.01.mean = colMeans(terminalWealth.sub.01)
328 terminalWealth.sub.01.mean.sel4 = terminalWealth.sub.01.mean[sel4]
329 sdWealth.sub.01.mean = colMeans(sdWealth.sub.01)
330 sdWealth.sub.01.mean.sel4 = sdWealth.sub.01.mean[sel4]
331 terminalWealth.sub.01.sd = colSds(terminalWealth.sub.01)
332 terminalWealth.sub.01.sd.sel4 = terminalWealth.sub.01.sd[sel4]
333 lossOfWealth.sub.01 = terminalWealth.th.01 - terminalWealth.sub.01
334 lossOfWealth.sub.01.mean = colMeans(lossOfWealth.sub.01)
335 lossOfWealth.sub.01.mean.sel4 = lossOfWealth.sub.01.mean[sel4]
336 terminalUtility.sub.01 = utility(terminalWealth.sub.01,riskAversion)
337 terminalUtility.sub.01.mean = colMeans(terminalUtility.sub.01)
338 terminalUtility.sub.01.mean.sel4 = terminalUtility.sub.01.mean[sel4]
339 lossOfUtility.sub.01 = terminalUtility.th.01 - terminalUtility.sub.01
340 lossOfUtility.sub.01.sel4 = lossOfUtility.sub.01[,sel4]
341 lossOfUtility.sub.01.mean = colMeans(lossOfUtility.sub.01)
342 lossOfUtility.sub.01.mean.sel4 = lossOfUtility.sub.01.mean[sel4]
343 lossOfUtility.sub.01.sd = colSds(lossOfUtility.sub.01)
344 lossOfUtility.sub.01.cumMean = apply(lossOfUtility.sub.01,2,cumMean)
345 lossOfUtility.sub.01.cumMean.sel4 = lossOfUtility.sub.01.cumMean[,sel4]
346 lossOfUtility.sub.01.cumSd = apply(lossOfUtility.sub.01,2,cumSd)
347 lossOfUtility.sub.01.cumSd.sel4 = lossOfUtility.sub.01.cumSd[,sel4]
348 lossOfUtility.sub.01.sdCumMean = apply(lossOfUtility.sub.01.cumSd,2,function(x)

```

```

    {x/sqrt(nn)})
349 lossOfUtility.sub.01.sdCumMean.sel4 = lossOfUtility.sub.01.sdCumMean[,sel4]
350 lossOfUtility.sub.01.cumMean.lowerCL = lossOfUtility.sub.01.cumMean - qAlpha.
    half * lossOfUtility.sub.01.sdCumMean
351 lossOfUtility.sub.01.cumMean.upperCL = lossOfUtility.sub.01.cumMean + qAlpha.
    half * lossOfUtility.sub.01.sdCumMean
352 lossOfUtility.sub.01.cumMean.lowerCL.sel4 = lossOfUtility.sub.01.cumMean.lowerCL
    [,sel4]
353 lossOfUtility.sub.01.cumMean.upperCL.sel4 = lossOfUtility.sub.01.cumMean.upperCL
    [,sel4]
354 terminalLogReturn.sub.01 = log(terminalWealth.sub.01)
355 terminalLogReturn.sub.01.mean = colMeans(terminalLogReturn.sub.01)
356 terminalLogReturn.sub.01.mean.sel4 = terminalLogReturn.sub.01.mean[sel4]
357 sdLogReturn.sub.01.mean = colMeans(sdLogReturn.sub.01)
358 sdLogReturn.sub.01.mean.sel4 = sdLogReturn.sub.01.mean[sel4]
359 annualizedSdLogReturn.sub.01 = sdLogReturn.sub.01 * sqrt(nTimePoints)
360 annualizedSdLogReturn.sub.01.mean = colMeans(annualizedSdLogReturn.sub.01)
361 terminalLogReturn.sub.01.sd = colSds(terminalLogReturn.sub.01)
362 terminalLogReturn.sub.01.sd.sel4 = terminalLogReturn.sub.01.sd[sel4]
363 excessReturn.sub.01 = terminalLogReturn.sub.01 - rent
364 sharpeRatio.sub.01 = excessReturn.sub.01 / (sqrt(nTimePoints)*sdLogReturn.sub
    .01)
365 sharpeRatio.sub.01.mean = colMeans(sharpeRatio.sub.01)
366 sharpeRatio.sub.01.mean.sel4 = sharpeRatio.sub.01.mean[sel4]
367 volOfVol.sub.01 = colSds(annualizedSdLogReturn.sub.01)
368 correlation.sub.01 = colCorrs(terminalLogReturn.sub.01, annualizedSdLogReturn.sub
    .01)
369 totalTransCost.sub.01.mean = colMeans(totalTransCost.sub.01)
370 totalTransCost.sub.01.mean.sel4 = totalTransCost.sub.01.mean[sel4]
371
372 # Plotting
373 y.rangeDiff.sub.01.sel4 = colRange(lossOfUtility.sub.01.cumMean.sel4)[2,] -
    colRange(lossOfUtility.sub.01.cumMean.sel4)[1,]
374 y.lim.sub.01.sel4 = rbind(lossOfUtility.sub.01.mean.sel4 - y.rangeDiff.sub.01.
    sel4/25, lossOfUtility.sub.01.mean.sel4 + y.rangeDiff.sub.01.sel4/25)
375
376 transformation = 1e2
377 y.title = expression(paste("Mean loss of utility",phantom(0) %%% 10^2))
378 niceplot(lossOfUtility.sub.01.cumMean.sel4[,1]*transformation, xLabels=x.labels,
    yTitle=y.title, figsPerPage=4, y.addCustom=.2, nCol=2, horizLines=T, downsample=T
    , ylim=y.lim.sub.01.sel4[,1]*transformation)
379 nicelines(lossOfUtility.sub.01.cumMean.lowerCL.sel4[,1]*transformation,
    downsample=T, col="darkgray", lty=3)
380 nicelines(lossOfUtility.sub.01.cumMean.upperCL.sel4[,1]*transformation,
    downsample=T, col="darkgray", lty=3)
381 legendText = c(expression(paste("(e)", lambda*"=.01")), "Transaction costs
    strategy : Subsequent", "Rebalancing strategy : Hourly")
382 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
383
384 niceplot(lossOfUtility.sub.01.cumMean.sel4[,2]*transformation, xLabels=x.labels,
    yTitle=y.title, figsPerPage=4, y.addCustom=.2, multiPlot=T, newDev=F, horizLines=
    T, downsample=T, ylim=y.lim.sub.01.sel4[,2]*transformation)
385 nicelines(lossOfUtility.sub.01.cumMean.lowerCL.sel4[,2]*transformation,
    downsample=T, col="darkgray", lty=3)
386 nicelines(lossOfUtility.sub.01.cumMean.upperCL.sel4[,2]*transformation,
    downsample=T, col="darkgray", lty=3)
387 legendText = c(expression(paste("(f)", lambda*"=.01")), "Transaction costs
    strategy : Subsequent", "Rebalancing strategy : Daily")
388 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
389 savePlot("images/lossOfUtility_01-sub_Hourly-Daily", type="eps")
390
391 niceplot(lossOfUtility.sub.01.cumMean.sel4[,3]*transformation, xLabels=x.labels,
    yTitle=y.title, figsPerPage=4, y.addCustom=.2, nCol=2, horizLines=T, downsample=T
    , ylim=y.lim.sub.01.sel4[,3]*transformation)
392 nicelines(lossOfUtility.sub.01.cumMean.lowerCL.sel4[,3]*transformation,

```

```

    downsample=T, col="darkgray", lty=3)
393 nicelines(lossOfUtility.sub.01.cumMean.upperCL.sel4[,3]*transformation,
    downsample=T, col="darkgray", lty=3)
394 legendText = c(expression(paste("(g)", lambda*=".01")), "Transaction costs
    strategy : Subsequent", "Rebalancing strategy : Monthly"))
395 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
396
397 niceplot(lossOfUtility.sub.01.cumMean.sel4[,4]*transformation, xLabels=x.labels,
    yTitle=y.title, figsPerPage=4, y.addCustom=.2, multiPlot=T, newDev=F, horizLines=
    T, downsample=T, ylim=y.lim.sub.01.sel4[,4]*transformation)
398 nicelines(lossOfUtility.sub.01.cumMean.lowerCL.sel4[,4]*transformation,
    downsample=T, col="darkgray", lty=3)
399 nicelines(lossOfUtility.sub.01.cumMean.upperCL.sel4[,4]*transformation,
    downsample=T, col="darkgray", lty=3)
400 legendText = c(expression(paste("(h)", lambda*=".01")), "Transaction costs
    strategy : Subsequent", "Rebalancing strategy : Annually"))
401 legendObject = nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
402 savePlot("images/lossOfUtility_01.sub.Monthly-Annually", type="eps")
403
404 # Setting up summarizing tables
405 tab1 = matrix(NA,36,5)
406 for (k in 1:9) {
407   tab1[k*4-3,] = c(terminalWealth.th.01.mean[k], 0, terminalUtility.th.01.mean[k]
    ], 0, 0)
408   tab1[k*4-2,] = c(terminalWealth.none.01.mean[k], 0, terminalUtility.none.01.mean
    [k], lossOfUtility.none.01.mean[k], lossOfUtility.none.01.sd[k])
409   tab1[k*4-1,] = c(terminalWealth.pre.01.mean[k], totalTransCost.pre.01.mean[k],
    terminalUtility.pre.01.mean[k], lossOfUtility.pre.01.mean[k], lossOfUtility.
    pre.01.sd[k])
410   tab1[k*4,] = c(terminalWealth.sub.01.mean[k], totalTransCost.sub.01.mean[k],
    terminalUtility.sub.01.mean[k], lossOfUtility.sub.01.mean[k], lossOfUtility.
    sub.01.sd[k])
411 }
412
413 tab1[,2] = tab1[,2] * 1e2
414 tab1[,4] = tab1[,4] * 1e2
415 tab1[,5] = tab1[,5] * 1e3
416
417 tab1 = round(tab1,4)
418
419 for (k in 1:36) {
420   tab1[k,2] = paste(tab1[k,2], "\\e{\\text -2}", sep="")
421   tab1[k,4] = paste(tab1[k,4], "\\e{\\text -2}", sep="")
422   tab1[k,5] = paste(tab1[k,5], "\\e{\\text -3}", sep="")
423 }
424
425 printex(tab1)
426
427 tab2 = matrix(NA,36,5)
428 for (k in 1:9) {
429   tab2[k*4-3,] = c(terminalLogReturn.th.01.mean[k], annualizedSdLogReturn.th.01.
    mean[k], sharpeRatio.th.01.mean[k], volOfVol.th.01[k], correlation.th.01[k])
430   tab2[k*4-2,] = c(terminalLogReturn.none.01.mean[k], annualizedSdLogReturn.none
    .01.mean[k], sharpeRatio.none.01.mean[k], volOfVol.none.01[k], correlation.
    none.01[k])
431   tab2[k*4-1,] = c(terminalLogReturn.pre.01.mean[k], annualizedSdLogReturn.pre
    .01.mean[k], sharpeRatio.pre.01.mean[k], volOfVol.pre.01[k], correlation.pre
    .01[k])
432   tab2[k*4,] = c(terminalLogReturn.sub.01.mean[k], annualizedSdLogReturn.sub
    .01.mean[k], sharpeRatio.sub.01.mean[k], volOfVol.sub.01[k], correlation.sub
    .01[k])
433 }
434
435 tab2[,1] = tab2[,1] * 1e2
436 tab2[,4] = tab2[,4] * 1e3

```

```

437 |
438 | tab2 = round(tab2,4)
439 |
440 | for (k in 1:36) {
441 |   tab2[k,1] = paste(tab2[k,1],"\e{\text -2}",sep="")
442 |   tab2[k,4] = paste(tab2[k,4],"\e{\text -3}",sep="")
443 | }
444 |
445 | printex(tab2)
446 |
447 | #
448 | # Calculating relevant statistics and plotting
449 | # Transaction cost proportion = .02
450 | # Preceding transaction costs
451 | #
452 |
453 | # Theoretical
454 | terminalWealth.th.02 = matrix(NA,nSims,n.entries)
455 | sdWealth.th.02 = matrix(NA,nSims,n.entries)
456 | sdLogReturn.th.02 = matrix(NA,nSims,n.entries)
457 | for (k in 1:n.entries) {
458 |   terminalWealth.th.02[,k] = rebStrategy.transCost.02[[c(k,1)]]$thWealth.
         terminal
459 |   sdWealth.th.02[,k] = rebStrategy.transCost.02[[c(k,1)]]$thWealth.sd
460 |   sdLogReturn.th.02[,k] = rebStrategy.transCost.02[[c(k,1)]]$thWealth.logReturn.
         sd
461 | }
462 | colnames(terminalWealth.th.02) = strategyNames
463 | terminalWealth.th.02.mean = colMeans(terminalWealth.th.02)
464 | terminalWealth.th.02.mean.sel4 = terminalWealth.th.02.mean[sel4]
465 | sdWealth.th.02.mean = colMeans(sdWealth.th.02)
466 | sdWealth.th.02.mean.sel4 = sdWealth.th.02.mean[sel4]
467 | terminalWealth.th.02.sd = colSds(terminalWealth.th.02)
468 | terminalWealth.th.02.sd.sel4 = terminalWealth.th.02.sd[sel4]
469 | terminalUtility.th.02 = utility(terminalWealth.th.02,riskAversion)
470 | terminalUtility.th.02.mean = colMeans(terminalUtility.th.02)
471 | terminalUtility.th.02.mean.sel4 = terminalUtility.th.02.mean[sel4]
472 | terminalLogReturn.th.02 = log(terminalWealth.th.02)
473 | terminalLogReturn.th.02.mean = colMeans(terminalLogReturn.th.02)
474 | terminalLogReturn.th.02.mean.sel4 = terminalLogReturn.th.02.mean[sel4]
475 | sdLogReturn.th.02.mean = colMeans(sdLogReturn.th.02)
476 | sdLogReturn.th.02.mean.sel4 = sdLogReturn.th.02.mean[sel4]
477 | annualizedSdLogReturn.th.02 = sdLogReturn.th.02 * sqrt(nTimePoints)
478 | annualizedSdLogReturn.th.02.mean = colMeans(annualizedSdLogReturn.th.02)
479 | terminalLogReturn.th.02.sd = colSds(terminalLogReturn.th.02)
480 | terminalLogReturn.th.02.sd.sel4 = terminalLogReturn.th.02.sd[sel4]
481 | excessReturn.th.02 = terminalLogReturn.th.02 - rent
482 | sharpeRatio.th.02 = excessReturn.th.02 / (sqrt(nTimePoints)*sdLogReturn.th.02)
483 | sharpeRatio.th.02.mean = colMeans(sharpeRatio.th.02)
484 | sharpeRatio.th.02.mean.sel4 = sharpeRatio.th.02.mean[sel4]
485 | volOfVol.th.02 = colSds(annualizedSdLogReturn.th.02)
486 | correlation.th.02 = colCorrs(terminalLogReturn.th.02,annualizedSdLogReturn.th.
         .02)
487 |
488 | # Simulated, no transaction costs
489 | terminalWealth.none.02 = matrix(NA,nSims,n.entries)
490 | sdWealth.none.02 = matrix(NA,nSims,n.entries)
491 | sdLogReturn.none.02 = matrix(NA,nSims,n.entries)
492 | for (k in 1:n.entries) {
493 |   terminalWealth.none.02[,k] = rebStrategy.transCost.02[[c(k,2)]]$simWealth.
         terminal
494 |   sdWealth.none.02[,k] = rebStrategy.transCost.02[[c(k,2)]]$simWealth.sd
495 |   sdLogReturn.none.02[,k] = rebStrategy.transCost.02[[c(k,2)]]$simWealth.
         logReturn.sd
496 | }

```

```

497 colnames(terminalWealth.none.02) = strategyNames
498 terminalWealth.none.02.mean = colMeans(terminalWealth.none.02)
499 terminalWealth.none.02.mean.sel4 = terminalWealth.none.02.mean[sel4]
500 sdWealth.none.02.mean = colMeans(sdWealth.none.02)
501 sdWealth.none.02.mean.sel4 = sdWealth.none.02.mean[sel4]
502 terminalWealth.none.02.sd = colSds(terminalWealth.none.02)
503 terminalWealth.none.02.sd.sel4 = terminalWealth.none.02.sd[sel4]
504 lossOfWealth.none.02 = terminalWealth.th.02 - terminalWealth.none.02
505 lossOfWealth.none.02.mean = colMeans(lossOfWealth.none.02)
506 lossOfWealth.none.02.mean.sel4 = lossOfWealth.none.02.mean[sel4]
507 terminalUtility.none.02 = utility(terminalWealth.none.02, riskAversion)
508 terminalUtility.none.02.mean = colMeans(terminalUtility.none.02)
509 terminalUtility.none.02.mean.sel4 = terminalUtility.none.02.mean[sel4]
510 lossOfUtility.none.02 = terminalUtility.th.02 - terminalUtility.none.02
511 lossOfUtility.none.02.mean = colMeans(lossOfUtility.none.02)
512 lossOfUtility.none.02.mean.sel4 = lossOfUtility.none.02.mean[sel4]
513 lossOfUtility.none.02.sd = colSds(lossOfUtility.none.02)
514 terminalLogReturn.none.02 = log(terminalWealth.none.02)
515 terminalLogReturn.none.02.mean = colMeans(terminalLogReturn.none.02)
516 terminalLogReturn.none.02.mean.sel4 = terminalLogReturn.none.02.mean[sel4]
517 sdLogReturn.none.02.mean = colMeans(sdLogReturn.none.02)
518 sdLogReturn.none.02.mean.sel4 = sdLogReturn.none.02.mean[sel4]
519 annualizedSdLogReturn.none.02 = sdLogReturn.none.02 * sqrt(nTimePoints)
520 annualizedSdLogReturn.none.02.mean = colMeans(annualizedSdLogReturn.none.02)
521 terminalLogReturn.none.02.sd = colSds(terminalLogReturn.none.02)
522 terminalLogReturn.none.02.sd.sel4 = terminalLogReturn.none.02.sd[sel4]
523 excessReturn.none.02 = terminalLogReturn.none.02 - rent
524 sharpeRatio.none.02 = excessReturn.none.02 / (sqrt(nTimePoints)*sdLogReturn.none
    .02)
525 sharpeRatio.none.02.mean = colMeans(sharpeRatio.none.02)
526 sharpeRatio.none.02.mean.sel4 = sharpeRatio.none.02.mean[sel4]
527 volOfVol.none.02 = colSds(annualizedSdLogReturn.none.02)
528 correlation.none.02 = colCorrs(terminalLogReturn.none.02, annualizedSdLogReturn.
    none.02)
529
530 # Simulated, preceding transaction costs
531 terminalWealth.pre.02 = matrix(NA, nSims, n.entries)
532 sdWealth.pre.02 = matrix(NA, nSims, n.entries)
533 sdLogReturn.pre.02 = matrix(NA, nSims, n.entries)
534 totalTransCost.pre.02 = matrix(NA, nSims, n.entries)
535 for (k in 1:n.entries) {
536   terminalWealth.pre.02[,k] = rebStrategy.transCost.02[[c(k,3)]]$simWealth.
    terminal
537   sdWealth.pre.02[,k] = rebStrategy.transCost.02[[c(k,3)]]$simWealth.sd
538   sdLogReturn.pre.02[,k] = rebStrategy.transCost.02[[c(k,3)]]$simWealth.
    logReturn.sd
539   totalTransCost.pre.02[,k] = rebStrategy.transCost.02[[c(k,3)]]$totalTransCost
540 }
541 colnames(terminalWealth.pre.02) = strategyNames
542 terminalWealth.pre.02.mean = colMeans(terminalWealth.pre.02)
543 terminalWealth.pre.02.mean.sel4 = terminalWealth.pre.02.mean[sel4]
544 sdWealth.pre.02.mean = colMeans(sdWealth.pre.02)
545 sdWealth.pre.02.mean.sel4 = sdWealth.pre.02.mean[sel4]
546 terminalWealth.pre.02.sd = colSds(terminalWealth.pre.02)
547 terminalWealth.pre.02.sd.sel4 = terminalWealth.pre.02.sd[sel4]
548 lossOfWealth.pre.02 = terminalWealth.th.02 - terminalWealth.pre.02
549 lossOfWealth.pre.02.mean = colMeans(lossOfWealth.pre.02)
550 lossOfWealth.pre.02.mean.sel4 = lossOfWealth.pre.02.mean[sel4]
551 terminalUtility.pre.02 = utility(terminalWealth.pre.02, riskAversion)
552 terminalUtility.pre.02.mean = colMeans(terminalUtility.pre.02)
553 terminalUtility.pre.02.mean.sel4 = terminalUtility.pre.02.mean[sel4]
554 lossOfUtility.pre.02 = terminalUtility.th.02 - terminalUtility.pre.02
555 lossOfUtility.pre.02.sel4 = lossOfUtility.pre.02[,sel4]
556 lossOfUtility.pre.02.mean = colMeans(lossOfUtility.pre.02)
557 lossOfUtility.pre.02.mean.sel4 = lossOfUtility.pre.02.mean[sel4]

```

```

558 lossOfUtility.pre.02.sd = colSds(lossOfUtility.pre.02)
559 lossOfUtility.pre.02.cumMean = apply(lossOfUtility.pre.02,2,cumMean)
560 lossOfUtility.pre.02.cumMean.sel4 = lossOfUtility.pre.02.cumMean[,sel4]
561 lossOfUtility.pre.02.cumSd = apply(lossOfUtility.pre.02,2,cumSd)
562 lossOfUtility.pre.02.cumSd.sel4 = lossOfUtility.pre.02.cumSd[,sel4]
563 lossOfUtility.pre.02.sdCumMean = apply(lossOfUtility.pre.02.cumSd,2,function(x)
    {x/sqrt(nn)})
564 lossOfUtility.pre.02.sdCumMean.sel4 = lossOfUtility.pre.02.sdCumMean[,sel4]
565 lossOfUtility.pre.02.cumMean.lowerCL = lossOfUtility.pre.02.cumMean - qAlpha.
    half * lossOfUtility.pre.02.sdCumMean
566 lossOfUtility.pre.02.cumMean.upperCL = lossOfUtility.pre.02.cumMean + qAlpha.
    half * lossOfUtility.pre.02.sdCumMean
567 lossOfUtility.pre.02.cumMean.lowerCL.sel4 = lossOfUtility.pre.02.cumMean.lowerCL
    [,sel4]
568 lossOfUtility.pre.02.cumMean.upperCL.sel4 = lossOfUtility.pre.02.cumMean.upperCL
    [,sel4]
569 terminalLogReturn.pre.02 = log(terminalWealth.pre.02)
570 terminalLogReturn.pre.02.mean = colMeans(terminalLogReturn.pre.02)
571 terminalLogReturn.pre.02.mean.sel4 = terminalLogReturn.pre.02.mean[sel4]
572 sdLogReturn.pre.02.mean = colMeans(sdLogReturn.pre.02)
573 sdLogReturn.pre.02.mean.sel4 = sdLogReturn.pre.02.mean[sel4]
574 annualizedSdLogReturn.pre.02 = sdLogReturn.pre.02 * sqrt(nTimePoints)
575 annualizedSdLogReturn.pre.02.mean = colMeans(annualizedSdLogReturn.pre.02)
576 terminalLogReturn.pre.02.sd = colSds(terminalLogReturn.pre.02)
577 terminalLogReturn.pre.02.sd.sel4 = terminalLogReturn.pre.02.sd[sel4]
578 excessReturn.pre.02 = terminalLogReturn.pre.02 - rent
579 sharpeRatio.pre.02 = excessReturn.pre.02 / (sqrt(nTimePoints)*sdLogReturn.pre
    .02)
580 sharpeRatio.pre.02.mean = colMeans(sharpeRatio.pre.02)
581 sharpeRatio.pre.02.mean.sel4 = sharpeRatio.pre.02.mean[sel4]
582 volOfVol.pre.02 = colSds(annualizedSdLogReturn.pre.02)
583 correlation.pre.02 = colCorrs(terminalLogReturn.pre.02, annualizedSdLogReturn.pre
    .02)
584 totalTransCost.pre.02.mean = colMeans(totalTransCost.pre.02)
585 totalTransCost.pre.02.mean.sel4 = totalTransCost.pre.02.mean[sel4]
586
587 y.rangeDiff.pre.02.sel4 = colRange(lossOfUtility.pre.02.cumMean.sel4)[2,] -
    colRange(lossOfUtility.pre.02.cumMean.sel4)[1,]
588 y.lim.pre.02.sel4 = rbind(lossOfUtility.pre.02.mean.sel4 - y.rangeDiff.pre.02.
    sel4/25,lossOfUtility.pre.02.mean.sel4 + y.rangeDiff.pre.02.sel4/25)
589
590 transformation = 1e2
591 y.title = expression(paste("Mean loss of utility",phantom(0) %%% 10^2))
592 niceplot(lossOfUtility.pre.02.cumMean.sel4[,1]*transformation, xLabels=x.labels,
    yTitle=y.title, figsPerPage=4,y.addCustom=.2,nCol=2,horizLines=T,downsample=T
    ,ylim=y.lim.pre.02.sel4[,1]*transformation)
593 nicelines(lossOfUtility.pre.02.cumMean.lowerCL.sel4[,1]*transformation,
    downsample=T,col="darkgray",lty=3)
594 nicelines(lossOfUtility.pre.02.cumMean.upperCL.sel4[,1]*transformation,
    downsample=T,col="darkgray",lty=3)
595 legendText = c(expression(paste("(a)",lambda*=".02")), "Transaction costs
    strategy : Preceding", "Rebalancing strategy : Hourly")
596 nicelegend("topleft",legendText,bty="n",bg="white",cex=.7)
597
598 niceplot(lossOfUtility.pre.02.cumMean.sel4[,2]*transformation, xLabels=x.labels,
    yTitle=y.title, figsPerPage=4,y.addCustom=.2,multiPlot=T,newDev=F,horizLines=
    T,downsample=T,ylim=y.lim.pre.02.sel4[,2]*transformation*c(1,1.0001))
599 nicelines(lossOfUtility.pre.02.cumMean.lowerCL.sel4[,2]*transformation,
    downsample=T,col="darkgray",lty=3)
600 nicelines(lossOfUtility.pre.02.cumMean.upperCL.sel4[,2]*transformation,
    downsample=T,col="darkgray",lty=3)
601 legendText = c(expression(paste("(b)",lambda*=".02")), "Transaction costs
    strategy : Preceding", "Rebalancing strategy : Daily")
602 nicelegend("topleft",legendText,bty="n",bg="white",cex=.7)
603 savePlot("images/lossOfUtility_02_pre_Hourly_Daily",type="eps")

```

```

604 niceplot(lossOfUtility.pre.02.cumMean.sel4[,3]*transformation, xLabels=x.labels,
605           yTitle=y.title, figsPerPage=4, y.addCustom=.2, nCol=2, horizLines=T, downsample=T,
           ylim=y.lim.pre.02.sel4[,3]*transformation)
606 nicelines(lossOfUtility.pre.02.cumMean.lowerCL.sel4[,3]*transformation,
           downsample=T, col="darkgray", lty=3)
607 nicelines(lossOfUtility.pre.02.cumMean.upperCL.sel4[,3]*transformation,
           downsample=T, col="darkgray", lty=3)
608 legendText = c(expression(paste("(c)", lambda*=".02")), "Transaction costs
           strategy : Preceding", "Rebalancing strategy : Monthly"))
609 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
610
611 niceplot(lossOfUtility.pre.02.cumMean.sel4[,4]*transformation, xLabels=x.labels,
           yTitle=y.title, figsPerPage=4, y.addCustom=.2, multiPlot=T, newDev=F, horizLines=
           T, downsample=T, ylim=y.lim.pre.02.sel4[,4]*transformation)
612 nicelines(lossOfUtility.pre.02.cumMean.lowerCL.sel4[,4]*transformation,
           downsample=T, col="darkgray", lty=3)
613 nicelines(lossOfUtility.pre.02.cumMean.upperCL.sel4[,4]*transformation,
           downsample=T, col="darkgray", lty=3)
614 legendText = c(expression(paste("(d)", lambda*=".02")), "Transaction costs
           strategy : Preceding", "Rebalancing strategy : Annually"))
615 legendObject = nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
616 savePlot("images/lossOfUtility_02_pre-Monthly-Annually", type="eps")
617
618 x.ticks = 1:9
619 x.title = "Rebalancing strategy"
620 niceplot(x.ticks, lossOfUtility.pre.02.mean*transformation, xLabels=strategyNames,
           xTitle=x.title, yTitle=y.title, y.addCustom=.2)
621 abline(v=x.ticks, lty=3)
622 legendText = expression(paste("(b)", lambda*=".02"))
623 nicelegend("left", legendText, horiz=T, bty="n", bg="white", cex=.7)
624 savePlot("images/rebStrategy_v_lossOfUtility_transCost_02", type="eps")
625
626 y.title = "Sharpe ratio"
627 niceplot(x.ticks, sharpeRatio.pre.02.mean, xLabels=strategyNames, xTitle=x.title,
           yTitle=y.title)
628 abline(v=x.ticks, lty=3)
629 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
630 savePlot("images/rebStrategy_v_sharpeRatio_transCost_02", type="eps")
631
632 # Simulated, subsequent transaction costs
633 terminalWealth.sub.02 = matrix(NA, nSims, n.entries)
634 sdWealth.sub.02 = matrix(NA, nSims, n.entries)
635 sdLogReturn.sub.02 = matrix(NA, nSims, n.entries)
636 totalTransCost.sub.02 = matrix(NA, nSims, n.entries)
637 for (k in 1:n.entries) {
638   terminalWealth.sub.02[,k] = rebStrategy.transCost.02[[c(k,4)]]$simWealth.
           terminal
639   sdWealth.sub.02[,k] = rebStrategy.transCost.02[[c(k,4)]]$simWealth.sd
640   sdLogReturn.sub.02[,k] = rebStrategy.transCost.02[[c(k,4)]]$simWealth.
           logReturn.sd
641   totalTransCost.sub.02[,k] = rebStrategy.transCost.02[[c(k,4)]]$totalTransCost
642 }
643 colnames(terminalWealth.sub.02) = strategyNames
644 terminalWealth.sub.02.mean = colMeans(terminalWealth.sub.02)
645 terminalWealth.sub.02.mean.sel4 = terminalWealth.sub.02.mean[sel4]
646 sdWealth.sub.02.mean = colMeans(sdWealth.sub.02)
647 sdWealth.sub.02.mean.sel4 = sdWealth.sub.02.mean[sel4]
648 terminalWealth.sub.02.sd = colSds(terminalWealth.sub.02)
649 terminalWealth.sub.02.sd.sel4 = terminalWealth.sub.02.sd[sel4]
650 lossOfWealth.sub.02 = terminalWealth.th.02 - terminalWealth.sub.02
651 lossOfWealth.sub.02.mean = colMeans(lossOfWealth.sub.02)
652 lossOfWealth.sub.02.mean.sel4 = lossOfWealth.sub.02.mean[sel4]
653 terminalUtility.sub.02 = utility(terminalWealth.sub.02, riskAversion)
654 terminalUtility.sub.02.mean = colMeans(terminalUtility.sub.02)

```

```

655 terminalUtility.sub.02.mean.sel4 = terminalUtility.sub.02.mean[sel4]
656 lossOfUtility.sub.02 = terminalUtility.th.02 - terminalUtility.sub.02
657 lossOfUtility.sub.02.sel4 = lossOfUtility.sub.02[,sel4]
658 lossOfUtility.sub.02.mean = colMeans(lossOfUtility.sub.02)
659 lossOfUtility.sub.02.mean.sel4 = lossOfUtility.sub.02.mean[sel4]
660 lossOfUtility.sub.02.sd = colSds(lossOfUtility.sub.02)
661 lossOfUtility.sub.02.cumMean = apply(lossOfUtility.sub.02,2,cumMean)
662 lossOfUtility.sub.02.cumMean.sel4 = lossOfUtility.sub.02.cumMean[,sel4]
663 lossOfUtility.sub.02.cumSd = apply(lossOfUtility.sub.02,2,cumSd)
664 lossOfUtility.sub.02.cumSd.sel4 = lossOfUtility.sub.02.cumSd[,sel4]
665 lossOfUtility.sub.02.sdCumMean = apply(lossOfUtility.sub.02.cumSd,2,function(x)
    {x/sqrt(nn)})
666 lossOfUtility.sub.02.sdCumMean.sel4 = lossOfUtility.sub.02.sdCumMean[,sel4]
667 lossOfUtility.sub.02.cumMean.lowerCL = lossOfUtility.sub.02.cumMean - qAlpha.
    half * lossOfUtility.sub.02.sdCumMean
668 lossOfUtility.sub.02.cumMean.upperCL = lossOfUtility.sub.02.cumMean + qAlpha.
    half * lossOfUtility.sub.02.sdCumMean
669 lossOfUtility.sub.02.cumMean.lowerCL.sel4 = lossOfUtility.sub.02.cumMean.lowerCL
    [,sel4]
670 lossOfUtility.sub.02.cumMean.upperCL.sel4 = lossOfUtility.sub.02.cumMean.upperCL
    [,sel4]
671 terminalLogReturn.sub.02 = log(terminalWealth.sub.02)
672 terminalLogReturn.sub.02.mean = colMeans(terminalLogReturn.sub.02)
673 terminalLogReturn.sub.02.mean.sel4 = terminalLogReturn.sub.02.mean[sel4]
674 sdLogReturn.sub.02.mean = colMeans(sdLogReturn.sub.02)
675 sdLogReturn.sub.02.mean.sel4 = sdLogReturn.sub.02.mean[sel4]
676 annualizedSdLogReturn.sub.02 = sdLogReturn.sub.02 * sqrt(nTimePoints)
677 annualizedSdLogReturn.sub.02.mean = colMeans(annualizedSdLogReturn.sub.02)
678 terminalLogReturn.sub.02.sd = colSds(terminalLogReturn.sub.02)
679 terminalLogReturn.sub.02.sd.sel4 = terminalLogReturn.sub.02.sd[sel4]
680 excessReturn.sub.02 = terminalLogReturn.sub.02 - rent
681 sharpeRatio.sub.02 = excessReturn.sub.02 / (sqrt(nTimePoints)*sdLogReturn.sub
    .02)
682 sharpeRatio.sub.02.mean = colMeans(sharpeRatio.sub.02)
683 sharpeRatio.sub.02.mean.sel4 = sharpeRatio.sub.02.mean[sel4]
684 volOfVol.sub.02 = colSds(annualizedSdLogReturn.sub.02)
685 correlation.sub.02 = colCorrs(terminalLogReturn.sub.02,annualizedSdLogReturn.sub
    .02)
686 totalTransCost.sub.02.mean = colMeans(totalTransCost.sub.02)
687 totalTransCost.sub.02.mean.sel4 = totalTransCost.sub.02.mean[sel4]
688
689 # Plotting
690 y.rangeDiff.sub.02.sel4 = colRange(lossOfUtility.sub.02.cumMean.sel4)[2,] -
    colRange(lossOfUtility.sub.02.cumMean.sel4)[1,]
691 y.lim.sub.02.sel4 = rbind(lossOfUtility.sub.02.mean.sel4 - y.rangeDiff.sub.02.
    sel4/25,lossOfUtility.sub.02.mean.sel4 + y.rangeDiff.sub.02.sel4/25)
692
693 transformation = 1e2
694 y.title = expression(paste("Mean loss of utility",phantom(0) %*% 10^2))
695 niceplot(lossOfUtility.sub.02.cumMean.sel4[,1]*transformation, xLabels=x.labels,
    yTitle=y.title, figsPerPage=4,y.addCustom=.2,nCol=2,horizLines=T,downsample=T
    ,ylim=y.lim.sub.02.sel4[,1]*transformation)
696 nicelines(lossOfUtility.sub.02.cumMean.lowerCL.sel4[,1]*transformation,
    downsample=T,col="darkgray",lty=3)
697 nicelines(lossOfUtility.sub.02.cumMean.upperCL.sel4[,1]*transformation,
    downsample=T,col="darkgray",lty=3)
698 legendText = c(expression(paste("(e)",lambda*"=.02")), "Transaction costs
    strategy : Subsequent", "Rebalancing strategy : Hourly")
699 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
700
701 niceplot(lossOfUtility.sub.02.cumMean.sel4[,2]*transformation, xLabels=x.labels,
    yTitle=y.title, figsPerPage=4,y.addCustom=.2, multiPlot=T, newDev=F, horizLines=
    T, downsample=T, ylim=y.lim.sub.02.sel4[,2]*transformation)
702 nicelines(lossOfUtility.sub.02.cumMean.lowerCL.sel4[,2]*transformation,
    downsample=T,col="darkgray",lty=3)

```



```

703 nicelines(lossOfUtility.sub.02.cumMean.upperCL.sel4[,2]*transformation,
704           downsamples=T,col="darkgray",lty=3)
704 legendText = c(expression(paste("(f)",lambda*=".02")), "Transaction costs
705           strategy : Subsequent", "Rebalancing strategy : Daily"))
705 nicelegend("topleft",legendText,bty="n",bg="white",cex=.7)
706 savePlot("images/lossOfUtility_02_sub_Hourly_Daily",type="eps")
707
708 niceplot(lossOfUtility.sub.02.cumMean.sel4[,3]*transformation,xLabels=x.labels,
709           yTitle=y.title,figsPerPage=4,y.addCustom=.2,nCol=2,horizLines=T,downsample=T
710           ,ylim=y.lim.sub.02.sel4[,3]*transformation)
709 nicelines(lossOfUtility.sub.02.cumMean.lowerCL.sel4[,3]*transformation,
710           downsamples=T,col="darkgray",lty=3)
710 nicelines(lossOfUtility.sub.02.cumMean.upperCL.sel4[,3]*transformation,
711           downsamples=T,col="darkgray",lty=3)
711 legendText = c(expression(paste("(g)",lambda*=".02")), "Transaction costs
712           strategy : Subsequent", "Rebalancing strategy : Monthly"))
712 nicelegend("topleft",legendText,bty="n",bg="white",cex=.7)
713
714 niceplot(lossOfUtility.sub.02.cumMean.sel4[,4]*transformation,xLabels=x.labels,
715           yTitle=y.title,figsPerPage=4,y.addCustom=.2,multiPlot=T,newDev=F,horizLines=
716           T,downsample=T,ylim=y.lim.sub.02.sel4[,4]*transformation)
715 nicelines(lossOfUtility.sub.02.cumMean.lowerCL.sel4[,4]*transformation,
716           downsamples=T,col="darkgray",lty=3)
716 nicelines(lossOfUtility.sub.02.cumMean.upperCL.sel4[,4]*transformation,
717           downsamples=T,col="darkgray",lty=3)
717 legendText = c(expression(paste("(h)",lambda*=".02")), "Transaction costs
718           strategy : Subsequent", "Rebalancing strategy : Annually"))
718 legendObject = nicelegend("topleft",legendText,bty="n",bg="white",cex=.7)
719 savePlot("images/lossOfUtility_02_sub_Monthly_Annually",type="eps")
720
721 # Setting up summarizing tables
722 tab1 = matrix(NA,36,5)
723 for (k in 1:9) {
724   tab1[k*4-3,] = c(terminalWealth.th.02.mean[k],0,terminalUtility.th.02.mean[k]
725   ,0,0)
725   tab1[k*4-2,] = c(terminalWealth.none.02.mean[k],0,terminalUtility.none.02.mean
726   [k],lossOfUtility.none.02.mean[k],lossOfUtility.none.02.sd[k])
726   tab1[k*4-1,] = c(terminalWealth.pre.02.mean[k],totalTransCost.pre.02.mean[k],
727   terminalUtility.pre.02.mean[k],lossOfUtility.pre.02.mean[k],lossOfUtility.
728   pre.02.sd[k])
727   tab1[k*4,] = c(terminalWealth.sub.02.mean[k],totalTransCost.sub.02.mean[k],
729   terminalUtility.sub.02.mean[k],lossOfUtility.sub.02.mean[k],lossOfUtility.
730   sub.02.sd[k])
731 }
729
730 tab1[,2] = tab1[,2] * 1e2
731 tab1[,4] = tab1[,4] * 1e2
732 tab1[,5] = tab1[,5] * 1e3
733
734 tab1 = round(tab1,4)
735
736 for (k in 1:36) {
737   tab1[k,2] = paste(tab1[k,2],"\\e{\\text{-2}}",sep="")
738   tab1[k,4] = paste(tab1[k,4],"\\e{\\text{-2}}",sep="")
739   tab1[k,5] = paste(tab1[k,5],"\\e{\\text{-3}}",sep="")
740 }
741
742 printex(tab1)
743
744 tab2 = matrix(NA,36,5)
745 for (k in 1:9) {
746   tab2[k*4-3,] = c(terminalLogReturn.th.02.mean[k],annualizedSdLogReturn.th.02.
747   mean[k],sharpeRatio.th.02.mean[k],volOfVol.th.02[k],correlation.th.02[k])
748   tab2[k*4-2,] = c(terminalLogReturn.none.02.mean[k],annualizedSdLogReturn.none
749   .02.mean[k],sharpeRatio.none.02.mean[k],volOfVol.none.02[k],correlation.

```

```

748     none.02[k])
749     tab2[k*4-1,] = c(terminalLogReturn.pre.02.mean[k], annualizedSdLogReturn.pre
      .02.mean[k], sharpeRatio.pre.02.mean[k], volOfVol.pre.02[k], correlation.pre
      .02[k])
749     tab2[k*4,] = c(terminalLogReturn.sub.02.mean[k], annualizedSdLogReturn.sub
      .02.mean[k], sharpeRatio.sub.02.mean[k], volOfVol.sub.02[k], correlation.sub
      .02[k])
750 }
751
752 tab2[,1] = tab2[,1] * 1e2
753 tab2[,4] = tab2[,4] * 1e3
754
755 tab2 = round(tab2,4)
756
757 for (k in 1:36) {
758     tab2[k,1] = paste(tab2[k,1], "\\e{\\text -2}", sep="")
759     tab2[k,4] = paste(tab2[k,4], "\\e{\\text -3}", sep="")
760 }
761
762 printex(tab2)
763
764 #
765 # Calculating relevant statistics and plotting
766 # Transaction cost proportion = .03
767 #
768
769 # Theoretical
770 terminalWealth.th.03 = matrix(NA, nSims, n.entries)
771 sdWealth.th.03 = matrix(NA, nSims, n.entries)
772 sdLogReturn.th.03 = matrix(NA, nSims, n.entries)
773 for (k in 1:n.entries) {
774     terminalWealth.th.03[,k] = rebStrategy.transCost.03[[c(k,1)]]$thWealth.
      terminal
775     sdWealth.th.03[,k] = rebStrategy.transCost.03[[c(k,1)]]$thWealth.sd
776     sdLogReturn.th.03[,k] = rebStrategy.transCost.03[[c(k,1)]]$thWealth.logReturn.
      sd
777 }
778 colnames(terminalWealth.th.03) = strategyNames
779 terminalWealth.th.03.mean = colMeans(terminalWealth.th.03)
780 terminalWealth.th.03.mean.sel4 = terminalWealth.th.03.mean[sel4]
781 sdWealth.th.03.mean = colMeans(sdWealth.th.03)
782 sdWealth.th.03.mean.sel4 = sdWealth.th.03.mean[sel4]
783 terminalWealth.th.03.sd = colSds(terminalWealth.th.03)
784 terminalWealth.th.03.sd.sel4 = terminalWealth.th.03.sd[sel4]
785 terminalUtility.th.03 = utility(terminalWealth.th.03, riskAversion)
786 terminalUtility.th.03.mean = colMeans(terminalUtility.th.03)
787 terminalUtility.th.03.mean.sel4 = terminalUtility.th.03.mean[sel4]
788 terminalLogReturn.th.03 = log(terminalWealth.th.03)
789 terminalLogReturn.th.03.mean = colMeans(terminalLogReturn.th.03)
790 terminalLogReturn.th.03.mean.sel4 = terminalLogReturn.th.03.mean[sel4]
791 sdLogReturn.th.03.mean = colMeans(sdLogReturn.th.03)
792 sdLogReturn.th.03.mean.sel4 = sdLogReturn.th.03.mean[sel4]
793 annualizedSdLogReturn.th.03 = sdLogReturn.th.03 * sqrt(nTimePoints)
794 annualizedSdLogReturn.th.03.mean = colMeans(annualizedSdLogReturn.th.03)
795 terminalLogReturn.th.03.sd = colSds(terminalLogReturn.th.03)
796 terminalLogReturn.th.03.sd.sel4 = terminalLogReturn.th.03.sd[sel4]
797 excessReturn.th.03 = terminalLogReturn.th.03 - rent
798 sharpeRatio.th.03 = excessReturn.th.03 / (sqrt(nTimePoints)*sdLogReturn.th.03)
799 sharpeRatio.th.03.mean = colMeans(sharpeRatio.th.03)
800 sharpeRatio.th.03.mean.sel4 = sharpeRatio.th.03.mean[sel4]
801 volOfVol.th.03 = colSds(annualizedSdLogReturn.th.03)
802 correlation.th.03 = colCorrs(terminalLogReturn.th.03, annualizedSdLogReturn.th
      .03)
803
804 # Simulated, no transaction costs

```

```

805 terminalWealth.none.03 = matrix(NA,nSims,n.entries)
806 sdWealth.none.03 = matrix(NA,nSims,n.entries)
807 sdLogReturn.none.03 = matrix(NA,nSims,n.entries)
808 for (k in 1:n.entries) {
809   terminalWealth.none.03[,k] = rebStrategy.transCost.03[[c(k,2)]]$simWealth.
      terminal
810   sdWealth.none.03[,k] = rebStrategy.transCost.03[[c(k,2)]]$simWealth.sd
811   sdLogReturn.none.03[,k] = rebStrategy.transCost.03[[c(k,2)]]$simWealth.
      logReturn.sd
812 }
813 colnames(terminalWealth.none.03) = strategyNames
814 terminalWealth.none.03.mean = colMeans(terminalWealth.none.03)
815 terminalWealth.none.03.mean.sel4 = terminalWealth.none.03.mean[sel4]
816 sdWealth.none.03.mean = colMeans(sdWealth.none.03)
817 sdWealth.none.03.mean.sel4 = sdWealth.none.03.mean[sel4]
818 terminalWealth.none.03.sd = colSds(terminalWealth.none.03)
819 terminalWealth.none.03.sd.sel4 = terminalWealth.none.03.sd[sel4]
820 lossOfWealth.none.03 = terminalWealth.th.03 - terminalWealth.none.03
821 lossOfWealth.none.03.mean = colMeans(lossOfWealth.none.03)
822 lossOfWealth.none.03.mean.sel4 = lossOfWealth.none.03.mean[sel4]
823 terminalUtility.none.03 = utility(terminalWealth.none.03,riskAversion)
824 terminalUtility.none.03.mean = colMeans(terminalUtility.none.03)
825 terminalUtility.none.03.mean.sel4 = terminalUtility.none.03.mean[sel4]
826 lossOfUtility.none.03 = terminalUtility.th.03 - terminalUtility.none.03
827 lossOfUtility.none.03.mean = colMeans(lossOfUtility.none.03)
828 lossOfUtility.none.03.mean.sel4 = lossOfUtility.none.03.mean[sel4]
829 lossOfUtility.none.03.sd = colSds(lossOfUtility.none.03)
830 terminalLogReturn.none.03 = log(terminalWealth.none.03)
831 terminalLogReturn.none.03.mean = colMeans(terminalLogReturn.none.03)
832 terminalLogReturn.none.03.mean.sel4 = terminalLogReturn.none.03.mean[sel4]
833 sdLogReturn.none.03.mean = colMeans(sdLogReturn.none.03)
834 sdLogReturn.none.03.mean.sel4 = sdLogReturn.none.03.mean[sel4]
835 annualizedSdLogReturn.none.03 = sdLogReturn.none.03 * sqrt(nTimePoints)
836 annualizedSdLogReturn.none.03.mean = colMeans(annualizedSdLogReturn.none.03)
837 terminalLogReturn.none.03.sd = colSds(terminalLogReturn.none.03)
838 terminalLogReturn.none.03.sd.sel4 = terminalLogReturn.none.03.sd[sel4]
839 excessReturn.none.03 = terminalLogReturn.none.03 - rent
840 sharpeRatio.none.03 = excessReturn.none.03 / (sqrt(nTimePoints)*sdLogReturn.none
      .03)
841 sharpeRatio.none.03.mean = colMeans(sharpeRatio.none.03)
842 sharpeRatio.none.03.mean.sel4 = sharpeRatio.none.03.mean[sel4]
843 volOfVol.none.03 = colSds(annualizedSdLogReturn.none.03)
844 correlation.none.03 = colCorrs(terminalLogReturn.none.03,annualizedSdLogReturn.
      none.03)
845
846 # Simulated, preceding transaction costs
847 terminalWealth.pre.03 = matrix(NA,nSims,n.entries)
848 sdWealth.pre.03 = matrix(NA,nSims,n.entries)
849 sdLogReturn.pre.03 = matrix(NA,nSims,n.entries)
850 totalTransCost.pre.03 = matrix(NA,nSims,n.entries)
851 for (k in 1:n.entries) {
852   terminalWealth.pre.03[,k] = rebStrategy.transCost.03[[c(k,3)]]$simWealth.
      terminal
853   sdWealth.pre.03[,k] = rebStrategy.transCost.03[[c(k,3)]]$simWealth.sd
854   sdLogReturn.pre.03[,k] = rebStrategy.transCost.03[[c(k,3)]]$simWealth.
      logReturn.sd
855   totalTransCost.pre.03[,k] = rebStrategy.transCost.03[[c(k,3)]]$totalTransCost
856 }
857 colnames(terminalWealth.pre.03) = strategyNames
858 terminalWealth.pre.03.mean = colMeans(terminalWealth.pre.03)
859 terminalWealth.pre.03.mean.sel4 = terminalWealth.pre.03.mean[sel4]
860 sdWealth.pre.03.mean = colMeans(sdWealth.pre.03)
861 sdWealth.pre.03.mean.sel4 = sdWealth.pre.03.mean[sel4]
862 terminalWealth.pre.03.sd = colSds(terminalWealth.pre.03)
863 terminalWealth.pre.03.sd.sel4 = terminalWealth.pre.03.sd[sel4]

```

```

864 lossOfWealth.pre.03 = terminalWealth.th.03 - terminalWealth.pre.03
865 lossOfWealth.pre.03.mean = colMeans(lossOfWealth.pre.03)
866 lossOfWealth.pre.03.mean.sel4 = lossOfWealth.pre.03.mean[sel4]
867 terminalUtility.pre.03 = utility(terminalWealth.pre.03,riskAversion)
868 terminalUtility.pre.03.mean = colMeans(terminalUtility.pre.03)
869 terminalUtility.pre.03.mean.sel4 = terminalUtility.pre.03.mean[sel4]
870 lossOfUtility.pre.03 = terminalUtility.th.03 - terminalUtility.pre.03
871 lossOfUtility.pre.03.sel4 = lossOfUtility.pre.03[,sel4]
872 lossOfUtility.pre.03.mean = colMeans(lossOfUtility.pre.03)
873 lossOfUtility.pre.03.mean.sel4 = lossOfUtility.pre.03.mean[sel4]
874 lossOfUtility.pre.03.sd = colSds(lossOfUtility.pre.03)
875 lossOfUtility.pre.03.cumMean = apply(lossOfUtility.pre.03,2,cumMean)
876 lossOfUtility.pre.03.cumMean.sel4 = lossOfUtility.pre.03.cumMean[,sel4]
877 lossOfUtility.pre.03.cumSd = apply(lossOfUtility.pre.03,2,cumSd)
878 lossOfUtility.pre.03.cumSd.sel4 = lossOfUtility.pre.03.cumSd[,sel4]
879 lossOfUtility.pre.03.sdCumMean = apply(lossOfUtility.pre.03.cumSd,2,function(x)
      {x/sqrt(nn)})
880 lossOfUtility.pre.03.sdCumMean.sel4 = lossOfUtility.pre.03.sdCumMean[,sel4]
881 lossOfUtility.pre.03.cumMean.lowerCL = lossOfUtility.pre.03.cumMean - qAlpha.
      half * lossOfUtility.pre.03.sdCumMean
882 lossOfUtility.pre.03.cumMean.upperCL = lossOfUtility.pre.03.cumMean + qAlpha.
      half * lossOfUtility.pre.03.sdCumMean
883 lossOfUtility.pre.03.cumMean.lowerCL.sel4 = lossOfUtility.pre.03.cumMean.lowerCL
      [,sel4]
884 lossOfUtility.pre.03.cumMean.upperCL.sel4 = lossOfUtility.pre.03.cumMean.upperCL
      [,sel4]
885 terminalLogReturn.pre.03 = log(terminalWealth.pre.03)
886 terminalLogReturn.pre.03.mean = colMeans(terminalLogReturn.pre.03)
887 terminalLogReturn.pre.03.mean.sel4 = terminalLogReturn.pre.03.mean[sel4]
888 sdLogReturn.pre.03.mean = colMeans(sdLogReturn.pre.03)
889 sdLogReturn.pre.03.mean.sel4 = sdLogReturn.pre.03.mean[sel4]
890 annualizedSdLogReturn.pre.03 = sdLogReturn.pre.03 * sqrt(nTimePoints)
891 annualizedSdLogReturn.pre.03.mean = colMeans(annualizedSdLogReturn.pre.03)
892 terminalLogReturn.pre.03.sd = colSds(terminalLogReturn.pre.03)
893 terminalLogReturn.pre.03.sd.sel4 = terminalLogReturn.pre.03.sd[sel4]
894 excessReturn.pre.03 = terminalLogReturn.pre.03 - rent
895 sharpeRatio.pre.03 = excessReturn.pre.03 / (sqrt(nTimePoints)*sdLogReturn.pre
      .03)
896 sharpeRatio.pre.03.mean = colMeans(sharpeRatio.pre.03)
897 sharpeRatio.pre.03.mean.sel4 = sharpeRatio.pre.03.mean[sel4]
898 volOfVol.pre.03 = colSds(annualizedSdLogReturn.pre.03)
899 correlation.pre.03 = colCorrs(terminalLogReturn.pre.03,annualizedSdLogReturn.pre
      .03)
900 totalTransCost.pre.03.mean = colMeans(totalTransCost.pre.03)
901 totalTransCost.pre.03.mean.sel4 = totalTransCost.pre.03.mean[sel4]
902
903 y.rangeDiff.pre.03.sel4 = colRange(lossOfUtility.pre.03.cumMean.sel4)[2,] -
      colRange(lossOfUtility.pre.03.cumMean.sel4)[1,]
904 y.lim.pre.03.sel4 = rbind(lossOfUtility.pre.03.mean.sel4 - y.rangeDiff.pre.03.
      sel4/25,lossOfUtility.pre.03.mean.sel4 + y.rangeDiff.pre.03.sel4/25)
905
906 transformation = 1e2
907 y.title = expression(paste("Mean loss of utility",phantom(0) %*% 10^2))
908 niceplot(lossOfUtility.pre.03.cumMean.sel4[,1]*transformation,xLabels=x.labels,
      yTitle=y.title,figsPerPage=4,y.addCustom=.2,nCol=2,horizLines=T,downsample=T
      ,ylim=y.lim.pre.03.sel4[,1]*transformation)
909 nicelines(lossOfUtility.pre.03.cumMean.lowerCL.sel4[,1]*transformation,
      downsample=T,col="darkgray",lty=3)
910 nicelines(lossOfUtility.pre.03.cumMean.upperCL.sel4[,1]*transformation,
      downsample=T,col="darkgray",lty=3)
911 legendText = c(expression(paste("(a)",lambda*"=.03")), "Transaction costs
      strategy : Preceding", "Rebalancing strategy : Hourly")
912 nicelegend("topleft",legendText,bty="n",bg="white",cex=.7)
913
914 niceplot(lossOfUtility.pre.03.cumMean.sel4[,2]*transformation,xLabels=x.labels,

```

```

    yTitle=y.title , figsPerPage=4,y.addCustom=.2 , multiPlot=T,newDev=F, horizLines=
    T,downsample=T,ylim=y.lim.pre.03.sel4[,2]*transformation*c(1,1.0001))
915 nicelines(lossOfUtility.pre.03.cumMean.lowerCL.sel4[,2]*transformation ,
    downsample=T,col="darkgray",lty=3)
916 nicelines(lossOfUtility.pre.03.cumMean.upperCL.sel4[,2]*transformation ,
    downsample=T,col="darkgray",lty=3)
917 legendText = c(expression(paste("(b)",lambda*=".03")), "Transaction costs
    strategy : Preceding", "Rebalancing strategy : Daily"))
918 nicelegend("topleft",legendText,bty="n",bg="white",cex=.7)
919 savePlot("images/lossOfUtility_03_pre_Hourly_Daily",type="eps")
920
921 niceplot(lossOfUtility.pre.03.cumMean.sel4[,3]*transformation , xLabels=x.labels ,
    yTitle=y.title , figsPerPage=4,y.addCustom=.2,nCol=2,horizLines=T,downsample=T
    ,ylim=y.lim.pre.03.sel4[,3]*transformation)
922 nicelines(lossOfUtility.pre.03.cumMean.lowerCL.sel4[,3]*transformation ,
    downsample=T,col="darkgray",lty=3)
923 nicelines(lossOfUtility.pre.03.cumMean.upperCL.sel4[,3]*transformation ,
    downsample=T,col="darkgray",lty=3)
924 legendText = c(expression(paste("(c)",lambda*=".03")), "Transaction costs
    strategy : Preceding", "Rebalancing strategy : Monthly"))
925 nicelegend("topleft",legendText,bty="n",bg="white",cex=.7)
926
927 niceplot(lossOfUtility.pre.03.cumMean.sel4[,4]*transformation , xLabels=x.labels ,
    yTitle=y.title , figsPerPage=4,y.addCustom=.2 , multiPlot=T,newDev=F, horizLines=
    T,downsample=T,ylim=y.lim.pre.03.sel4[,4]*transformation)
928 nicelines(lossOfUtility.pre.03.cumMean.lowerCL.sel4[,4]*transformation ,
    downsample=T,col="darkgray",lty=3)
929 nicelines(lossOfUtility.pre.03.cumMean.upperCL.sel4[,4]*transformation ,
    downsample=T,col="darkgray",lty=3)
930 legendText = c(expression(paste("(d)",lambda*=".03")), "Transaction costs
    strategy : Preceding", "Rebalancing strategy : Annually"))
931 legendObject = nicelegend("topleft",legendText,bty="n",bg="white",cex=.7)
932 savePlot("images/lossOfUtility_03_pre_Monthly_Annually",type="eps")
933
934 x.ticks = 1:9
935 x.title = "Rebalancing strategy"
936 niceplot(x.ticks , lossOfUtility.pre.03.mean*transformation , xLabels=strategyNames ,
    xTitle=x.title , yTitle=y.title , y.addCustom=.2)
937 abline(v=x.ticks , lty=3)
938 legendText = expression(paste("(c)",lambda*=".03"))
939 nicelegend("left",legendText,horiz=T,bty="n",bg="white",cex=.7)
940 savePlot("images/rebStrategy_v_lossOfUtility_transCost_03",type="eps")
941
942 y.title = "Sharpe ratio"
943 niceplot(x.ticks , sharpeRatio.pre.03.mean , xLabels=strategyNames , xTitle=x.title ,
    yTitle=y.title)
944 abline(v=x.ticks , lty=3)
945 nicelegend("topleft",legendText,bty="n",bg="white",cex=.7)
946 savePlot("images/rebStrategy_v_sharpeRatio_transCost_03",type="eps")
947
948 # Simulated, subsequent transaction costs
949 terminalWealth.sub.03 = matrix(NA,nSims,n.entries)
950 sdWealth.sub.03 = matrix(NA,nSims,n.entries)
951 sdLogReturn.sub.03 = matrix(NA,nSims,n.entries)
952 totalTransCost.sub.03 = matrix(NA,nSims,n.entries)
953 for (k in 1:n.entries) {
954     terminalWealth.sub.03[k] = rebStrategy.transCost.03[[c(k,4)]]$simWealth.
        terminal
955     sdWealth.sub.03[k] = rebStrategy.transCost.03[[c(k,4)]]$simWealth.sd
956     sdLogReturn.sub.03[k] = rebStrategy.transCost.03[[c(k,4)]]$simWealth.
        logReturn.sd
957     totalTransCost.sub.03[k] = rebStrategy.transCost.03[[c(k,4)]]$totalTransCost
958 }
959 colnames(terminalWealth.sub.03) = strategyNames
960 terminalWealth.sub.03.mean = colMeans(terminalWealth.sub.03)

```

```

961 terminalWealth.sub.03.mean.sel4 = terminalWealth.sub.03.mean[sel4]
962 sdWealth.sub.03.mean = colMeans(sdWealth.sub.03)
963 sdWealth.sub.03.mean.sel4 = sdWealth.sub.03.mean[sel4]
964 terminalWealth.sub.03.sd = colSds(terminalWealth.sub.03)
965 terminalWealth.sub.03.sd.sel4 = terminalWealth.sub.03.sd[sel4]
966 lossOfWealth.sub.03 = terminalWealth.th.03 - terminalWealth.sub.03
967 lossOfWealth.sub.03.mean = colMeans(lossOfWealth.sub.03)
968 lossOfWealth.sub.03.mean.sel4 = lossOfWealth.sub.03.mean[sel4]
969 terminalUtility.sub.03 = utility(terminalWealth.sub.03,riskAversion)
970 terminalUtility.sub.03.mean = colMeans(terminalUtility.sub.03)
971 terminalUtility.sub.03.mean.sel4 = terminalUtility.sub.03.mean[sel4]
972 lossOfUtility.sub.03 = terminalUtility.th.03 - terminalUtility.sub.03
973 lossOfUtility.sub.03.sel4 = lossOfUtility.sub.03[,sel4]
974 lossOfUtility.sub.03.mean = colMeans(lossOfUtility.sub.03)
975 lossOfUtility.sub.03.mean.sel4 = lossOfUtility.sub.03.mean[sel4]
976 lossOfUtility.sub.03.sd = colSds(lossOfUtility.sub.03)
977 lossOfUtility.sub.03.cumMean = apply(lossOfUtility.sub.03,2,cumMean)
978 lossOfUtility.sub.03.cumMean.sel4 = lossOfUtility.sub.03.cumMean[,sel4]
979 lossOfUtility.sub.03.cumSd = apply(lossOfUtility.sub.03,2,cumSd)
980 lossOfUtility.sub.03.cumSd.sel4 = lossOfUtility.sub.03.cumSd[,sel4]
981 lossOfUtility.sub.03.sdCumMean = apply(lossOfUtility.sub.03.cumSd,2,function(x)
    {x/sqrt(nn)})
982 lossOfUtility.sub.03.sdCumMean.sel4 = lossOfUtility.sub.03.sdCumMean[,sel4]
983 lossOfUtility.sub.03.cumMean.lowerCL = lossOfUtility.sub.03.cumMean - qAlpha.
    half * lossOfUtility.sub.03.sdCumMean
984 lossOfUtility.sub.03.cumMean.upperCL = lossOfUtility.sub.03.cumMean + qAlpha.
    half * lossOfUtility.sub.03.sdCumMean
985 lossOfUtility.sub.03.cumMean.lowerCL.sel4 = lossOfUtility.sub.03.cumMean.lowerCL
    [,sel4]
986 lossOfUtility.sub.03.cumMean.upperCL.sel4 = lossOfUtility.sub.03.cumMean.upperCL
    [,sel4]
987 terminalLogReturn.sub.03 = log(terminalWealth.sub.03)
988 terminalLogReturn.sub.03.mean = colMeans(terminalLogReturn.sub.03)
989 terminalLogReturn.sub.03.mean.sel4 = terminalLogReturn.sub.03.mean[sel4]
990 sdLogReturn.sub.03.mean = colMeans(sdLogReturn.sub.03)
991 sdLogReturn.sub.03.mean.sel4 = sdLogReturn.sub.03.mean[sel4]
992 annualizedSdLogReturn.sub.03 = sdLogReturn.sub.03 * sqrt(nTimePoints)
993 annualizedSdLogReturn.sub.03.mean = colMeans(annualizedSdLogReturn.sub.03)
994 terminalLogReturn.sub.03.sd = colSds(terminalLogReturn.sub.03)
995 terminalLogReturn.sub.03.sd.sel4 = terminalLogReturn.sub.03.sd[sel4]
996 excessReturn.sub.03 = terminalLogReturn.sub.03 - rent
997 sharpeRatio.sub.03 = excessReturn.sub.03 / (sqrt(nTimePoints)*sdLogReturn.sub
    .03)
998 sharpeRatio.sub.03.mean = colMeans(sharpeRatio.sub.03)
999 sharpeRatio.sub.03.mean.sel4 = sharpeRatio.sub.03.mean[sel4]
1000 volOfVol.sub.03 = colSds(annualizedSdLogReturn.sub.03)
1001 correlation.sub.03 = colCorrs(terminalLogReturn.sub.03,annualizedSdLogReturn.sub
    .03)
1002 totalTransCost.sub.03.mean = colMeans(totalTransCost.sub.03)
1003 totalTransCost.sub.03.mean.sel4 = totalTransCost.sub.03.mean[sel4]
1004
1005 # Plotting
1006 y.rangeDiff.sub.03.sel4 = colRange(lossOfUtility.sub.03.cumMean.sel4)[2,] -
    colRange(lossOfUtility.sub.03.cumMean.sel4)[1,]
1007 y.lim.sub.03.sel4 = rbind(lossOfUtility.sub.03.mean.sel4 - y.rangeDiff.sub.03.
    sel4/25,lossOfUtility.sub.03.mean.sel4 + y.rangeDiff.sub.03.sel4/25)
1008
1009 transformation = 1e2
1010 y.title = expression(paste("Mean loss of utility",phantom(0) %*% 10^2))
1011 niceplot(lossOfUtility.sub.03.cumMean.sel4[,1]*transformation,xLabels=x.labels,
    yTitle=y.title,figsPerPage=4,y.addCustom=.2,nCol=2,horizLines=T,downsample=T
    ,ylim=y.lim.sub.03.sel4[,1]*transformation)
1012 nicelines(lossOfUtility.sub.03.cumMean.lowerCL.sel4[,1]*transformation,
    downsample=T,col="darkgray",lty=3)
1013 nicelines(lossOfUtility.sub.03.cumMean.upperCL.sel4[,1]*transformation,

```

```

    downsamples=T, col="darkgray", lty=3)
1014 legendText = c(expression(paste("e)", lambda*=".03")), "Transaction costs
    strategy : Subsequent", "Rebalancing strategy : Hourly"))
1015 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
1016
1017 niceplot(lossOfUtility.sub.03.cumMean.sel4[,2]*transformation, xLabels=x.labels,
    yTitle=y.title, figsPerPage=4, y.addCustom=.2, multiPlot=T, newDev=F, horizLines=
    T, downsamples=T, ylim=y.lim.sub.03.sel4[,2]*transformation)
1018 nicelines(lossOfUtility.sub.03.cumMean.lowerCL.sel4[,2]*transformation,
    downsamples=T, col="darkgray", lty=3)
1019 nicelines(lossOfUtility.sub.03.cumMean.upperCL.sel4[,2]*transformation,
    downsamples=T, col="darkgray", lty=3)
1020 legendText = c(expression(paste("f)", lambda*=".03")), "Transaction costs
    strategy : Subsequent", "Rebalancing strategy : Daily"))
1021 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
1022 savePlot("images/lossOfUtility_03_sub_Hourly_Daily", type="eps")
1023
1024 niceplot(lossOfUtility.sub.03.cumMean.sel4[,3]*transformation, xLabels=x.labels,
    yTitle=y.title, figsPerPage=4, y.addCustom=.2, nCol=2, horizLines=T, downsamples=T,
    ylim=y.lim.sub.03.sel4[,3]*transformation)
1025 nicelines(lossOfUtility.sub.03.cumMean.lowerCL.sel4[,3]*transformation,
    downsamples=T, col="darkgray", lty=3)
1026 nicelines(lossOfUtility.sub.03.cumMean.upperCL.sel4[,3]*transformation,
    downsamples=T, col="darkgray", lty=3)
1027 legendText = c(expression(paste("g)", lambda*=".03")), "Transaction costs
    strategy : Subsequent", "Rebalancing strategy : Monthly"))
1028 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
1029
1030 niceplot(lossOfUtility.sub.03.cumMean.sel4[,4]*transformation, xLabels=x.labels,
    yTitle=y.title, figsPerPage=4, y.addCustom=.2, multiPlot=T, newDev=F, horizLines=
    T, downsamples=T, ylim=y.lim.sub.03.sel4[,4]*transformation)
1031 nicelines(lossOfUtility.sub.03.cumMean.lowerCL.sel4[,4]*transformation,
    downsamples=T, col="darkgray", lty=3)
1032 nicelines(lossOfUtility.sub.03.cumMean.upperCL.sel4[,4]*transformation,
    downsamples=T, col="darkgray", lty=3)
1033 legendText = c(expression(paste("h)", lambda*=".03")), "Transaction costs
    strategy : Subsequent", "Rebalancing strategy : Annually"))
1034 legendObject = nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
1035 savePlot("images/lossOfUtility_03_sub_Monthly_Annually", type="eps")
1036
1037 # Setting up summarizing tables
1038 tab1 = matrix(NA, 36, 5)
1039 for (k in 1:9) {
1040   tab1[k*4-3,] = c(terminalWealth.th.03.mean[k], 0, terminalUtility.th.03.mean[k],
    0, 0)
1041   tab1[k*4-2,] = c(terminalWealth.none.03.mean[k], 0, terminalUtility.none.03.mean
    [k], lossOfUtility.none.03.mean[k], lossOfUtility.none.03.sd[k])
1042   tab1[k*4-1,] = c(terminalWealth.pre.03.mean[k], totalTransCost.pre.03.mean[k],
    terminalUtility.pre.03.mean[k], lossOfUtility.pre.03.mean[k], lossOfUtility.
    pre.03.sd[k])
1043   tab1[k*4,] = c(terminalWealth.sub.03.mean[k], totalTransCost.sub.03.mean[k],
    terminalUtility.sub.03.mean[k], lossOfUtility.sub.03.mean[k], lossOfUtility.
    sub.03.sd[k])
1044 }
1045
1046 tab1[,2] = tab1[,2] * 1e2
1047 tab1[,4] = tab1[,4] * 1e2
1048 tab1[,5] = tab1[,5] * 1e3
1049
1050 tab1 = round(tab1, 4)
1051
1052 for (k in 1:36) {
1053   tab1[k,2] = paste(tab1[k,2], "\\e{\\text{-2}}", sep="")
1054   tab1[k,4] = paste(tab1[k,4], "\\e{\\text{-2}}", sep="")
1055   tab1[k,5] = paste(tab1[k,5], "\\e{\\text{-3}}", sep="")

```

```

1056 }
1057
1058 printex(tab1)
1059
1060 tab2 = matrix(NA,36,5)
1061 for (k in 1:9) {
1062   tab2[k*4-3,] = c(terminalLogReturn.th.03.mean[k], annualizedSdLogReturn.th.03.
1063     mean[k], sharpeRatio.th.03.mean[k], volOfVol.th.03[k], correlation.th.03[k])
1064   tab2[k*4-2,] = c(terminalLogReturn.none.03.mean[k], annualizedSdLogReturn.none
1065     .03.mean[k], sharpeRatio.none.03.mean[k], volOfVol.none.03[k], correlation.
1066     none.03[k])
1067   tab2[k*4-1,] = c(terminalLogReturn.pre.03.mean[k], annualizedSdLogReturn.pre
1068     .03.mean[k], sharpeRatio.pre.03.mean[k], volOfVol.pre.03[k], correlation.pre
1069     .03[k])
1070   tab2[k*4,] = c(terminalLogReturn.sub.03.mean[k], annualizedSdLogReturn.sub
1071     .03.mean[k], sharpeRatio.sub.03.mean[k], volOfVol.sub.03[k], correlation.sub
1072     .03[k])
1073 }
1074
1075 tab2[,1] = tab2[,1] * 1e2
1076 tab2[,4] = tab2[,4] * 1e3
1077
1078 tab2 = round(tab2,4)
1079
1080 for (k in 1:36) {
1081   tab2[k,1] = paste(tab2[k,1], "\\ e{\\text -2}", sep="")
1082   tab2[k,4] = paste(tab2[k,4], "\\ e{\\text -3}", sep="")
1083 }
1084
1085 printex(tab2)
1086
1087 #
1088 # Analysis of distributions of total transaction costs, lambda = .01
1089 #
1090
1091 x.title = "Total transaction cost"
1092 y.title = "Frequency"
1093 breaksLength = 70
1094
1095 # Hourly rebalancings
1096 dataSet = totalTransCost.pre.01[,1]
1097 print(range(dataSet))
1098 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1099 histObject = hist(dataSet,breaks=res,plot=F)
1100 y.lim = range(histObject$counts) * 1.3
1101 nicehist(dataSet,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
1102 legendText = c(expression(paste("(a)",lambda*=".01")), "T.c. strategy : Preceding
1103   ", "Reb. strategy : Hourly"))
1104 nicelegend("topleft",legendText,bty="n",cex=.7)
1105
1106 # Daily rebalancings
1107 dataSet = totalTransCost.pre.01[,3]
1108 print(range(dataSet))
1109 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1110 histObject = hist(dataSet,breaks=res,plot=F)
1111 y.lim = range(histObject$counts) * 1.3
1112 nicehist(dataSet,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,ylim=y.lim,
1113   breaks=res)
1114 legendText = c(expression(paste("(b)",lambda*=".01")), "T.c. strategy : Preceding
1115   ", "Reb. strategy : Daily"))
1116 nicelegend("topleft",legendText,bty="n",cex=.7)
1117
1118 # Saving dual-plot
1119 savePlot("images/hist_transCost_HourlyDaily",type="eps")
1120

```



```

1111 # Every 3rd day rebalancings
1112 dataSet = totalTransCost.pre.01[,4]
1113 print(range(dataSet))
1114 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1115 histObject = hist(dataSet,breaks=res,plot=F)
1116 y.lim = range(histObject$counts) * 1.3
1117 nicehist(dataSet,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
1118 legendText = c(expression(paste("(c)",lambda*=".01")), "T.c. strategy : Preceding
      ", "Reb. strategy : Ev. 3rd day"))
1119 nicelegend("topleft",legendText,bty="n",cex=.7)
1120
1121 # Every 12th day rebalancings
1122 dataSet = totalTransCost.pre.01[,5]
1123 print(range(dataSet))
1124 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1125 histObject = hist(dataSet,breaks=res,plot=F)
1126 y.lim = range(histObject$counts) * 1.3
1127 nicehist(dataSet,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,ylim=y.lim,
      breaks=res)
1128 legendText = c(expression(paste("(d)",lambda*=".01")), "T.c. strategy : Preceding
      ", "Reb. strategy : Ev. 12th day"))
1129 nicelegend("topleft",legendText,bty="n",cex=.7)
1130
1131 # Saving dual-plot
1132 savePlot("images/hist_transCost_3rd12th",type="eps")
1133
1134 # Hourly rebalancings
1135 dataSet = totalTransCost.pre.01[,6]
1136 print(range(dataSet))
1137 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1138 histObject = hist(dataSet,breaks=res,plot=F)
1139 y.lim = range(histObject$counts) * 1.3
1140 nicehist(dataSet,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
1141 legendText = c(expression(paste("(e)",lambda*=".01")), "T.c. strategy : Preceding
      ", "Reb. strategy : Monthly"))
1142 nicelegend("topleft",legendText,bty="n",cex=.7)
1143
1144 # Daily rebalancings
1145 dataSet = totalTransCost.pre.01[,7]
1146 print(range(dataSet))
1147 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1148 histObject = hist(dataSet,breaks=res,plot=F)
1149 y.lim = range(histObject$counts) * 1.3
1150 nicehist(dataSet,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,ylim=y.lim,
      breaks=res)
1151 legendText = c(expression(paste("(f)",lambda*=".01")), "T.c. strategy : Preceding
      ", "Reb. strategy : Bimonthly"))
1152 nicelegend("topleft",legendText,bty="n",cex=.7)
1153
1154 # Saving dual-plot
1155 savePlot("images/hist_transCost_MonthlyBi",type="eps")
1156
1157 # Hourly rebalancings
1158 dataSet = totalTransCost.pre.01[,8]
1159 print(range(dataSet))
1160 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1161 histObject = hist(dataSet,breaks=res,plot=F)
1162 y.lim = range(histObject$counts) * 1.3
1163 nicehist(dataSet,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
1164 legendText = c(expression(paste("(g)",lambda*=".01")), "T.c. strategy : Preceding
      ", "Reb. strategy : Semiannually"))
1165 nicelegend("topleft",legendText,bty="n",cex=.7)
1166
1167 # Daily rebalancings
1168 dataSet = totalTransCost.pre.01[,9]

```

```

1169 print(range(dataSet))
1170 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1171 histObject = hist(dataSet,breaks=res,plot=F)
1172 y.lim = range(histObject$counts) * 1.3
1173 nicehist(dataSet,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,ylim=y.lim,
           breaks=res)
1174 legendText = c(expression(paste("(h)",lambda*=".01")), "T.c. strategy : Preceding
           ", "Reb. strategy : Annually"))
1175 nicelegend("topleft",legendText,bty="n",cex=.7)
1176
1177 # Saving dual-plot
1178 savePlot("images/hist_transCost_SemiAnnually",type="eps")
1179
1180 #
1181 # Analysis of distributions of total transaction costs, lambda = .02
1182 #
1183
1184 x.title = "Total transaction cost"
1185 y.title = "Frequency"
1186 breaksLength = 70
1187
1188 # Hourly rebalancings
1189 dataSet = totalTransCost.pre.02[,1]
1190 print(range(dataSet))
1191 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1192 histObject = hist(dataSet,breaks=res,plot=F)
1193 y.lim = range(histObject$counts) * 1.3
1194 nicehist(dataSet,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
1195 legendText = c(expression(paste("(a)",lambda*=".02")), "T.c. strategy : Preceding
           ", "Reb. strategy : Hourly"))
1196 nicelegend("topleft",legendText,bty="n",cex=.7)
1197
1198 # Daily rebalancings
1199 dataSet = totalTransCost.pre.02[,3]
1200 print(range(dataSet))
1201 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1202 histObject = hist(dataSet,breaks=res,plot=F)
1203 y.lim = range(histObject$counts) * 1.3
1204 nicehist(dataSet,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,ylim=y.lim,
           breaks=res)
1205 legendText = c(expression(paste("(b)",lambda*=".02")), "T.c. strategy : Preceding
           ", "Reb. strategy : Daily"))
1206 nicelegend("topleft",legendText,bty="n",cex=.7)
1207
1208 # Saving dual-plot
1209 savePlot("images/hist_transCost02_HourlyDaily",type="eps")
1210
1211 # Every 3rd day rebalancings
1212 dataSet = totalTransCost.pre.02[,4]
1213 print(range(dataSet))
1214 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1215 histObject = hist(dataSet,breaks=res,plot=F)
1216 y.lim = range(histObject$counts) * 1.3
1217 nicehist(dataSet,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
1218 legendText = c(expression(paste("(c)",lambda*=".02")), "T.c. strategy : Preceding
           ", "Reb. strategy : Ev. 3rd day"))
1219 nicelegend("topleft",legendText,bty="n",cex=.7)
1220
1221 # Every 12th day rebalancings
1222 dataSet = totalTransCost.pre.02[,5]
1223 print(range(dataSet))
1224 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1225 histObject = hist(dataSet,breaks=res,plot=F)
1226 y.lim = range(histObject$counts) * 1.3
1227 nicehist(dataSet,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,ylim=y.lim,

```

```

        breaks=res)
1228 legendText = c(expression(paste("(d) ",lambda*=".02")), "T.c. strategy : Preceding
        ", "Reb. strategy : Ev. 12th day"))
1229 nicelegend(" topleft", legendText, bty="n", cex=.7)
1230
1231 # Saving dual-plot
1232 savePlot(" images/hist_transCost02_3rd12th", type="eps")
1233
1234 # Hourly rebalancings
1235 dataSet = totalTransCost.pre.02[,6]
1236 print(range(dataSet))
1237 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1238 histObject = hist(dataSet,breaks=res,plot=F)
1239 y.lim = range(histObject$counts) * 1.3
1240 nicehist(dataSet,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
1241 legendText = c(expression(paste("(e) ",lambda*=".02")), "T.c. strategy : Preceding
        ", "Reb. strategy : Monthly"))
1242 nicelegend(" topleft", legendText, bty="n", cex=.7)
1243
1244 # Daily rebalancings
1245 dataSet = totalTransCost.pre.02[,7]
1246 print(range(dataSet))
1247 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1248 histObject = hist(dataSet,breaks=res,plot=F)
1249 y.lim = range(histObject$counts) * 1.3
1250 nicehist(dataSet,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,ylim=y.lim,
        breaks=res)
1251 legendText = c(expression(paste("(f) ",lambda*=".02")), "T.c. strategy : Preceding
        ", "Reb. strategy : Bimonthly"))
1252 nicelegend(" topleft", legendText, bty="n", cex=.7)
1253
1254 # Saving dual-plot
1255 savePlot(" images/hist_transCost02_MonthlyBi", type="eps")
1256
1257 # Hourly rebalancings
1258 dataSet = totalTransCost.pre.02[,8]
1259 print(range(dataSet))
1260 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1261 histObject = hist(dataSet,breaks=res,plot=F)
1262 y.lim = range(histObject$counts) * 1.3
1263 nicehist(dataSet,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
1264 legendText = c(expression(paste("(g) ",lambda*=".02")), "T.c. strategy : Preceding
        ", "Reb. strategy : Semiannually"))
1265 nicelegend(" topleft", legendText, bty="n", cex=.7)
1266
1267 # Daily rebalancings
1268 dataSet = totalTransCost.pre.02[,9]
1269 print(range(dataSet))
1270 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1271 histObject = hist(dataSet,breaks=res,plot=F)
1272 y.lim = range(histObject$counts) * 1.3
1273 nicehist(dataSet,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,ylim=y.lim,
        breaks=res)
1274 legendText = c(expression(paste("(h) ",lambda*=".02")), "T.c. strategy : Preceding
        ", "Reb. strategy : Annually"))
1275 nicelegend(" topleft", legendText, bty="n", cex=.7)
1276
1277 # Saving dual-plot
1278 savePlot(" images/hist_transCost02_SemiAnnually", type="eps")
1279
1280 #
1281 # Analysis of distributions of total transaction costs, lambda = .03
1282 #
1283
1284 x.title = "Total transaction cost"

```

```

1285 y.title = "Frequency"
1286 breaksLength = 70
1287
1288 # Hourly rebalancings
1289 dataSet = totalTransCost.pre.03[,1]
1290 print(range(dataSet))
1291 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1292 histObject = hist(dataSet,breaks=res,plot=F)
1293 y.lim = range(histObject$counts) * 1.3
1294 nicehist(dataSet,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
1295 legendText = c(expression(paste("(a)",lambda*=".03")),
1296   "T.c. strategy : Preceding",
1297   "Reb. strategy : Hourly"))
1298 nicelegend("topleft",legendText,bty="n",cex=.7)
1299
1300 # Daily rebalancings
1301 dataSet = totalTransCost.pre.03[,3]
1302 print(range(dataSet))
1303 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1304 histObject = hist(dataSet,breaks=res,plot=F)
1305 y.lim = range(histObject$counts) * 1.3
1306 nicehist(dataSet,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,ylim=y.lim,
1307   breaks=res)
1308 legendText = c(expression(paste("(b)",lambda*=".03")),
1309   "T.c. strategy : Preceding",
1310   "Reb. strategy : Daily"))
1311 nicelegend("topleft",legendText,bty="n",cex=.7)
1312
1313 # Saving dual-plot
1314 savePlot("images/hist_transCost03-HourlyDaily",type="eps")
1315
1316 # Every 3rd day rebalancings
1317 dataSet = totalTransCost.pre.03[,4]
1318 print(range(dataSet))
1319 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1320 histObject = hist(dataSet,breaks=res,plot=F)
1321 y.lim = range(histObject$counts) * 1.3
1322 nicehist(dataSet,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
1323 legendText = c(expression(paste("(c)",lambda*=".03")),
1324   "T.c. strategy : Preceding",
1325   "Reb. strategy : Ev. 3rd day"))
1326 nicelegend("topleft",legendText,bty="n",cex=.7)
1327
1328 # Every 12th day rebalancings
1329 dataSet = totalTransCost.pre.03[,5]
1330 print(range(dataSet))
1331 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1332 histObject = hist(dataSet,breaks=res,plot=F)
1333 y.lim = range(histObject$counts) * 1.3
1334 nicehist(dataSet,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,ylim=y.lim,
1335   breaks=res)
1336 legendText = c(expression(paste("(d)",lambda*=".03")),
1337   "T.c. strategy : Preceding",
1338   "Reb. strategy : Ev. 12th day"))
1339 nicelegend("topleft",legendText,bty="n",cex=.7)
1340
1341 # Saving dual-plot
1342 savePlot("images/hist_transCost03-3rd12th",type="eps")
1343
1344 # Hourly rebalancings
1345 dataSet = totalTransCost.pre.03[,6]
1346 print(range(dataSet))
1347 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1348 histObject = hist(dataSet,breaks=res,plot=F)
1349 y.lim = range(histObject$counts) * 1.3
1350 nicehist(dataSet,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
1351 legendText = c(expression(paste("(e)",lambda*=".03")),
1352   "T.c. strategy : Preceding",
1353   "Reb. strategy : Monthly"))
1354 nicelegend("topleft",legendText,bty="n",cex=.7)

```

```

1343
1344 # Daily rebalancings
1345 dataSet = totalTransCost.pre.03[,7]
1346 print(range(dataSet))
1347 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1348 histObject = hist(dataSet,breaks=res,plot=F)
1349 y.lim = range(histObject$counts) * 1.3
1350 nicehist(dataSet,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,ylim=y.lim,
           breaks=res)
1351 legendText = c(expression(paste("(f)",lambda*=".03")),
                  "T.c. strategy : Preceding",
                  "Reb. strategy : Bimonthly"))
1352 nicelegend("topleft",legendText,bty="n",cex=.7)
1353
1354 # Saving dual-plot
1355 savePlot("images/hist_transCost03_MonthlyBi",type="eps")
1356
1357 # Hourly rebalancings
1358 dataSet = totalTransCost.pre.03[,8]
1359 print(range(dataSet))
1360 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1361 histObject = hist(dataSet,breaks=res,plot=F)
1362 y.lim = range(histObject$counts) * 1.3
1363 nicehist(dataSet,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
1364 legendText = c(expression(paste("(g)",lambda*=".03")),
                  "T.c. strategy : Preceding",
                  "Reb. strategy : Semiannually"))
1365 nicelegend("topleft",legendText,bty="n",cex=.7)
1366
1367 # Daily rebalancings
1368 dataSet = totalTransCost.pre.03[,9]
1369 print(range(dataSet))
1370 res = seq(min(dataSet),max(dataSet),length=breaksLength)
1371 histObject = hist(dataSet,breaks=res,plot=F)
1372 y.lim = range(histObject$counts) * 1.3
1373 nicehist(dataSet,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,ylim=y.lim,
           breaks=res)
1374 legendText = c(expression(paste("(h)",lambda*=".03")),
                  "T.c. strategy : Preceding",
                  "Reb. strategy : Annually"))
1375 nicelegend("topleft",legendText,bty="n",cex=.7)
1376
1377 # Saving dual-plot
1378 savePlot("images/hist_transCost03_SemiAnnually",type="eps")

```

## B.6 Simulation model IV

### B.6.1 Simulation machinery

```

1 ##
2 # Master Thesis
3 # Simulation model IV
4 # Simulation algorithm
5 #
6
7 require(mnormt)
8
9 simPortfolio.stochVol = function(nSims,paramSet,dualBrownianFileName=NULL) {
10 #
11 # Simulates nSims portfolios following the 14 parameter values of paramSet
12 # and returns terminal utilities of theoretical and simulated wealth and

```

```

13 # the loss of utility. Includes transaction costs and stochastic
14 # volatility!
15 #
16 require(mnormt)
17
18 logReturn = function(x) {
19   #
20   # Computes the log returns of a time series x.
21   #
22   n = length(x)
23   xUp = x[2:n]
24   xLow = x[1:(n-1)]
25   logReturns = log(xUp/xLow)
26   return(logReturns)
27 }
28
29 riskAversion = function(drift , volatility , rent , VaR, delta , alpha) {
30   #
31   # Computes the risk aversion parameter of a power-type utility function
32   # through Value at Risk.
33   #
34   qAlpha = qnorm(alpha)
35   solution = 1:2*NA
36   a = drift - rent + qAlpha*volatility/sqrt(delta)
37   b = 2*volatility^2*(VaR/delta+rent)
38   solution[1] = 1 + (drift-rent)*(a+sqrt(a^2+b))/b
39   solution[2] = 1 + (drift-rent)*(a-sqrt(a^2+b))/b
40   return(solution)
41 }
42
43 optimalControl = function(drift , volatility , rent , riskAversion) {
44   #
45   # Computes the optimal control following a power-type utility function.
46   #
47   control = pmax(pmin((drift-rent)/((1-riskAversion)*volatility^2),1),0)
48   return(control)
49 }
50
51 dualBrownianIncrements = function(n,delta , correlation) {
52   #
53   # Simulates random series of n brownian increments with variance delta.
54   #
55   varcov = matrix(c(1,correlation , correlation ,1)*delta ,2,2)
56   meanVector = c(0,0)
57   return(rmnorm(n, meanVector , varcov))
58 }
59
60 #
61 # Assigning variables.
62 #
63 varNames = c("initWealth" ,"nTradingDays" ,"nDailyIncrements" ,"nDailyRebs" ,"
64             "drift" ,"rent" ,"aversion" ,"costProp" ,"var.init" ,"reversionRate" ,"var.long
65             " ,"volOfVol" ,"correlation")
66 nParams = length(paramSet)
67 nParams.required = length(varNames)
68 if (nParams != nParams.required) stop(paste("Number of input parameters equals
69           ",nParams,". Must equal ",nParams.required ,sep=""))
70 for (j in 1:nParams.required) { assign(varNames[j],paramSet[j]) }
71
72 #
73 # Initializing the simulation structure.
74 #
75 simIndex = 1:nSims
76 nTimePoints = nTradingDays * nDailyIncrements

```

```

75 | lastIndex = nTimePoints
76 | delta = 1 / nTimePoints
77 | timePoints = seq(delta,1,delta)
78 | nRebDelay = nDailyIncrements / nDailyRebs
79 | rebIndex = seq(nRebDelay,nTimePoints,nRebDelay)
80 | days = seq(delta*nTradingDays,nTradingDays,delta*nTradingDays)
81 | rebDays = days[rebIndex]
82 | ones = rep(1,nRebDelay)
83 |
84 | # Start of simulation time
85 | timeStart = proc.time()[3][[1]]
86 |
87 | # Initializing simulation vectors
88 | simWealth.none = NA
89 | simWealth.transCost = NA
90 |
91 | #
92 | # Using full simulation scheme if nSims = 1
93 | #
94 |
95 | if (nSims == 1) {
96 |
97 |   # Intializing other statistics
98 |   return.risky = 1:nTimePoints * NA
99 |   return.riskfree = 1:nTimePoints * NA
100 |
101 |   # Intializing simulated wealth without transaction costs
102 |   simWealth = NA
103 |   simWealth.risky = NA
104 |   simWealth.riskfree = NA
105 |   transQuantity = 1:nTimePoints * 0
106 |   propInRisky = NA
107 |   propInRiskfree = NA
108 |
109 |   # Intializing simulated wealth with preceding transaction costs
110 |   simWealth.tc = NA
111 |   simWealth.tc.risky = NA
112 |   simWealth.tc.riskfree = NA
113 |   transQuantity.tc = 1:nTimePoints * 0
114 |   transCost.tc = 1:nTimePoints * 0
115 |   propInRisky.tc = NA
116 |   propInRiskfree.tc = NA
117 |
118 |   # Generation of Brownian motions
119 |   if (!is.null(dualBrownianFileName) && file.exists(dualBrownianFileName, sep
120 |     =")) { cat("Loading brownian increments...\n"); load(
121 |     dualBrownianFileName) }
122 |   else { dualInc = dualBrownianIncrements(nTimePoints,delta,correlation) }
123 |   if (!is.null(dualBrownianFileName) && !file.exists(dualBrownianFileName)) {
124 |     cat("Saving brownian increments...\n"); save(dualInc, file=
125 |     dualBrownianFileName) }
126 |   inc.risky = dualInc[,1]
127 |   inc.var = dualInc[,2]
128 |   dualBM = colCumsums(dualInc)
129 |   BM.risky = dualBM[,1]
130 |   BM.vol = dualBM[,2]
131 |
132 |   #
133 |   # First part of the simulations
134 |   #
135 |
136 |   # Simulation of the stochastic volatility
137 |   stochVar = 1:nTimePoints * NA
138 |   stochVar[1] = var.init + reversionRate*(var.long-var.init)*delta + volOfVol*
139 |     sqrt(var.init)*inc.var[1]

```

```

135   for (k in 2:nTimePoints) { stochVar[k] = stochVar[k-1] + reversionRate*(var.
136     long-stochVar[k-1])*delta + volOfVol*sqrt(stochVar[k-1])*inc.var[k] }
137   stochVol = sqrt(stochVar)
138
139   # Calculation of optimal strategy
140   u.star.init = optimalControl(drift,sqrt(var.init),rent,aversion)
141   u.star = optimalControl(drift,stochVol,rent,aversion)
142
143   # Time points to be simulated (active time points)
144   activeIndices = 1:nRebDelay
145   rebPoint = tail(activeIndices,1)
146
147   # Determining active variables
148   inc.risky.active = inc.risky[activeIndices]
149   stochVol.active = stochVol[activeIndices]
150   u.star.rebPoint = u.star[rebPoint]
151   return.risky[activeIndices] = cumprod(1+drift*delta+stochVol.active*inc.
152     risky.active) - 1
153   return.riskfree[activeIndices] = cumprod((1+rent*delta)*ones) - 1
154
155   # Without transaction costs
156   simWealth.risky[activeIndices] = u.star.init * initWealth * cumprod(1 +
157     drift*delta + stochVol.active*inc.risky.active)
158   simWealth.riskfree[activeIndices] = (1-u.star.init) * initWealth * cumprod
159     ((1+rent*delta)*ones)
160   simWealth[activeIndices] = simWealth.risky[activeIndices] + simWealth.
161     riskfree[activeIndices]
162   simWealth.risky.prime = simWealth.risky[rebPoint]
163   simWealth.riskfree.prime = simWealth.riskfree[rebPoint]
164   transQuantity[rebPoint] = (1-u.star.rebPoint)*simWealth.risky.prime - u.star
165     .rebPoint*simWealth.riskfree.prime
166   simWealth.risky[rebPoint] = simWealth.risky.prime - transQuantity[rebPoint]
167   simWealth.riskfree[rebPoint] = simWealth.riskfree.prime + transQuantity[
168     rebPoint]
169   simWealth[rebPoint] = simWealth.risky[rebPoint] + simWealth.riskfree[
170     rebPoint]
171   propInRisky[activeIndices] = simWealth.risky[activeIndices] / simWealth[
172     activeIndices]
173   propInRiskfree[activeIndices] = simWealth.riskfree[activeIndices] /
174     simWealth[activeIndices]
175
176   # With transaction costs (preceding)
177   simWealth.tc.risky[activeIndices] = u.star.init * initWealth * cumprod(1 +
178     drift*delta + stochVol.active*inc.risky.active)
179   simWealth.tc.riskfree[activeIndices] = (1-u.star.init) * initWealth *
180     cumprod((1+rent*delta)*ones)
181   simWealth.tc[activeIndices] = simWealth.tc.risky[activeIndices] + simWealth.
182     tc.riskfree[activeIndices]
183   simWealth.tc.risky.prime = simWealth.tc.risky[rebPoint]
184   simWealth.tc.riskfree.prime = simWealth.tc.riskfree[rebPoint]
185   signDiffReturn = sign((1-u.star.rebPoint)*u.star.init*prod(1+drift*delta+
186     stochVol.active*inc.risky.active) - u.star.rebPoint*(1-u.star.init)*prod
187     ((1+rent*delta)*ones))
188   transQuantity.tc[rebPoint] = ((1-u.star.rebPoint)*simWealth.tc.risky.prime -
189     u.star.rebPoint*simWealth.tc.riskfree.prime) / (1 - signDiffReturn*
190     costProp*u.star.rebPoint)
191   transCost.tc[rebPoint] = abs(costProp*transQuantity.tc[rebPoint])
192   simWealth.tc.risky[rebPoint] = simWealth.tc.risky.prime - transQuantity.tc[
193     rebPoint]
194   simWealth.tc.riskfree[rebPoint] = simWealth.tc.riskfree.prime +
195     transQuantity.tc[rebPoint] - transCost.tc[rebPoint]
196   simWealth.tc[rebPoint] = simWealth.tc.risky[rebPoint] + simWealth.tc.
197     riskfree[rebPoint]
198   propInRisky.tc[activeIndices] = simWealth.tc.risky[activeIndices] /
199     simWealth.tc[activeIndices]

```



```

179     propInRiskfree.tc[activeIndices] = simWealth.tc.riskfree[activeIndices] /
180         simWealth.tc[activeIndices]
181
182 # Storing last rebalancing time point rebalancing strategy
183 u.star.last = u.star.rebPoint
184
185 for (j in rebIndex[-length(rebIndex)] + 1) {
186     activeIndices = j:(j+nRebDelay-1)
187     rebPoint = tail(activeIndices,1)
188
189 # Determining active variables
190 inc.risky.active = inc.risky[activeIndices]
191 stochVol.active = stochVol[activeIndices]
192 u.star.rebPoint = u.star[rebPoint]
193 return.risky[activeIndices] = cumprod(1+drift*delta+stochVol.active*inc.
194     risky.active) - 1
195 return.riskfree[activeIndices] = cumprod((1+rent*delta)*ones) - 1
196
197 # Without transaction costs
198 simWealth.risky[activeIndices] = u.star.last * simWealth[j-1] * cumprod(1
199     + drift*delta + stochVol.active*inc.risky.active)
200 simWealth.riskfree[activeIndices] = (1-u.star.last) * simWealth[j-1] *
201     cumprod((1+rent*delta)*ones)
202 simWealth[activeIndices] = simWealth.risky[activeIndices] + simWealth.
203     riskfree[activeIndices]
204 simWealth.risky.prime = simWealth.risky[rebPoint]
205 simWealth.riskfree.prime = simWealth.riskfree[rebPoint]
206 transQuantity[rebPoint] = (1-u.star.rebPoint)*simWealth.risky.prime - u.
207     star.rebPoint*simWealth.riskfree.prime
208 simWealth.risky[rebPoint] = simWealth.risky.prime - transQuantity[rebPoint
209     ]
210 simWealth.riskfree[rebPoint] = simWealth.riskfree.prime + transQuantity[
211     rebPoint]
212 simWealth[rebPoint] = simWealth.risky[rebPoint] + simWealth.riskfree[
213     rebPoint]
214 propInRisky[activeIndices] = simWealth.risky[activeIndices] / simWealth[
215     activeIndices]
216 propInRiskfree[activeIndices] = simWealth.riskfree[activeIndices] /
217     simWealth[activeIndices]
218
219 # With transaction costs (preceding)
220 simWealth.tc.risky[activeIndices] = u.star.last * simWealth.tc[j-1] *
221     cumprod(1 + drift*delta + stochVol.active*inc.risky.active)
222 simWealth.tc.riskfree[activeIndices] = (1-u.star.last) * simWealth.tc[j-1]
223     * cumprod((1+rent*delta)*ones)
224 simWealth.tc[activeIndices] = simWealth.tc.risky[activeIndices] +
225     simWealth.tc.riskfree[activeIndices]
226 simWealth.tc.risky.prime = simWealth.tc.risky[rebPoint]
227 simWealth.tc.riskfree.prime = simWealth.tc.riskfree[rebPoint]
228 signDiffReturn = sign((1-u.star.rebPoint)*u.star.last*prod(1+drift*delta+
229     stochVol.active*inc.risky.active) - u.star.rebPoint*(1-u.star.last)*
230     prod((1+rent*delta)*ones))
231 transQuantity.tc[rebPoint] = ((1-u.star.rebPoint)*simWealth.tc.risky.prime
232     - u.star.rebPoint*simWealth.tc.riskfree.prime) / (1 - signDiffReturn*
233     costProp*u.star.rebPoint)
234 transCost.tc[rebPoint] = abs(costProp*transQuantity.tc[rebPoint])
235 simWealth.tc.risky[rebPoint] = simWealth.tc.risky.prime - transQuantity.tc
236     [rebPoint]
237 simWealth.tc.riskfree[rebPoint] = simWealth.tc.riskfree.prime +
238     transQuantity.tc[rebPoint] - transCost.tc[rebPoint]
239 simWealth.tc[rebPoint] = simWealth.tc.risky[rebPoint] + simWealth.tc.
240     riskfree[rebPoint]
241 propInRisky.tc[activeIndices] = simWealth.tc.risky[activeIndices] /
242     simWealth.tc[activeIndices]

```

```

221     propInRiskfree.tc[activeIndices] = simWealth.tc.riskfree[activeIndices] /
      simWealth.tc[activeIndices]
222
223     # Storing last rebalancing time point rebalancing strategy
224     u.star.last = u.star.rebPoint
225     }
226 }
227
228 #
229 # Using compact form of simulation scheme if nSims > 1
230 #
231
232 else {
233     print("nSims > 1...")
234
235     corrInc = simIndex * NA
236     stochVol.mean = simIndex * NA
237     stochVol.sd = simIndex * NA
238     u.star.mean = simIndex*NA
239
240     simWealth.sd = simIndex * NA
241     simWealth.terminal = simIndex * NA
242     simWealth.logReturn.sd = simIndex * NA
243
244     simWealth.tc.sd = simIndex * NA
245     simWealth.tc.terminal = simIndex * NA
246     simWealth.tc.logReturn.sd = simIndex * NA
247     totalTransCost = simIndex * 0
248
249     for (k in 1:nSims) {
250
251         # Generation of Brownian motion
252         if (!is.null(dualBrownianFileName) && file.exists(dualBrownianFileName, sep
            ="")) { cat("Loading brownian increments...\n"); load(
            dualBrownianFileName) }
253         else { dualInc = dualBrownianIncrements(nTimePoints, delta, correlation) }
254         if (!is.null(dualBrownianFileName) && !file.exists(dualBrownianFileName))
            { cat("Saving brownian increments...\n"); save(dualInc, file=
            dualBrownianFileName) }
255         inc.risky = dualInc[,1]
256         inc.var = dualInc[,2]
257         corrInc[k] = cor(inc.risky, inc.var)
258
259         #
260         # Simulated wealths until first rebalancing time point
261         #
262
263         # Simulation of the stochastic volatility
264         stochVar = 1:nTimePoints * NA
265         stochVar[1] = var.init + reversionRate*(var.long-var.init)*delta +
            volOfVol*sqrt(var.init)*inc.var[1]
266         for (i in 2:nTimePoints) { stochVar[i] = stochVar[i-1] + reversionRate*(
            var.long-stochVar[i-1])*delta + volOfVol*sqrt(stochVar[i-1])*inc.var[i
            ] }
267         stochVol = sqrt(stochVar)
268         stochVol.mean[k] = mean(stochVol)
269         stochVol.sd[k] = sd(stochVol)
270
271         # Calculation of optimal strategy
272         u.star.init = optimalControl(drift, sqrt(var.init), rent, aversion)
273         u.star = optimalControl(drift, stochVol, rent, aversion)
274         u.star.mean[k] = mean(u.star)
275
276         # Time points to be simulated (active time points)
277         activeIndices = 1:nRebDelay

```

```

278     rebPoint = tail(activeIndices,1)
279
280     # Determining active variables
281     inc.risky.active = inc.risky[activeIndices]
282     stochVol.active = stochVol[activeIndices]
283     u.star.rebPoint = u.star[rebPoint]
284     return.risky.rebPoint = prod(1+drift*delta+stochVol.active*inc.risky.
        active)
285     return.riskfree.rebPoint = prod((1+rent*delta)*ones)
286
287     # No transaction costs
288     simWealth = u.star.init*initWealth*cumprod(1+drift*delta+stochVol.active*
        inc.risky.active) + (1-u.star.init)*initWealth*cumprod((1+rent*delta)*
        ones)
289
290     # With transaction costs
291     simWealth.tc = u.star.init*initWealth*cumprod(1+drift*delta+stochVol.
        active*inc.risky.active) + (1-u.star.init)*initWealth*cumprod((1+rent*
        delta)*ones)
292     signDiff.rebPoint = sign((1-u.star.rebPoint)*u.star.init*return.risky.
        rebPoint - u.star.rebPoint*(1-u.star.init)*return.riskfree.rebPoint)
293     transCost = costProp * abs(((1-u.star.rebPoint)*u.star.init*initWealth*
        return.risky.rebPoint - u.star.rebPoint*(1-u.star.init)*initWealth*
        return.riskfree.rebPoint) / (1-signDiff.rebPoint*costProp*u.star.
        rebPoint))
294     totalTransCost[k] = totalTransCost[k] + transCost
295     simWealth.tc[rebPoint] = u.star.init*initWealth*return.risky.rebPoint +
        (1-u.star.init)*initWealth*return.riskfree.rebPoint - transCost
296
297     # Storing last rebalancing time point rebalancing strategy
298     u.star.last = u.star.rebPoint
299
300     #
301     # The rest of the simulated wealths
302     #
303
304     for (j in rebIndex[-length(rebIndex)] + 1) {
305
306         # Time points to be simulated (active time points)
307         activeIndices = j:(j+nRebDelay-1)
308         rebPoint = tail(activeIndices,1)
309
310         # Determining active variables
311         inc.risky.active = inc.risky[activeIndices]
312         stochVol.active = stochVol[activeIndices]
313         u.star.rebPoint = u.star[rebPoint]
314         return.risky.rebPoint = prod(1+drift*delta+stochVol.active*inc.risky.
            active)
315         return.riskfree.rebPoint = prod((1+rent*delta)*ones)
316
317         # No transaction costs
318         simWealth[activeIndices] = u.star.last*simWealth[j-1]*cumprod(1+drift*
            delta+stochVol.active*inc.risky.active) + (1-u.star.last)*simWealth[
            j-1]*cumprod((1+rent*delta)*ones)
319
320         # With transaction costs
321         simWealth.tc[activeIndices] = u.star.last*simWealth.tc[j-1]*cumprod(1+
            drift*delta+stochVol.active*inc.risky.active) + (1-u.star.last)*
            simWealth.tc[j-1]*cumprod((1+rent*delta)*ones)
322         signDiff.rebPoint = sign((1-u.star.rebPoint)*u.star.last*return.risky.
            rebPoint - u.star.rebPoint*(1-u.star.last)*return.riskfree.rebPoint)
323         transCost = costProp * abs(((1-u.star.rebPoint)*u.star.last*simWealth.tc
            [j-1]*return.risky.rebPoint - u.star.rebPoint*(1-u.star.last)*
            simWealth.tc[j-1]*return.riskfree.rebPoint) / (1-signDiff.rebPoint*
            costProp*u.star.rebPoint))

```

```

324     totalTransCost[k] = totalTransCost[k] + transCost
325     simWealth.tc[rebPoint] = u.star.last*simWealth.tc[j-1]*return.risky.
        rebPoint + (1-u.star.last)*simWealth.tc[j-1]*return.riskfree.
        rebPoint - transCost
326
327     # Storing last rebalancing time point rebalancing strategy
328     u.star.last = u.star.rebPoint
329     }
330
331     simWealth.sd[k] = sd(simWealth)
332     simWealth.terminal[k] = simWealth[lastIndex]
333     simWealth.logReturn = logReturn(c(initWealth, simWealth))
334     simWealth.logReturn.sd[k] = sd(simWealth.logReturn)
335
336     simWealth.tc.sd[k] = sd(simWealth.tc)
337     simWealth.tc.terminal[k] = simWealth.tc[lastIndex]
338     simWealth.tc.logReturn = logReturn(c(initWealth, simWealth.tc))
339     simWealth.tc.logReturn.sd[k] = sd(simWealth.tc.logReturn)
340   }
341 }
342
343 # Calculation of total simulation time
344 timeElapsed = proc.time()[3][[1]] - timeStart
345 cat(nSims," simulation(s) completed in",timeElapsed," seconds.\n")
346 flush.console()
347
348 # Construction of the list of data to be returned from the function.
349 if (nSims == 1) {
350   stdNames = c("simWealth.risky", "simWealth.riskfree", "simWealth", "
        transQuantity", "transCost", "propInRisky", "propInRiskfree")
351   returnList.none = list(simWealth.risky, simWealth.riskfree, simWealth,
        transQuantity, propInRisky, propInRiskfree)
352   names(returnList.none) = stdNames[-5]
353   returnList.tc = list(simWealth.tc.risky, simWealth.tc.riskfree, simWealth.tc,
        transQuantity.tc, transCost.tc, propInRisky.tc, propInRiskfree.tc)
354   names(returnList.tc) = stdNames
355   returnList = list(days, rebDays, rebIndex, inc.risky, inc.var, BM.risky, BM.vol,
        stochVol, u.star, return.risky, return.riskfree, returnList.none, returnList.
        tc)
356   names(returnList) = c("days", "rebDays", "rebIndex", "increments.risky", "
        increments.vol", "BM.risky", "BM.vol", "volatility", "u.star", "return.risky",
        "return.riskfree", "noTransCost", "transCost")
357 }
358 else {
359   paramSet = c(initWealth, nTradingDays, nDailyIncrements, nDailyRebs, drift, rent,
        aversion, costProp, var.init, reversionRate, var.long, volOfVol, correlation)
360   stdNames = c("simWealth.terminal", "simWealth.sd", "simWealth.logReturn.sd", "
        totalTransCost")
361   returnList.none = list(simWealth.terminal, simWealth.sd, simWealth.logReturn.
        sd)
362   names(returnList.none) = stdNames[-4]
363   returnList.tc = list(simWealth.tc.terminal, simWealth.tc.sd, simWealth.tc.
        logReturn.sd, totalTransCost)
364   names(returnList.tc) = stdNames
365   returnList = list(paramSet, corrInc, stochVol.mean, stochVol.sd, u.star.mean,
        returnList.none, returnList.tc)
366   names(returnList) = c("parameters", "correlation", "stochVol.mean", "stochVol.
        sd", "u.star.mean", "noTransCost", "transCost")
367 }
368
369 return(returnList)
370 }

```

## B.6.2 Execution

```

1  ##
2  # Master thesis
3  # Simulation using stochastic volatility
4  #
5
6  setwd("M:/pc/dokumenter/Master")
7  if(getwd()=="M:/pc/dokumenter/Master") Sys.setenv(TMP="E:/work/joachiah")
8
9  require(doSMP)
10 source("R/supportFunctions.R")
11 source("R/machinery_general.R")
12 source("R/initParameters.R")
13 source("R/machinery_basic.R")
14 source("R/machinery_transCost.R")
15 source("R/machinery_stochVol.R")
16
17 alpha = .05
18 qAlpha.half = qnorm(1-alpha/2)
19
20 #
21 # One test run
22 #
23
24 nSims = 1
25 paramSet.stochVol[4] = 24
26 simObject.stochVol = simPortfolio.stochVol(nSims, paramSet.stochVol, "constVsStoch
  .RData")
27 paramSet.constVol[4] = 24
28 simObject.constVol = simPortfolio.transCost(nSims, paramSet.constVol, "
  constVsStoch.RData")
29
30 days = simObject.stochVol$days
31 stochVol = simObject.stochVol$volatility
32 x.ticks = seq(0,252,21)
33 x.title = "Trading days"
34 y.title = "Volatility"
35 niceplot(days, stochVol, x.ticks, xTitle=x.title, yTitle=y.title)
36 abline(h=sqrt(var.long), lty=3)
37 legendText = "(a)"
38 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
39 savePlot("images/stochVol", type="eps")
40
41 uStar.stoch = simObject.stochVol$u.star
42 y.title = "u*"
43 niceplot(days, uStar.stoch, x.ticks, xTitle=x.title, yTitle=y.title)
44 abline(h=uStar.constVol, lty=3)
45 legendText = "(b)"
46 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
47 savePlot("images/uStar_stoch", type="eps")
48
49 breaksLength = 70
50 scalar = 1e4
51 transCost.diff = simObject.constVol$precedingTransCost$transCost - simObject.
  stochVol$transCost$transCost
52 res = seq(min(scalar*transCost.diff), max(scalar*transCost.diff), length=
  breaksLength)
53 x.title = expression(paste("Transaction cost difference", phantom(0) %*% 10^4))
54 y.title = "Frequency"
55 nicehist(scalar*transCost.diff, xTitle=x.title, yTitle=y.title, breaks=res)
56 savePlot("images/transCost_diff", type="eps")
57 print(sum(simObject.constVol$precedingTransCost$transCost))
58 print(sum(simObject.stochVol$transCost$transCost))

```

```

59 |
60 | nSims = 1
61 | paramSet.stochVol[4] = 21/252
62 | paramSet.constVol[4] = 21/252
63 | rebIndex = seq(288,6048,288)
64 | transCost.diff = 1:6000 * NA
65 |
66 | for (i in 1:500) {
67 |   simObject.stochVol = simPortfolio.stochVol(nSims, paramSet.stochVol)
68 |   simObject.constVol = simPortfolio.transCost(nSims, paramSet.constVol)
69 |   transCost.diff[(i*12-11):(i*12)] = simObject.
       |     constVol$precedingTransCost$transCost[rebIndex] - simObject.
       |     stochVol$transCost$transCost[rebIndex]
70 | }
71 |
72 | breaksLength = 70
73 | scalar = 1e3
74 | res = seq(min(scalar*transCost.diff), max(scalar*transCost.diff), length=
       |     breaksLength)
75 | x.title = expression(paste("Transaction cost difference", phantom(0) %%% 10^3))
76 | y.title = "Frequency"
77 | nicehist(scalar*transCost.diff, xTitle=x.title, yTitle=y.title, breaks=res)
78 | savePlot("images/transCost_diff_monthly", type="eps")
79 | print(sum(simObject.constVol$precedingTransCost$transCost))
80 | print(sum(simObject.stochVol$transCost$transCost))
81 |
82 | #
83 | # Multiple runs
84 | #
85 |
86 | # Performing reference simulations for comparison
87 | nSims = 50000
88 | nCores = 25
89 | nDailyRebs = 24
90 | nDailyRebs = c(24,6,1,1/2,1/12,1/21,1/42,1/126,1/252)
91 | strategyNames = c("Hourly", "Every 4th hour", "Daily", "Every 3rd day", "Every 12th
       |     day", "Monthly", "Bimonthly", "Semiannually", "Annually")
92 | volatility.const = sqrt(var.long)
93 | u.star.const = optimalControl(drift, volatility.const, rent, riskAversion)
94 | paramSets.basic = cbind(initWealth, nTradingDays, nDailyIncrements, nDailyRebs,
       |     drift, volatility.const, rent, riskAversion, u.star.const)
95 | rebStrategy.benchmark.none = distribute(nSims, nCores, simPortfolio, paramSets.
       |     basic)
96 | names(rebStrategy.benchmark.none) = strategyNames
97 |
98 | paramSets.transCost.tc01 = cbind(initWealth, nTradingDays, nDailyIncrements,
       |     nDailyRebs, drift, volatility.const, rent, riskAversion, u.star.const, costProp
       |     =.01)
99 | rebStrategy.benchmark.tc01 = distribute(nSims, nCores, simPortfolio.transCost,
       |     paramSets.transCost.tc01)
100 | names(rebStrategy.benchmark.tc01) = strategyNames
101 | n.entries = length(rebStrategy.benchmark.tc01)
102 |
103 | for (k in 1:n.entries) {
104 |
105 |   th = rebStrategy.benchmark.tc01[[k]]$theoretical
106 |   rebStrategy.benchmark.tc01[[k]]$theoretical = list(merge.list(th[seq(1,3*
       |     nCores-2,3)]), merge.list(th[seq(2,3*nCores-1,3)]), merge.list(th[seq(3,3*
       |     nCores,3)]))
107 |   names(rebStrategy.benchmark.tc01[[k]]$theoretical) = c("thWealth.terminal", "
       |     thWealth.sd", "thWealth.logReturn.sd")
108 |
109 |   no = rebStrategy.benchmark.tc01[[k]]$noTransCost
110 |   rebStrategy.benchmark.tc01[[k]]$noTransCost = list(merge.list(no[seq(1,3*
       |     nCores-2,3)]), merge.list(no[seq(2,3*nCores-1,3)]), merge.list(no[seq(3,3*

```

```

    nCores, 3) ] )
111 names(rebStrategy.benchmark.tc01[[k]]$noTransCost) = c("simWealth.terminal",
    simWealth.sd", "simWealth.logReturn.sd")
112
113 pre = rebStrategy.benchmark.tc01[[k]]$precedingTransCost
114 rebStrategy.benchmark.tc01[[k]]$precedingTransCost = list(merge.list(pre[seq
    (1,4*nCores-3,4)]), merge.list(pre[seq(2,4*nCores-2,4)]), merge.list(pre[
    seq(3,4*nCores-1,4)]), merge.list(pre[seq(4,4*nCores,4)]))
115 names(rebStrategy.benchmark.tc01[[k]]$precedingTransCost) = c("simWealth.
    terminal", "simWealth.sd", "simWealth.logReturn.sd", "totalTransCost")
116
117 sub = rebStrategy.benchmark.tc01[[k]]$subsequentTransCost
118 rebStrategy.benchmark.tc01[[k]]$subsequentTransCost = list(merge.list(sub[seq
    (1,4*nCores-3,4)]), merge.list(sub[seq(2,4*nCores-2,4)]), merge.list(sub[
    seq(3,4*nCores-1,4)]), merge.list(sub[seq(4,4*nCores,4)]))
119 names(rebStrategy.benchmark.tc01[[k]]$subsequentTransCost) = c("simWealth.
    terminal", "simWealth.sd", "simWealth.logReturn.sd", "totalTransCost")
120 }
121 save(rebStrategy.benchmark.tc01, file="Datasett/rebStrategy_stochVol_tc01_bench.
    RData")
122
123 paramSets.transCost.tc02 = cbind(initWealth, nTradingDays, nDailyIncrements,
    nDailyRebs, drift, volatility.const, rent, riskAversion, u.star.const, costProp
    =.02)
124 rebStrategy.benchmark.tc02 = distribute(nSims, nCores, simPortfolio.transCost,
    paramSets.transCost.tc02)
125 names(rebStrategy.benchmark.tc02) = strategyNames
126
127 for (k in 1:n.entries) {
128
129 th = rebStrategy.benchmark.tc02[[k]]$theoretical
130 rebStrategy.benchmark.tc02[[k]]$theoretical = list(merge.list(th[seq(1,3*
    nCores-2,3)]), merge.list(th[seq(2,3*nCores-1,3)]), merge.list(th[seq(3,3*
    nCores,3)]))
131 names(rebStrategy.benchmark.tc02[[k]]$theoretical) = c("thWealth.terminal",
    thWealth.sd", "thWealth.logReturn.sd")
132
133 no = rebStrategy.benchmark.tc02[[k]]$noTransCost
134 rebStrategy.benchmark.tc02[[k]]$noTransCost = list(merge.list(no[seq(1,3*
    nCores-2,3)]), merge.list(no[seq(2,3*nCores-1,3)]), merge.list(no[seq(3,3*
    nCores,3)]))
135 names(rebStrategy.benchmark.tc02[[k]]$noTransCost) = c("simWealth.terminal",
    simWealth.sd", "simWealth.logReturn.sd")
136
137 pre = rebStrategy.benchmark.tc02[[k]]$precedingTransCost
138 rebStrategy.benchmark.tc02[[k]]$precedingTransCost = list(merge.list(pre[seq
    (1,4*nCores-3,4)]), merge.list(pre[seq(2,4*nCores-2,4)]), merge.list(pre[
    seq(3,4*nCores-1,4)]), merge.list(pre[seq(4,4*nCores,4)]))
139 names(rebStrategy.benchmark.tc02[[k]]$precedingTransCost) = c("simWealth.
    terminal", "simWealth.sd", "simWealth.logReturn.sd", "totalTransCost")
140
141 sub = rebStrategy.benchmark.tc02[[k]]$subsequentTransCost
142 rebStrategy.benchmark.tc02[[k]]$subsequentTransCost = list(merge.list(sub[seq
    (1,4*nCores-3,4)]), merge.list(sub[seq(2,4*nCores-2,4)]), merge.list(sub[
    seq(3,4*nCores-1,4)]), merge.list(sub[seq(4,4*nCores,4)]))
143 names(rebStrategy.benchmark.tc02[[k]]$subsequentTransCost) = c("simWealth.
    terminal", "simWealth.sd", "simWealth.logReturn.sd", "totalTransCost")
144 }
145 save(rebStrategy.benchmark.tc02, file="Datasett/rebStrategy_stochVol_tc02_bench.
    RData")
146
147 paramSets.transCost.tc03 = cbind(initWealth, nTradingDays, nDailyIncrements,
    nDailyRebs, drift, volatility.const, rent, riskAversion, u.star.const, costProp
    =.03)

```

```

148 rebStrategy.benchmark.tc03 = distribute(nSims, nCores, simPortfolio.transCost,
    paramSets.transCost.tc03)
149 names(rebStrategy.benchmark.tc03) = strategyNames
150
151 for (k in 1:n.entries) {
152
153   th = rebStrategy.benchmark.tc03[[k]]$theoretical
154   rebStrategy.benchmark.tc03[[k]]$theoretical = list(merge.list(th[seq(1,3*
    nCores-2,3)]), merge.list(th[seq(2,3*nCores-1,3)]), merge.list(th[seq(3,3*
    nCores,3)]))
155   names(rebStrategy.benchmark.tc03[[k]]$theoretical) = c("thWealth.terminal",
    "thWealth.sd", "thWealth.logReturn.sd")
156
157   no = rebStrategy.benchmark.tc03[[k]]$noTransCost
158   rebStrategy.benchmark.tc03[[k]]$noTransCost = list(merge.list(no[seq(1,3*
    nCores-2,3)]), merge.list(no[seq(2,3*nCores-1,3)]), merge.list(no[seq(3,3*
    nCores,3)]))
159   names(rebStrategy.benchmark.tc03[[k]]$noTransCost) = c("simWealth.terminal",
    "simWealth.sd", "simWealth.logReturn.sd")
160
161   pre = rebStrategy.benchmark.tc03[[k]]$precedingTransCost
162   rebStrategy.benchmark.tc03[[k]]$precedingTransCost = list(merge.list(pre[seq
    (1,4*nCores-3,4)]), merge.list(pre[seq(2,4*nCores-2,4)]), merge.list(pre[
    seq(3,4*nCores-1,4)]), merge.list(pre[seq(4,4*nCores,4)]))
163   names(rebStrategy.benchmark.tc03[[k]]$precedingTransCost) = c("simWealth.
    terminal", "simWealth.sd", "simWealth.logReturn.sd", "totalTransCost")
164
165   sub = rebStrategy.benchmark.tc03[[k]]$subsequentTransCost
166   rebStrategy.benchmark.tc03[[k]]$subsequentTransCost = list(merge.list(sub[seq
    (1,4*nCores-3,4)]), merge.list(sub[seq(2,4*nCores-2,4)]), merge.list(sub[
    seq(3,4*nCores-1,4)]), merge.list(sub[seq(4,4*nCores,4)]))
167   names(rebStrategy.benchmark.tc03[[k]]$subsequentTransCost) = c("simWealth.
    terminal", "simWealth.sd", "simWealth.logReturn.sd", "totalTransCost")
168 }
169 save(rebStrategy.benchmark.tc03, file="Datasett/rebStrategy_stochVol_tc03_bench.
    RData")
170
171 #
172 # Performing simulations, transaction cost proportion = .01
173 #
174
175 nDailyRebs = c(24,6,1,1/2,1/12,1/21,1/42,1/126,1/252)
176 strategyNames = c("Hourly", "Every 4th hour", "Daily", "Every 3rd day", "Every 12th
    day", "Monthly", "Bimonthly", "Semiannually", "Annually")
177
178 costProp = .01
179 paramSets.stochVol = cbind(initWealth, nTradingDays, nDailyIncrements, nDailyRebs,
    drift, rent, riskAversion, costProp, var.init, reversionRate, var.long, volOfVol,
    correlation)
180 rebStrategy.stochVol.tc01 = distribute(nSims, nCores, simPortfolio.stochVol,
    paramSets.stochVol)
181 names(rebStrategy.stochVol.tc01) = strategyNames
182
183 Organizing returned data
184 n.entries = length(rebStrategy.stochVol.tc01)
185 for (k in 1:n.entries) {
186   rebStrategy.stochVol.tc01[[k]]$parameters = rebStrategy.stochVol.tc01[[k]]
    $parameters[1:nParam.stochVol]
187   names(rebStrategy.stochVol.tc01[[k]]$parameters) = c("initWealth",
    "nTradingDays", "nDailyIncrements", "nDailyRebs", "drift", "rent", "riskAversion",
    "costProp", "var.init", "reversionRate", "var.long", "volOfVol", "correlation")
188
189   none = rebStrategy.stochVol.tc01[[k]]$noTransCost

```



```

190 rebStrategy.stochVol.tc01[[k]]$noTransCost = list(merge.list(none[seq(1,3*
      nCores-2,3)]), merge.list(none[seq(2,3*nCores-1,3)]), merge.list(none[seq
      (3,3*nCores,3)]))
191 names(rebStrategy.stochVol.tc01[[k]]$noTransCost) = c("simWealth.terminal",
      simWealth.sd", "simWealth.logReturn.sd")
192
193 tc = rebStrategy.stochVol.tc01[[k]]$transCost
194 rebStrategy.stochVol.tc01[[k]]$transCost = list(merge.list(tc[seq(1,4*nCores
      -3,4)]), merge.list(tc[seq(2,4*nCores-2,4)]), merge.list(tc[seq(3,4*nCores
      -1,4)]), merge.list(tc[seq(4,4*nCores,4)]))
195 names(rebStrategy.stochVol.tc01[[k]]$transCost) = c("simWealth.terminal",
      simWealth.sd", "simWealth.logReturn.sd", "totalTransCost")
196 }
197 save(rebStrategy.stochVol.tc01, file="Datsett/rebStrategy.stochVol.tc01.RData")
198
199 #
200 # Performing simulations, transaction cost proportion = .02
201 #
202
203 costProp = .02
204 paramSets.stochVol = cbind(initWealth, nTradingDays, nDailyIncrements, nDailyRebs,
      drift, rent, riskAversion, costProp, var.init, reversionRate, var.long, volOfVol,
      correlation)
205 rebStrategy.stochVol.tc02 = distribute(nSims, nCores, simPortfolio.stochVol,
      paramSets.stochVol)
206 names(rebStrategy.stochVol.tc02) = strategyNames
207
208 Organizing returned data
209 for (k in 1:n.entries) {
210 rebStrategy.stochVol.tc02[[k]]$parameters = rebStrategy.stochVol.tc02[[k]]
      $parameters[1:nParam.stochVol]
211 names(rebStrategy.stochVol.tc02[[k]]$parameters) = c("initWealth",
      nTradingDays", "nDailyIncrements", "nDailyRebs", "drift", "rent", "riskAversion
      ", "costProp", "var.init", "reversionRate", "var.long", "volOfVol", "correlation
      ")
212
213 none = rebStrategy.stochVol.tc02[[k]]$noTransCost
214 rebStrategy.stochVol.tc02[[k]]$noTransCost = list(merge.list(none[seq(1,3*
      nCores-2,3)]), merge.list(none[seq(2,3*nCores-1,3)]), merge.list(none[seq
      (3,3*nCores,3)]))
215 names(rebStrategy.stochVol.tc02[[k]]$noTransCost) = c("simWealth.terminal",
      simWealth.sd", "simWealth.logReturn.sd")
216
217 tc = rebStrategy.stochVol.tc02[[k]]$transCost
218 rebStrategy.stochVol.tc02[[k]]$transCost = list(merge.list(tc[seq(1,4*nCores
      -3,4)]), merge.list(tc[seq(2,4*nCores-2,4)]), merge.list(tc[seq(3,4*nCores
      -1,4)]), merge.list(tc[seq(4,4*nCores,4)]))
219 names(rebStrategy.stochVol.tc02[[k]]$transCost) = c("simWealth.terminal",
      simWealth.sd", "simWealth.logReturn.sd", "totalTransCost")
220 }
221 save(rebStrategy.stochVol.tc02, file="Datsett/rebStrategy.stochVol.tc02.RData")
222
223
224 Performing simulations, transaction cost proportion = .03
225
226
227 costProp = .03
228 paramSets.stochVol = cbind(initWealth, nTradingDays, nDailyIncrements, nDailyRebs,
      drift, rent, riskAversion, costProp, var.init, reversionRate, var.long, volOfVol,
      correlation)
229 rebStrategy.stochVol.tc03 = distribute(nSims, nCores, simPortfolio.stochVol,
      paramSets.stochVol)
230 names(rebStrategy.stochVol.tc03) = strategyNames
231
232 Organizing returned data

```

```

233 for (k in 1:n.entries) {
234   rebStrategy.stochVol.tc03[[k]]$parameters = rebStrategy.stochVol.tc03[[k]]
      $parameters[1:nParam.stochVol]
235   names(rebStrategy.stochVol.tc03[[k]]$parameters) = c("initWealth",
      "nTradingDays", "nDailyIncrements", "nDailyRebs", "drift", "rent", "riskAversion",
      "costProp", "var.init", "reversionRate", "var.long", "volOfVol", "correlation")
236
237   none = rebStrategy.stochVol.tc03[[k]]$noTransCost
238   rebStrategy.stochVol.tc03[[k]]$noTransCost = list(merge.list(none[seq(1,3*
      nCores-2,3)]), merge.list(none[seq(2,3*nCores-1,3)]), merge.list(none[seq(
      3,3*nCores,3)]))
239   names(rebStrategy.stochVol.tc03[[k]]$noTransCost) = c("simWealth.terminal",
      "simWealth.sd", "simWealth.logReturn.sd")
240
241   tc = rebStrategy.stochVol.tc03[[k]]$transCost
242   rebStrategy.stochVol.tc03[[k]]$transCost = list(merge.list(tc[seq(1,4*nCores
      -3,4)]), merge.list(tc[seq(2,4*nCores-2,4)]), merge.list(tc[seq(3,4*nCores
      -1,4)]), merge.list(tc[seq(4,4*nCores,4)]))
243   names(rebStrategy.stochVol.tc03[[k]]$transCost) = c("simWealth.terminal",
      "simWealth.sd", "simWealth.logReturn.sd", "totalTransCost")
244 }
245 save(rebStrategy.stochVol.tc03, file="Datasett/rebStrategy_stochVol_tc03.RData")
246
247 #
248 # Calculating relevant statistics and plotting
249 # Transaction cost proportion = 0
250 #
251
252 cat("\nTransaction cost proportion = 0\n\n")
253
254 x.labels = c("0", ".5k", "1.0k", "1.5k", "2.0k", "2.5k", "3.0k", "3.5k", "4.0k", "4.5k",
      "5.0k")
255 nn = 1:nSims
256
257 terminalWealth.tc01.none.bench = matrix(NA, nSims, n.entries)
258 sdWealth.tc01.none.bench = matrix(NA, nSims, n.entries)
259 sdLogReturn.tc01.none.bench = matrix(NA, nSims, n.entries)
260 for (k in 1:n.entries) {
261   terminalWealth.tc01.none.bench[,k] = rebStrategy.benchmark.tc01[[c(k,2)]]
      $simWealth.terminal
262   sdWealth.tc01.none.bench[,k] = rebStrategy.benchmark.tc01[[c(k,2)]]$simWealth.
      sd
263   sdLogReturn.tc01.none.bench[,k] = rebStrategy.benchmark.tc01[[c(k,2)]]
      $simWealth.logReturn.sd
264 }
265 terminalWealth.tc02.none.bench = matrix(NA, nSims, n.entries)
266 sdWealth.tc02.none.bench = matrix(NA, nSims, n.entries)
267 sdLogReturn.tc02.none.bench = matrix(NA, nSims, n.entries)
268 for (k in 1:n.entries) {
269   terminalWealth.tc02.none.bench[,k] = rebStrategy.benchmark.tc02[[c(k,2)]]
      $simWealth.terminal
270   sdWealth.tc02.none.bench[,k] = rebStrategy.benchmark.tc02[[c(k,2)]]$simWealth.
      sd
271   sdLogReturn.tc02.none.bench[,k] = rebStrategy.benchmark.tc02[[c(k,2)]]
      $simWealth.logReturn.sd
272 }
273 terminalWealth.tc03.none.bench = matrix(NA, nSims, n.entries)
274 sdWealth.tc03.none.bench = matrix(NA, nSims, n.entries)
275 sdLogReturn.tc03.none.bench = matrix(NA, nSims, n.entries)
276 for (k in 1:n.entries) {
277   terminalWealth.tc03.none.bench[,k] = rebStrategy.benchmark.tc03[[c(k,2)]]
      $simWealth.terminal
278   sdWealth.tc03.none.bench[,k] = rebStrategy.benchmark.tc03[[c(k,2)]]$simWealth.
      sd

```

```

279   sdLogReturn.tc03.none.bench[,k] = rebStrategy.benchmark.tc03[[c(k,2)]]
      $simWealth.logReturn.sd
280 }
281
282 terminalWealth.none.bench = rbind(terminalWealth.tc01.none.bench, terminalWealth.
      tc02.none.bench, terminalWealth.tc03.none.bench)
283 terminalWealth.none.mean.bench = colMeans(terminalWealth.none.bench)
284 terminalUtility.none.bench = utility(terminalWealth.none.bench, riskAversion)
285 terminalUtility.none.mean.bench = colMeans(terminalUtility.none.bench)
286 terminalUtility.none.sd.bench = colSds(terminalUtility.none.bench)
287
288 logReturn.none.bench = log(terminalWealth.none.bench)
289 logReturn.none.mean.bench = colMeans(logReturn.none.bench)
290 sdLogReturn.none.bench = rbind(sdLogReturn.tc01.none.bench, sdLogReturn.tc02.none
      .bench, sdLogReturn.tc03.none.bench)
291 volatility.none.bench = sdLogReturn.none.bench * sqrt(nTimePoints)
292 volatility.none.mean.bench = colMeans(volatility.none.bench)
293 excessReturn.none.bench = logReturn.none.bench - rent
294 sharpeRatio.none.bench = excessReturn.none.bench / volatility.none.bench
295 sharpeRatio.none.mean.bench = colMeans(sharpeRatio.none.bench)
296 volOfVol.none.bench = colSds(volatility.none.bench)
297 correlation.none.bench = colCorrs(logReturn.none.bench, volatility.none.bench)
298
299 terminalWealth.tc01.none = matrix(NA, nSims, n.entries)
300 sdWealth.tc01.none = matrix(NA, nSims, n.entries)
301 sdLogReturn.tc01.none = matrix(NA, nSims, n.entries)
302 for (k in 1:n.entries) {
303   terminalWealth.tc01.none[,k] = rebStrategy.stochVol.tc01[[c(k,6)]] $simWealth.
      terminal
304   sdWealth.tc01.none[,k] = rebStrategy.stochVol.tc01[[c(k,6)]] $simWealth.sd
305   sdLogReturn.tc01.none[,k] = rebStrategy.stochVol.tc01[[c(k,6)]] $simWealth.
      logReturn.sd
306 }
307 terminalWealth.tc02.none = matrix(NA, nSims, n.entries)
308 sdWealth.tc02.none = matrix(NA, nSims, n.entries)
309 sdLogReturn.tc02.none = matrix(NA, nSims, n.entries)
310 for (k in 1:n.entries) {
311   terminalWealth.tc02.none[,k] = rebStrategy.stochVol.tc02[[c(k,6)]] $simWealth.
      terminal
312   sdWealth.tc02.none[,k] = rebStrategy.stochVol.tc02[[c(k,6)]] $simWealth.sd
313   sdLogReturn.tc02.none[,k] = rebStrategy.stochVol.tc02[[c(k,6)]] $simWealth.
      logReturn.sd
314 }
315 terminalWealth.tc03.none = matrix(NA, nSims, n.entries)
316 sdWealth.tc03.none = matrix(NA, nSims, n.entries)
317 sdLogReturn.tc03.none = matrix(NA, nSims, n.entries)
318 for (k in 1:n.entries) {
319   terminalWealth.tc03.none[,k] = rebStrategy.stochVol.tc03[[c(k,6)]] $simWealth.
      terminal
320   sdWealth.tc03.none[,k] = rebStrategy.stochVol.tc03[[c(k,6)]] $simWealth.sd
321   sdLogReturn.tc03.none[,k] = rebStrategy.stochVol.tc03[[c(k,6)]] $simWealth.
      logReturn.sd
322 }
323
324 terminalWealth.none = rbind(terminalWealth.tc01.none, terminalWealth.tc02.none,
      terminalWealth.tc03.none)
325 nSims = nrow(terminalWealth.none)
326 terminalWealth.none.mean = colMeans(terminalWealth.none)
327 terminalUtility.none = utility(terminalWealth.none, riskAversion)
328 terminalUtility.none.mean = colMeans(terminalUtility.none)
329 terminalUtility.none.sd = colSds(terminalUtility.none)
330 lossOfUtility.none.mean = terminalUtility.none.mean.bench - terminalUtility.none
      .mean
331 lossOfUtility.none.mean.CL.lower = lossOfUtility.none.mean - qAlpha.half * (1/
      sqrt(nSims)) * sqrt(terminalUtility.none.sd.bench^2 + terminalUtility.none.

```

```

    sd^2)
332 lossOfUtility.none.mean.CL.upper = lossOfUtility.none.mean + qAlpha.half * (1/
    sqrt(nSims)) * sqrt(terminalUtility.none.sd.bench^2 + terminalUtility.none.
    sd^2)
333
334 logReturn.none = log(terminalWealth.none)
335 logReturn.none.mean = colMeans(logReturn.none)
336 sdLogReturn.none = rbind(sdLogReturn.tc01.none, sdLogReturn.tc02.none, sdLogReturn
    .tc03.none)
337 volatility.none = sdLogReturn.none * sqrt(nTimePoints)
338 volatility.none.mean = colMeans(volatility.none)
339 excessReturn.none = logReturn.none - rent
340 sharpeRatio.none = excessReturn.none / volatility.none
341 sharpeRatio.none.mean = colMeans(sharpeRatio.none)
342 sharpeRatio.none.sd = colSds(sharpeRatio.none)
343 sharpeRatio.none.mean.CL.lower = sharpeRatio.none.mean - qAlpha.half *
    sharpeRatio.none.sd / sqrt(nSims)
344 sharpeRatio.none.mean.CL.upper = sharpeRatio.none.mean + qAlpha.half *
    sharpeRatio.none.sd / sqrt(nSims)
345 volOfVol.none = colSds(volatility.none)
346 correlation.none = colCorrs(logReturn.none, volatility.none)
347
348 tab1.none = matrix(NA,18,4)
349
350 for (i in 1:9) {
351   tab1.none[2*i-1,] = c(terminalWealth.none.mean.bench[i],0, terminalUtility.none
    .mean.bench[i],0)
352   tab1.none[2*i,] = c(terminalWealth.none.mean[i],0, terminalUtility.none.mean[
    i], lossOfUtility.none.mean[i])
353 }
354 tab1.none[,4] = tab1.none[,4] * 1e4
355 tab1.none = round(tab1.none,4)
356
357 for (i in 1:18) { tab1.none[i,4] = paste(tab1.none[i,4], "\\e{\\text-4}", sep="")
    }
358 tab1.none[,2] = "-"
359 for (i in seq(1,17,2)) { tab1.none[i,4] = "-" }
360
361 printex(tab1.none)
362
363 tab2.none = matrix(NA,18,5)
364
365 for (i in 1:9) {
366   tab2.none[2*i-1,] = c(logReturn.none.mean.bench[i], volatility.none.mean.bench[
    i], sharpeRatio.none.mean.bench[i], volOfVol.none.bench[i], correlation.none.
    bench[i])
367   tab2.none[2*i,] = c(logReturn.none.mean[i], volatility.none.mean[i], sharpeRatio
    .none.mean[i], volOfVol.none[i], correlation.none[i])
368 }
369 tab2.none[,1] = tab2.none[,1] * 1e2
370 tab2.none[,3] = tab2.none[,3] * 1e2
371 tab2.none[,4] = tab2.none[,4] * 1e3
372 tab2.none = round(tab2.none,4)
373
374 for (i in 1:18) {
375   tab2.none[i,1] = paste(tab2.none[i,1], "\\e{\\text-2}", sep="")
376   tab2.none[i,3] = paste(tab2.none[i,3], "\\e{\\text-2}", sep="")
377   tab2.none[i,4] = paste(tab2.none[i,4], "\\e{\\text-3}", sep="")
378 }
379
380 printex(tab2.none)
381
382 scalar = 1e4
383 x.labels = strategyNames
384 x.title = "Rebalancing strategy"

```

```

385 y.title = expression(paste("Mean loss of utility",phantom(0) %*% 10^4))
386 x.ticks = 1:9
387 y.range = range(c(lossOfUtility.none.mean.CL.lower*scalar, lossOfUtility.none.
    mean.CL.upper*scalar))
388 niceplot(lossOfUtility.none.mean*scalar, xLabels=x.labels, xTitle=x.title, yTitle=y
    .title, y.addCustom=.2, figsPerPage=4, ylim=y.range)
389 abline(h=0, col="darkgray", lty=3)
390 abline(v=x.ticks, col="darkgray", lty=3)
391 nicelines(lossOfUtility.none.mean.CL.lower*scalar, lty=2)
392 nicelines(lossOfUtility.none.mean.CL.upper*scalar, lty=2)
393 legendText = "(a) No transaction costs"
394 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
395 savePlot("images/lossOfUtility_stochVol_none", type="eps")
396
397 y.title = "Mean Sharpe ratio"
398 y.range = range(c(sharpeRatio.none.mean.CL.lower, sharpeRatio.none.mean.CL.upper)
    )
399 niceplot(sharpeRatio.none.mean, xLabels=x.labels, xTitle=x.title, yTitle=y.title,
    figsPerPage=4, ylim=y.range)
400 abline(v=x.ticks, col="darkgray", lty=3)
401 nicelines(sharpeRatio.none.mean.CL.lower, lty=2)
402 nicelines(sharpeRatio.none.mean.CL.upper, lty=2)
403 legendText = "(a) No transaction costs"
404 nicelegend("left", legendText, bty="n", bg="white", cex=.7)
405 savePlot("images/sharpeRatio_stochVol_none", type="eps")
406
407 #
408 # Calculating relevant statistics and plotting
409 # Transaction cost proportion = .01
410 #
411
412 nSims = 50000
413
414 cat("Transaction cost proportion = .01\n")
415
416 terminalWealth.tc01.bench = matrix(NA, nSims, n.entries)
417 sdWealth.tc01.bench = matrix(NA, nSims, n.entries)
418 sdLogReturn.tc01.bench = matrix(NA, nSims, n.entries)
419 transCost.tc01.bench = matrix(NA, nSims, n.entries)
420 for (k in 1:n.entries) {
421   terminalWealth.tc01.bench[,k] = rebStrategy.benchmark.tc01[[c(k,3)]]$simWealth
    .terminal
422   sdWealth.tc01.bench[,k] = rebStrategy.benchmark.tc01[[c(k,3)]]$simWealth.sd
423   sdLogReturn.tc01.bench[,k] = rebStrategy.benchmark.tc01[[c(k,3)]]$simWealth.
    logReturn.sd
424   transCost.tc01.bench[,k] = rebStrategy.benchmark.tc01[[c(k,3)]]$totalTransCost
425 }
426
427 terminalWealth.tc01.mean.bench = colMeans(terminalWealth.tc01.bench)
428 transCost.tc01.mean.bench = colMeans(transCost.tc01.bench)
429 terminalUtility.tc01.bench = utility(terminalWealth.tc01.bench, riskAversion)
430 terminalUtility.tc01.mean.bench = colMeans(terminalUtility.tc01.bench)
431 terminalUtility.tc01.sd.bench = colSds(terminalUtility.tc01.bench)
432
433 logReturn.tc01.bench = log(terminalWealth.tc01.bench)
434 logReturn.tc01.mean.bench = colMeans(logReturn.tc01.bench)
435 volatility.tc01.bench = sdLogReturn.tc01.bench * sqrt(nTimePoints)
436 volatility.tc01.mean.bench = colMeans(volatility.tc01.bench)
437 excessReturn.tc01.bench = logReturn.tc01.bench - rent
438 sharpeRatio.tc01.bench = excessReturn.tc01.bench / volatility.tc01.bench
439 sharpeRatio.tc01.mean.bench = colMeans(sharpeRatio.tc01.bench)
440 volOfVol.tc01.bench = colSds(volatility.tc01.bench)
441 correlation.tc01.bench = colCorrs(logReturn.tc01.bench, volatility.tc01.bench)
442
443 terminalWealth.tc01 = matrix(NA, nSims, n.entries)

```

```

444 sdWealth.tc01 = matrix(NA,nSims,n.entries)
445 sdLogReturn.tc01 = matrix(NA,nSims,n.entries)
446 transCost.tc01 = matrix(NA,nSims,n.entries)
447 for (k in 1:n.entries) {
448   terminalWealth.tc01[,k] = rebStrategy.stochVol.tc01[[c(k,7)]]$simWealth.
      terminal
449   sdWealth.tc01[,k] = rebStrategy.stochVol.tc01[[c(k,7)]]$simWealth.sd
450   sdLogReturn.tc01[,k] = rebStrategy.stochVol.tc01[[c(k,7)]]$simWealth.logReturn
      .sd
451   transCost.tc01[,k] = rebStrategy.stochVol.tc01[[c(k,7)]]$totalTransCost
452 }
453
454 terminalWealth.tc01.mean = colMeans(terminalWealth.tc01)
455 transCost.tc01.mean = colMeans(transCost.tc01)
456 terminalUtility.tc01 = utility(terminalWealth.tc01,riskAversion)
457 terminalUtility.tc01.mean = colMeans(terminalUtility.tc01)
458 terminalUtility.tc01.sd = colSds(terminalUtility.tc01)
459 lossOfUtility.tc01.mean = terminalUtility.tc01.mean.bench - terminalUtility.tc01
      .mean
460 lossOfUtility.tc01.mean.CL.lower = lossOfUtility.tc01.mean - qAlpha.half * (1/
      sqrt(nSims)) * sqrt(terminalUtility.tc01.sd.bench^2 + terminalUtility.tc01.
      sd^2)
461 lossOfUtility.tc01.mean.CL.upper = lossOfUtility.tc01.mean + qAlpha.half * (1/
      sqrt(nSims)) * sqrt(terminalUtility.tc01.sd.bench^2 + terminalUtility.tc01.
      sd^2)
462
463 lossOfUtility.tc01.mean.prime = terminalUtility.tc01.mean.bench -
      terminalUtility.tc01.mean
464
465 logReturn.tc01 = log(terminalWealth.tc01)
466 logReturn.tc01.mean = colMeans(logReturn.tc01)
467 volatility.tc01 = sdLogReturn.tc01 * sqrt(nTimePoints)
468 volatility.tc01.mean = colMeans(volatility.tc01)
469 excessReturn.tc01 = logReturn.tc01 - rent
470 sharpeRatio.tc01 = excessReturn.tc01 / volatility.tc01
471 sharpeRatio.tc01.mean = colMeans(sharpeRatio.tc01)
472 sharpeRatio.tc01.sd = colSds(sharpeRatio.tc01)
473 sharpeRatio.tc01.mean.CL.lower = sharpeRatio.tc01.mean - qAlpha.half *
      sharpeRatio.tc01.sd / sqrt(nSims)
474 sharpeRatio.tc01.mean.CL.upper = sharpeRatio.tc01.mean + qAlpha.half *
      sharpeRatio.tc01.sd / sqrt(nSims)
475 volOfVol.tc01 = colSds(volatility.tc01)
476 correlation.tc01 = colCorrs(logReturn.tc01,volatility.tc01)
477
478 tab1.tc01 = matrix(NA,18,4)
479
480 for (i in 1:9) {
481   tab1.tc01[2*i-1,] = c(terminalWealth.tc01.mean.bench[i],transCost.tc01.mean.
      bench[i],terminalUtility.tc01.mean.bench[i],0)
482   tab1.tc01[2*i,] = c(terminalWealth.tc01.mean[i],transCost.tc01.mean[i],
      terminalUtility.tc01.mean[i],lossOfUtility.tc01.mean[i])
483 }
484
485 tab1.tc01[,2] = tab1.tc01[,2] * 1e2
486 tab1.tc01[,4] = tab1.tc01[,4] * 1e2
487 tab1.tc01 = round(tab1.tc01,4)
488
489 for (i in 1:18) {
490   tab1.tc01[i,2] = paste(tab1.tc01[i,2],"\\e{\\text-2}",sep="")
491   tab1.tc01[i,4] = paste(tab1.tc01[i,4],"\\e{\\text-2}",sep="")
492 }
493 for (i in seq(1,17,2)) { tab1.tc01[i,4] = "-" }
494
495 printex(tab1.tc01)
496

```

```

497 tab2.tc01 = matrix(NA,18,5)
498
499 for (i in 1:9) {
500   tab2.tc01[2*i-1,] = c(logReturn.tc01.mean.bench[i], volatility.tc01.mean.bench[
      i], sharpeRatio.tc01.mean.bench[i], volOfVol.tc01.bench[i], correlation.tc01.
      bench[i])
501   tab2.tc01[2*i,] = c(logReturn.tc01.mean[i], volatility.tc01.mean[i], sharpeRatio
      .tc01.mean[i], volOfVol.tc01[i], correlation.tc01[i])
502 }
503
504 tab2.tc01[,1] = tab2.tc01[,1] * 1e2
505 tab2.tc01[,4] = tab2.tc01[,4] * 1e3
506 tab2.tc01 = round(tab2.tc01,4)
507
508 for (i in 1:18) {
509   tab2.tc01[i,1] = paste(tab2.tc01[i,1], "\\e{\\text-2}", sep="")
510   tab2.tc01[i,4] = paste(tab2.tc01[i,4], "\\e{\\text-3}", sep="")
511 }
512
513 printex(tab2.tc01)
514
515 scalar = 1e2
516 x.labels = strategyNames
517 x.title = "Rebalancing strategy"
518 y.title = expression(paste("Mean loss of utility",phantom(0) %*% 10^2))
519 x.ticks = 1:9
520 y.range = range(c(lossOfUtility.tc01.mean.CL.lower*scalar, lossOfUtility.tc01.
      mean.CL.upper*scalar))
521 niceplot(lossOfUtility.tc01.mean*scalar, xLabels=x.labels, xTitle=x.title, yTitle=y
      .title, y.addCustom=.2, figsPerPage=4, ylim=y.range)
522 abline(h=0, col="darkgray", lty=3)
523 abline(v=x.ticks, col="darkgray", lty=3)
524 nicelines(lossOfUtility.tc01.mean.CL.lower*scalar, lty=2)
525 nicelines(lossOfUtility.tc01.mean.CL.upper*scalar, lty=2)
526 legendText = c(expression(paste("(b)", lambda*=".01")))
527 nicelegend("left", legendText, bty="n", bg="white", cex=.7)
528 savePlot("images/lossOfUtility_stochVol_tc01", type="eps")
529
530 y.title = "Mean Sharpe ratio"
531 y.range = range(c(sharpeRatio.tc01.mean.CL.lower, sharpeRatio.tc01.mean.CL.upper)
      )
532 niceplot(sharpeRatio.tc01.mean, xLabels=x.labels, xTitle=x.title, yTitle=y.title,
      figsPerPage=4, ylim=y.range)
533 abline(v=x.ticks, col="darkgray", lty=3)
534 nicelines(sharpeRatio.tc01.mean.CL.lower, lty=2)
535 nicelines(sharpeRatio.tc01.mean.CL.upper, lty=2)
536 legendText = c(expression(paste("(b)", lambda*=".01")))
537 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
538 savePlot("images/sharpeRatio_stochVol_tc01", type="eps")
539
540 #
541 # Calculating relevant statistics and plotting
542 # Transaction cost proportion = .02
543 #
544
545 cat("Transaction cost proportion = .02\n")
546
547 terminalWealth.tc02.bench = matrix(NA, nSims, n.entries)
548 sdWealth.tc02.bench = matrix(NA, nSims, n.entries)
549 sdLogReturn.tc02.bench = matrix(NA, nSims, n.entries)
550 transCost.tc02.bench = matrix(NA, nSims, n.entries)
551 for (k in 1:n.entries) {
552   terminalWealth.tc02.bench[,k] = rebStrategy.benchmark.tc02[[c(k,3)]]$simWealth
      .terminal
553   sdWealth.tc02.bench[,k] = rebStrategy.benchmark.tc02[[c(k,3)]]$simWealth.sd

```

```

554 | sdLogReturn.tc02.bench[,k] = rebStrategy.benchmark.tc02[[c(k,3)]]$simWealth.
      |   logReturn.sd
555 |   transCost.tc02.bench[,k] = rebStrategy.benchmark.tc02[[c(k,3)]]$totalTransCost
556 | }
557 |
558 | terminalWealth.tc02.mean.bench = colMeans(terminalWealth.tc02.bench)
559 | transCost.tc02.mean.bench = colMeans(transCost.tc02.bench)
560 | terminalUtility.tc02.bench = utility(terminalWealth.tc02.bench, riskAversion)
561 | terminalUtility.tc02.mean.bench = colMeans(terminalUtility.tc02.bench)
562 | terminalUtility.tc02.sd.bench = colSds(terminalUtility.tc02.bench)
563 |
564 | logReturn.tc02.bench = log(terminalWealth.tc02.bench)
565 | logReturn.tc02.mean.bench = colMeans(logReturn.tc02.bench)
566 | volatility.tc02.bench = sdLogReturn.tc02.bench * sqrt(nTimePoints)
567 | volatility.tc02.mean.bench = colMeans(volatility.tc02.bench)
568 | excessReturn.tc02.bench = logReturn.tc02.bench - rent
569 | sharpeRatio.tc02.bench = excessReturn.tc02.bench / volatility.tc02.bench
570 | sharpeRatio.tc02.mean.bench = colMeans(sharpeRatio.tc02.bench)
571 | volOfVol.tc02.bench = colSds(volatility.tc02.bench)
572 | correlation.tc02.bench = colCorrs(logReturn.tc02.bench, volatility.tc02.bench)
573 |
574 | terminalWealth.tc02 = matrix(NA, nSims, n.entries)
575 | sdWealth.tc02 = matrix(NA, nSims, n.entries)
576 | sdLogReturn.tc02 = matrix(NA, nSims, n.entries)
577 | transCost.tc02 = matrix(NA, nSims, n.entries)
578 | for (k in 1:n.entries) {
579 |   terminalWealth.tc02[,k] = rebStrategy.stochVol.tc02[[c(k,7)]]$simWealth.
      |     terminal
580 |   sdWealth.tc02[,k] = rebStrategy.stochVol.tc02[[c(k,7)]]$simWealth.sd
581 |   sdLogReturn.tc02[,k] = rebStrategy.stochVol.tc02[[c(k,7)]]$simWealth.logReturn
      |     .sd
582 |   transCost.tc02[,k] = rebStrategy.stochVol.tc02[[c(k,7)]]$totalTransCost
583 | }
584 |
585 | terminalWealth.tc02.mean = colMeans(terminalWealth.tc02)
586 | transCost.tc02.mean = colMeans(transCost.tc02)
587 | terminalUtility.tc02 = utility(terminalWealth.tc02, riskAversion)
588 | terminalUtility.tc02.mean = colMeans(terminalUtility.tc02)
589 | terminalUtility.tc02.sd = colSds(terminalUtility.tc02)
590 | lossOfUtility.tc02.mean = terminalUtility.tc02.mean.bench - terminalUtility.tc02
      | .mean
591 | lossOfUtility.tc02.mean.CL.lower = lossOfUtility.tc02.mean - qAlpha.half * (1/
      | sqrt(nSims)) * sqrt(terminalUtility.tc02.sd.bench^2 + terminalUtility.tc02.
      | sd^2)
592 | lossOfUtility.tc02.mean.CL.upper = lossOfUtility.tc02.mean + qAlpha.half * (1/
      | sqrt(nSims)) * sqrt(terminalUtility.tc02.sd.bench^2 + terminalUtility.tc02.
      | sd^2)
593 |
594 | logReturn.tc02 = log(terminalWealth.tc02)
595 | logReturn.tc02.mean = colMeans(logReturn.tc02)
596 | volatility.tc02 = sdLogReturn.tc02 * sqrt(nTimePoints)
597 | volatility.tc02.mean = colMeans(volatility.tc02)
598 | excessReturn.tc02 = logReturn.tc02 - rent
599 | sharpeRatio.tc02 = excessReturn.tc02 / volatility.tc02
600 | sharpeRatio.tc02.mean = colMeans(sharpeRatio.tc02)
601 | sharpeRatio.tc02.sd = colSds(sharpeRatio.tc02)
602 | sharpeRatio.tc02.mean.CL.lower = sharpeRatio.tc02.mean - qAlpha.half *
      | sharpeRatio.tc02.sd / sqrt(nSims)
603 | sharpeRatio.tc02.mean.CL.upper = sharpeRatio.tc02.mean + qAlpha.half *
      | sharpeRatio.tc02.sd / sqrt(nSims)
604 | volOfVol.tc02 = colSds(volatility.tc02)
605 | correlation.tc02 = colCorrs(logReturn.tc02, volatility.tc02)
606 |
607 | tab1.tc02 = matrix(NA, 18, 4)
608 |

```



```

609 for (i in 1:9) {
610   tab1.tc02[2*i-1,] = c(terminalWealth.tc02.mean.bench[i], transCost.tc02.mean.
      bench[i], terminalUtility.tc02.mean.bench[i], 0)
611   tab1.tc02[2*i,] = c(terminalWealth.tc02.mean[i], transCost.tc02.mean[i],
      terminalUtility.tc02.mean[i], lossOfUtility.tc02.mean[i])
612 }
613
614 tab1.tc02[,2] = tab1.tc02[,2] * 1e2
615 tab1.tc02[,4] = tab1.tc02[,4] * 1e2
616 tab1.tc02 = round(tab1.tc02,4)
617
618 for (i in 1:18) {
619   tab1.tc02[i,2] = paste(tab1.tc02[i,2], "\\e{\\text-2}", sep="")
620   tab1.tc02[i,4] = paste(tab1.tc02[i,4], "\\e{\\text-2}", sep="")
621 }
622 for (i in seq(1,17,2)) { tab1.tc02[i,4] = "-" }
623
624 printex(tab1.tc02)
625
626 tab2.tc02 = matrix(NA,18,5)
627
628 for (i in 1:9) {
629   tab2.tc02[2*i-1,] = c(logReturn.tc02.mean.bench[i], volatility.tc02.mean.bench[
      i], sharpeRatio.tc02.mean.bench[i], volOfVol.tc02.bench[i], correlation.tc02.
      bench[i])
630   tab2.tc02[2*i,] = c(logReturn.tc02.mean[i], volatility.tc02.mean[i], sharpeRatio
      .tc02.mean[i], volOfVol.tc02[i], correlation.tc02[i])
631 }
632
633 tab2.tc02[,1] = tab2.tc02[,1] * 1e2
634 tab2.tc02[,4] = tab2.tc02[,4] * 1e3
635 tab2.tc02 = round(tab2.tc02,4)
636
637 for (i in 1:18) {
638   tab2.tc02[i,1] = paste(tab2.tc02[i,1], "\\e{\\text-2}", sep="")
639   tab2.tc02[i,4] = paste(tab2.tc02[i,4], "\\e{\\text-3}", sep="")
640 }
641
642 printex(tab2.tc02)
643
644 scalar = 1e2
645 x.labels = strategyNames
646 x.title = "Rebalancing strategy"
647 y.title = expression(paste("Mean loss of utility",phantom(0) %*% 10^2))
648 x.ticks = 1:9
649 y.range = range(c(lossOfUtility.tc02.mean.CL.lower*scalar, lossOfUtility.tc02.
      mean.CL.upper*scalar))
650 niceplot(lossOfUtility.tc02.mean*scalar, xLabels=x.labels, xTitle=x.title, yTitle=y
      .title, y.addCustom=.2, figsPerPage=4, ylim=y.range)
651 abline(h=0, col="darkgray", lty=3)
652 abline(v=x.ticks, col="darkgray", lty=3)
653 nicelines(lossOfUtility.tc02.mean.CL.lower*scalar, lty=2)
654 nicelines(lossOfUtility.tc02.mean.CL.upper*scalar, lty=2)
655 legendText = c(expression(paste("(c)", lambda*=".02")))
656 nicelegend("left", legendText, bty="n", bg="white", cex=.7)
657 savePlot("images/lossOfUtility_stochVol_tc02", type="eps")
658
659 y.title = "Mean Sharpe ratio"
660 y.range = range(c(sharpeRatio.tc02.mean.CL.lower, sharpeRatio.tc02.mean.CL.upper)
      )
661 niceplot(sharpeRatio.tc02.mean, xLabels=x.labels, xTitle=x.title, yTitle=y.title,
      figsPerPage=4, ylim=y.range)
662 abline(v=x.ticks, col="darkgray", lty=3)
663 nicelines(sharpeRatio.tc02.mean.CL.lower, lty=2)
664 nicelines(sharpeRatio.tc02.mean.CL.upper, lty=2)

```

```

665 legendText = c(expression(paste("(c)", lambda*=".02")))
666 nicelegend("topleft", legendText, bty="n", bg="white", cex=.7)
667 savePlot("images/sharpeRatio_stochVol_tc02", type="eps")
668
669 #
670 # Calculating relevant statistics and plotting
671 # Transaction cost proportion = .03
672 #
673
674 cat("\nTransaction cost proportion = .03\n\n")
675
676 terminalWealth.tc03.bench = matrix(NA, nSims, n.entries)
677 sdWealth.tc03.bench = matrix(NA, nSims, n.entries)
678 sdLogReturn.tc03.bench = matrix(NA, nSims, n.entries)
679 transCost.tc03.bench = matrix(NA, nSims, n.entries)
680 for (k in 1:n.entries) {
681   terminalWealth.tc03.bench[,k] = rebStrategy.benchmark.tc03[[c(k,3)]]$simWealth
682     .terminal
683   sdWealth.tc03.bench[,k] = rebStrategy.benchmark.tc03[[c(k,3)]]$simWealth.sd
684   sdLogReturn.tc03.bench[,k] = rebStrategy.benchmark.tc03[[c(k,3)]]$simWealth.
685     logReturn.sd
686   transCost.tc03.bench[,k] = rebStrategy.benchmark.tc03[[c(k,3)]]$totalTransCost
687 }
688
689 terminalWealth.tc03.mean.bench = colMeans(terminalWealth.tc03.bench)
690 transCost.tc03.mean.bench = colMeans(transCost.tc03.bench)
691 terminalUtility.tc03.bench = utility(terminalWealth.tc03.bench, riskAversion)
692 terminalUtility.tc03.mean.bench = colMeans(terminalUtility.tc03.bench)
693 terminalUtility.tc03.sd.bench = colSds(terminalUtility.tc03.bench)
694
695 logReturn.tc03.bench = log(terminalWealth.tc03.bench)
696 logReturn.tc03.mean.bench = colMeans(logReturn.tc03.bench)
697 volatility.tc03.bench = sdLogReturn.tc03.bench * sqrt(nTimePoints)
698 volatility.tc03.mean.bench = colMeans(volatility.tc03.bench)
699 excessReturn.tc03.bench = logReturn.tc03.bench - rent
700 sharpeRatio.tc03.bench = excessReturn.tc03.bench / volatility.tc03.bench
701 sharpeRatio.tc03.mean.bench = colMeans(sharpeRatio.tc03.bench)
702 volOfVol.tc03.bench = colSds(volatility.tc03.bench)
703 correlation.tc03.bench = colCorrs(logReturn.tc03.bench, volatility.tc03.bench)
704
705 terminalWealth.tc03 = matrix(NA, nSims, n.entries)
706 sdWealth.tc03 = matrix(NA, nSims, n.entries)
707 sdLogReturn.tc03 = matrix(NA, nSims, n.entries)
708 transCost.tc03 = matrix(NA, nSims, n.entries)
709 for (k in 1:n.entries) {
710   terminalWealth.tc03[,k] = rebStrategy.stochVol.tc03[[c(k,7)]]$simWealth.
711     terminal
712   sdWealth.tc03[,k] = rebStrategy.stochVol.tc03[[c(k,7)]]$simWealth.sd
713   sdLogReturn.tc03[,k] = rebStrategy.stochVol.tc03[[c(k,7)]]$simWealth.logReturn
714     .sd
715   transCost.tc03[,k] = rebStrategy.stochVol.tc03[[c(k,7)]]$totalTransCost
716 }
717
718 terminalWealth.tc03.mean = colMeans(terminalWealth.tc03)
719 transCost.tc03.mean = colMeans(transCost.tc03)
720 terminalUtility.tc03 = utility(terminalWealth.tc03, riskAversion)
721 terminalUtility.tc03.mean = colMeans(terminalUtility.tc03)
722 terminalUtility.tc03.sd = colSds(terminalUtility.tc03)
723 lossOfUtility.tc03.mean = terminalUtility.tc03.mean.bench - terminalUtility.tc03
724   .mean
725 lossOfUtility.tc03.mean.CL.lower = lossOfUtility.tc03.mean - qAlpha.half * (1/
726   sqrt(nSims)) * sqrt(terminalUtility.tc03.sd.bench^2 + terminalUtility.tc03.
727   sd^2)
728 lossOfUtility.tc03.mean.CL.upper = lossOfUtility.tc03.mean + qAlpha.half * (1/
729   sqrt(nSims)) * sqrt(terminalUtility.tc03.sd.bench^2 + terminalUtility.tc03.

```

```

sd^2)
722 logReturn.tc03 = log(terminalWealth.tc03)
723 logReturn.tc03.mean = colMeans(logReturn.tc03)
724 volatility.tc03 = sdLogReturn.tc03 * sqrt(nTimePoints)
725 volatility.tc03.mean = colMeans(volatility.tc03)
726 excessReturn.tc03 = logReturn.tc03 - rent
727 sharpeRatio.tc03 = excessReturn.tc03 / volatility.tc03
728 sharpeRatio.tc03.mean = colMeans(sharpeRatio.tc03)
729 sharpeRatio.tc03.sd = colSds(sharpeRatio.tc03)
730 sharpeRatio.tc03.mean.CL.lower = sharpeRatio.tc03.mean - qAlpha.half *
    sharpeRatio.tc03.sd / sqrt(nSims)
731 sharpeRatio.tc03.mean.CL.upper = sharpeRatio.tc03.mean + qAlpha.half *
    sharpeRatio.tc03.sd / sqrt(nSims)
732 volOfVol.tc03 = colSds(volatility.tc03)
733 correlation.tc03 = colCorrs(logReturn.tc03, volatility.tc03)
734
735 tab1.tc03 = matrix(NA,18,4)
736
737 for (i in 1:9) {
738   tab1.tc03[2*i-1,] = c(terminalWealth.tc03.mean.bench[i], transCost.tc03.mean.
739     bench[i], terminalUtility.tc03.mean.bench[i], 0)
740   tab1.tc03[2*i,] = c(terminalWealth.tc03.mean[i], transCost.tc03.mean[i],
741     terminalUtility.tc03.mean[i], lossOfUtility.tc03.mean[i])
742 }
743 tab1.tc03[,2] = tab1.tc03[,2] * 1e2
744 tab1.tc03[,4] = tab1.tc03[,4] * 1e2
745 tab1.tc03 = round(tab1.tc03,4)
746
747 for (i in 1:18) {
748   tab1.tc03[i,2] = paste(tab1.tc03[i,2], "\\e{\\text-2}", sep="")
749   tab1.tc03[i,4] = paste(tab1.tc03[i,4], "\\e{\\text-2}", sep="")
750 }
751 for (i in seq(1,17,2)) { tab1.tc03[i,4] = "-" }
752
753 printex(tab1.tc03)
754
755 tab2.tc03 = matrix(NA,18,5)
756
757 for (i in 1:9) {
758   tab2.tc03[2*i-1,] = c(logReturn.tc03.mean.bench[i], volatility.tc03.mean.bench[
759     i], sharpeRatio.tc03.mean.bench[i], volOfVol.tc03.bench[i], correlation.tc03.
760     bench[i])
761   tab2.tc03[2*i,] = c(logReturn.tc03.mean[i], volatility.tc03.mean[i], sharpeRatio
762     .tc03.mean[i], volOfVol.tc03[i], correlation.tc03[i])
763 }
764 tab2.tc03[,1] = tab2.tc03[,1] * 1e2
765 tab2.tc03[,4] = tab2.tc03[,4] * 1e3
766 tab2.tc03 = round(tab2.tc03,4)
767
768 for (i in 1:18) {
769   tab2.tc03[i,1] = paste(tab2.tc03[i,1], "\\e{\\text-2}", sep="")
770   tab2.tc03[i,4] = paste(tab2.tc03[i,4], "\\e{\\text-3}", sep="")
771 }
772 printex(tab2.tc03)
773
774 scalar = 1e2
775 x.labels = strategyNames
776 x.title = "Rebalancing strategy"
777 y.title = expression(paste("Mean loss of utility",phantom(0) %*% 10^2))
778 x.ticks = 1:9

```

```

778 y.range = range(c(lossOfUtility.tc03.mean.CL.lower*scalar,lossOfUtility.tc03.
      mean.CL.upper*scalar))
779 niceplot(lossOfUtility.tc03.mean*scalar,xLabels=x.labels,xTitle=x.title,yTitle=y
      .title,y.addCustom=.2,figsPerPage=4,ylim=y.range)
780 abline(h=0,col="darkgray",lty=3)
781 abline(v=x.ticks,col="darkgray",lty=3)
782 nicelines(lossOfUtility.tc03.mean.CL.lower*scalar,lty=2)
783 nicelines(lossOfUtility.tc03.mean.CL.upper*scalar,lty=2)
784 legendText = c(expression(paste("(d)",lambda*=".03")))
785 nicelegend("left",legendText,bty="n",bg="white",cex=.7)
786 savePlot("images/lossOfUtility_stochVol_tc03",type="eps")
787
788 y.title = "Mean Sharpe ratio"
789 y.range = range(c(sharpeRatio.tc03.mean.CL.lower,sharpeRatio.tc03.mean.CL.upper)
      )
790 niceplot(sharpeRatio.tc03.mean,xLabels=x.labels,xTitle=x.title,yTitle=y.title,
      figsPerPage=4,ylim=y.range)
791 abline(v=x.ticks,col="darkgray",lty=3)
792 nicelines(sharpeRatio.tc03.mean.CL.lower,lty=2)
793 nicelines(sharpeRatio.tc03.mean.CL.upper,lty=2)
794 legendText = c(expression(paste("(d)",lambda*=".03")))
795 nicelegend("topleft",legendText,bty="n",bg="white",cex=.7)
796 savePlot("images/sharpeRatio_stochVol_tc03",type="eps")
797
798 # Plotting transaction cost histograms, lambda = .01
799
800 graphics.off()
801
802 x.title = "Total transaction cost"
803 y.title = "Frequency"
804 breaksLength = 70
805
806 # Hourly rebalancings
807 dataSet1 = transCost.tc01[,1]
808 dataSet2 = transCost.tc01.bench[,1]
809 x.range = range(c(dataSet1,dataSet2))
810 x.min = min(x.range)
811 x.max = max(x.range)
812 res = seq(x.min,x.max,length=breaksLength)
813 histObject1 = hist(dataSet1,breaks=res,plot=F)
814 histObject2 = hist(dataSet2,breaks=res,plot=F)
815 y.lim = range(c(histObject1$counts,histObject2$counts)) * 1.3
816 nicehist(dataSet1,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
817 legendText = c(expression(paste("(a)",lambda*=".01"),"T.c. strategy : Preceding
      ","Reb. strategy : Hourly"))
818 nicelegend("topleft",legendText,bty="n",cex=.7)
819 addHist(dataSet2,breaks=res,density=30)
820
821 # Daily rebalancings
822 dataSet1 = transCost.tc01[,3]
823 dataSet2 = transCost.tc01.bench[,3]
824 x.range = range(c(dataSet1,dataSet2))
825 x.min = min(x.range)
826 x.max = max(x.range)
827 res = seq(x.min,x.max,length=breaksLength)
828 histObject1 = hist(dataSet1,breaks=res,plot=F)
829 histObject2 = hist(dataSet2,breaks=res,plot=F)
830 y.lim = range(c(histObject1$counts,histObject2$counts)) * 1.3
831 nicehist(dataSet1,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,nCol=2,ylim
      =y.lim,breaks=res)
832 legendText = c(expression(paste("(b)",lambda*=".01"),"T.c. strategy : Preceding
      ","Reb. strategy : Daily"))
833 nicelegend("topleft",legendText,bty="n",cex=.7)
834 addHist(dataSet2,breaks=res,density=30)
835

```

```

836 savePlot("images/hist_transCost01_stochVol_HourlyDaily",type="eps")
837
838 # 'Every 3rd day' rebalancings
839 dataSet1 = transCost.tc01[,4]
840 dataSet2 = transCost.tc01.bench[,4]
841 x.range = range(c(dataSet1, dataSet2))
842 x.min = min(x.range)
843 x.max = max(x.range)
844 res = seq(x.min, x.max, length=breaksLength)
845 histObject1 = hist(dataSet1, breaks=res, plot=F)
846 histObject2 = hist(dataSet2, breaks=res, plot=F)
847 y.lim = range(c(histObject1$counts, histObject2$counts)) * 1.3
848 nicehist(dataSet1, xTitle=x.title, yTitle=y.title, nCol=2, ylim=y.lim, breaks=res)
849 legendText = c(expression(paste("(c)", lambda*=".01")), "T.c. strategy : Preceding
      ", "Reb. strategy : Ev. 3rd day"))
850 nicelegend("topleft", legendText, bty="n", cex=.7)
851 addHist(dataSet2, breaks=res, density=30)
852
853 # 'Every 12th day' rebalancings
854 dataSet1 = transCost.tc01[,5]
855 dataSet2 = transCost.tc01.bench[,5]
856 x.range = range(c(dataSet1, dataSet2))
857 x.min = min(x.range)
858 x.max = max(x.range)
859 res = seq(x.min, x.max, length=breaksLength)
860 histObject1 = hist(dataSet1, breaks=res, plot=F)
861 histObject2 = hist(dataSet2, breaks=res, plot=F)
862 y.lim = range(c(histObject1$counts, histObject2$counts)) * 1.3
863 nicehist(dataSet1, xTitle=x.title, yTitle=y.title, multiPlot=T, newDev=F, nCol=2, ylim
      =y.lim, breaks=res)
864 legendText = c(expression(paste("(d)", lambda*=".01")), "T.c. strategy : Preceding
      ", "Reb. strategy : Ev. 12th day"))
865 nicelegend("topleft", legendText, bty="n", cex=.7)
866 addHist(dataSet2, breaks=res, density=30)
867
868 savePlot("images/hist_transCost01_stochVol_3rd12th", type="eps")
869
870 # Monthly rebalancings
871 dataSet1 = transCost.tc01[,6]
872 dataSet2 = transCost.tc01.bench[,6]
873 x.range = range(c(dataSet1, dataSet2))
874 x.min = min(x.range)
875 x.max = max(x.range)
876 res = seq(x.min, x.max, length=breaksLength)
877 histObject1 = hist(dataSet1, breaks=res, plot=F)
878 histObject2 = hist(dataSet2, breaks=res, plot=F)
879 y.lim = range(c(histObject1$counts, histObject2$counts)) * 1.3
880 nicehist(dataSet1, xTitle=x.title, yTitle=y.title, nCol=2, ylim=y.lim, breaks=res)
881 legendText = c(expression(paste("(e)", lambda*=".01")), "T.c. strategy : Preceding
      ", "Reb. strategy : Monthly"))
882 nicelegend("topleft", legendText, bty="n", cex=.7)
883 addHist(dataSet2, breaks=res, density=30)
884
885 # Bimonthly rebalancings
886 dataSet1 = transCost.tc01[,7]
887 dataSet2 = transCost.tc01.bench[,7]
888 x.range = range(c(dataSet1, dataSet2))
889 x.min = min(x.range)
890 x.max = max(x.range)
891 res = seq(x.min, x.max, length=breaksLength)
892 histObject1 = hist(dataSet1, breaks=res, plot=F)
893 histObject2 = hist(dataSet2, breaks=res, plot=F)
894 y.lim = range(c(histObject1$counts, histObject2$counts)) * 1.3
895 nicehist(dataSet1, xTitle=x.title, yTitle=y.title, multiPlot=T, newDev=F, nCol=2, ylim
      =y.lim, breaks=res)

```

```

896 legendText = c(expression(paste("(f)", lambda*=".01"), "T.c. strategy : Preceding
      ", "Reb. strategy : Bimonthly"))
897 nicelegend("topleft", legendText, bty="n", cex=.7)
898 addHist(dataSet2, breaks=res, density=30)
899
900 savePlot("images/hist_transCost01_stochVol_MonthlyBi", type="eps")
901
902 # Semiannual rebalancings
903 dataSet1 = transCost.tc01[,8]
904 dataSet2 = transCost.tc01.bench[,8]
905 x.range = range(c(dataSet1, dataSet2))
906 x.min = min(x.range)
907 x.max = max(x.range)
908 res = seq(x.min, x.max, length=breaksLength)
909 histObject1 = hist(dataSet1, breaks=res, plot=F)
910 histObject2 = hist(dataSet2, breaks=res, plot=F)
911 y.lim = range(c(histObject1$counts, histObject2$counts)) * 1.3
912 nicehist(dataSet1, xTitle=x.title, yTitle=y.title, nCol=2, ylim=y.lim, breaks=res)
913 legendText = c(expression(paste("(e)", lambda*=".01"), "T.c. strategy : Preceding
      ", "Reb. strategy : Semiannual"))
914 nicelegend("topleft", legendText, bty="n", cex=.7)
915 addHist(dataSet2, breaks=res, density=30)
916
917 # Annual rebalancings
918 dataSet1 = transCost.tc01[,9]
919 dataSet2 = transCost.tc01.bench[,9]
920 x.range = range(c(dataSet1, dataSet2))
921 x.min = min(x.range)
922 x.max = max(x.range)
923 res = seq(x.min, 2*x.max, length=breaksLength)
924 histObject1 = hist(dataSet1, breaks=res, plot=F)
925 histObject2 = hist(dataSet2, breaks=res, plot=F)
926 y.lim = range(c(histObject1$counts, histObject2$counts)) * 1.3
927 nicehist(dataSet1, xTitle=x.title, yTitle=y.title, multiPlot=T, newDev=F, nCol=2, ylim
      =y.lim, breaks=res)
928 legendText = c(expression(paste("(f)", lambda*=".01"), "T.c. strategy : Preceding
      ", "Reb. strategy : Annual"))
929 nicelegend("topleft", legendText, bty="n", cex=.7)
930 addObj = addHist(dataSet2, breaks=res, density=30)
931
932 savePlot("images/hist_transCost01_stochVol_SemiAnnual", type="eps")
933
934 #
935 # Plotting transaction cost histograms, lambda = .02
936 #
937
938 # Hourly rebalancings
939 dataSet1 = transCost.tc02[,1]
940 dataSet2 = transCost.tc02.bench[,1]
941 x.range = range(c(dataSet1, dataSet2))
942 x.min = min(x.range)
943 x.max = max(x.range)
944 res = seq(x.min, x.max, length=breaksLength)
945 histObject1 = hist(dataSet1, breaks=res, plot=F)
946 histObject2 = hist(dataSet2, breaks=res, plot=F)
947 y.lim = range(c(histObject1$counts, histObject2$counts)) * 1.3
948 nicehist(dataSet1, xTitle=x.title, yTitle=y.title, nCol=2, ylim=y.lim, breaks=res)
949 legendText = c(expression(paste("(a)", lambda*=".02"), "T.c. strategy : Preceding
      ", "Reb. strategy : Hourly"))
950 nicelegend("topleft", legendText, bty="n", cex=.7)
951 addHist(dataSet2, breaks=res, density=30)
952
953 # Daily rebalancings
954 dataSet1 = transCost.tc02[,3]
955 dataSet2 = transCost.tc02.bench[,3]

```

```

956 x.range = range(c(dataSet1 , dataSet2))
957 x.min = min(x.range)
958 x.max = max(x.range)
959 res = seq(x.min,x.max,length=breaksLength)
960 histObject1 = hist (dataSet1 , breaks=res , plot=F)
961 histObject2 = hist (dataSet2 , breaks=res , plot=F)
962 y.lim = range(c(histObject1$counts , histObject2$counts)) * 1.3
963 nicehist (dataSet1 , xTitle=x.title , yTitle=y.title , multiPlot=T , newDev=F , nCol=2 , ylim
=y.lim , breaks=res)
964 legendText = c(expression (paste (" (b) " , lambda*" =.02" ) , "T.c. strategy : Preceding
" , "Reb. strategy : Daily" ))
965 nicelegend (" topleft " , legendText , bty="n" , cex=.7)
966 addHist (dataSet2 , breaks=res , density=30)
967
968 savePlot (" images/hist_transCost02_stochVol_HourlyDaily " , type="eps" )
969
970 # 'Every 3rd day' rebalancings
971 dataSet1 = transCost.tc02 [,4]
972 dataSet2 = transCost.tc02.bench [,4]
973 x.range = range(c(dataSet1 , dataSet2))
974 x.min = min(x.range)
975 x.max = max(x.range)
976 res = seq(x.min,x.max,length=breaksLength)
977 histObject1 = hist (dataSet1 , breaks=res , plot=F)
978 histObject2 = hist (dataSet2 , breaks=res , plot=F)
979 y.lim = range(c(histObject1$counts , histObject2$counts)) * 1.3
980 nicehist (dataSet1 , xTitle=x.title , yTitle=y.title , nCol=2 , ylim=y.lim , breaks=res)
981 legendText = c(expression (paste (" (c) " , lambda*" =.02" ) , "T.c. strategy : Preceding
" , "Reb. strategy : Ev. 3rd day" ))
982 nicelegend (" topleft " , legendText , bty="n" , cex=.7)
983 addHist (dataSet2 , breaks=res , density=30)
984
985 # 'Every 12th day' rebalancings
986 dataSet1 = transCost.tc02 [,5]
987 dataSet2 = transCost.tc02.bench [,5]
988 x.range = range(c(dataSet1 , dataSet2))
989 x.min = min(x.range)
990 x.max = max(x.range)
991 res = seq(x.min,x.max,length=breaksLength)
992 histObject1 = hist (dataSet1 , breaks=res , plot=F)
993 histObject2 = hist (dataSet2 , breaks=res , plot=F)
994 y.lim = range(c(histObject1$counts , histObject2$counts)) * 1.3
995 nicehist (dataSet1 , xTitle=x.title , yTitle=y.title , multiPlot=T , newDev=F , nCol=2 , ylim
=y.lim , breaks=res)
996 legendText = c(expression (paste (" (d) " , lambda*" =.02" ) , "T.c. strategy : Preceding
" , "Reb. strategy : Ev. 12th day" ))
997 nicelegend (" topleft " , legendText , bty="n" , cex=.7)
998 addHist (dataSet2 , breaks=res , density=30)
999
1000 savePlot (" images/hist_transCost02_stochVol_3rd12th " , type="eps" )
1001
1002 # Monthly rebalancings
1003 dataSet1 = transCost.tc02 [,6]
1004 dataSet2 = transCost.tc02.bench [,6]
1005 x.range = range(c(dataSet1 , dataSet2))
1006 x.min = min(x.range)
1007 x.max = max(x.range)
1008 res = seq(x.min,x.max,length=breaksLength)
1009 histObject1 = hist (dataSet1 , breaks=res , plot=F)
1010 histObject2 = hist (dataSet2 , breaks=res , plot=F)
1011 y.lim = range(c(histObject1$counts , histObject2$counts)) * 1.3
1012 nicehist (dataSet1 , xTitle=x.title , yTitle=y.title , nCol=2 , ylim=y.lim , breaks=res)
1013 legendText = c(expression (paste (" (e) " , lambda*" =.02" ) , "T.c. strategy : Preceding
" , "Reb. strategy : Monthly" ))
1014 nicelegend (" topleft " , legendText , bty="n" , cex=.7)

```

```

1015 addHist(dataSet2, breaks=res, density=30)
1016
1017 # Bimonthly rebalancings
1018 dataSet1 = transCost.tc02[,7]
1019 dataSet2 = transCost.tc02.bench[,7]
1020 x.range = range(c(dataSet1, dataSet2))
1021 x.min = min(x.range)
1022 x.max = max(x.range)
1023 res = seq(x.min, x.max, length=breaksLength)
1024 histObject1 = hist(dataSet1, breaks=res, plot=F)
1025 histObject2 = hist(dataSet2, breaks=res, plot=F)
1026 y.lim = range(c(histObject1$counts, histObject2$counts)) * 1.3
1027 nicehist(dataSet1, xTitle=x.title, yTitle=y.title, multiPlot=T, newDev=F, nCol=2, ylim
=y.lim, breaks=res)
1028 legendText = c(expression(paste("(f)", lambda*=".02")), "T.c. strategy : Preceding
", "Reb. strategy : Bimonthly"))
1029 nicelegend("topleft", legendText, bty="n", cex=.7)
1030 addHist(dataSet2, breaks=res, density=30)
1031
1032 savePlot("images/hist_transCost02_stochVol_MonthlyBi", type="eps")
1033
1034 # Semiannual rebalancings
1035 dataSet1 = transCost.tc02[,8]
1036 dataSet2 = transCost.tc02.bench[,8]
1037 x.range = range(c(dataSet1, dataSet2))
1038 x.min = min(x.range)
1039 x.max = max(x.range)
1040 res = seq(x.min, x.max, length=breaksLength)
1041 histObject1 = hist(dataSet1, breaks=res, plot=F)
1042 histObject2 = hist(dataSet2, breaks=res, plot=F)
1043 y.lim = range(c(histObject1$counts, histObject2$counts)) * 1.3
1044 nicehist(dataSet1, xTitle=x.title, yTitle=y.title, nCol=2, ylim=y.lim, breaks=res)
1045 legendText = c(expression(paste("(e)", lambda*=".02")), "T.c. strategy : Preceding
", "Reb. strategy : Semiannual"))
1046 nicelegend("topleft", legendText, bty="n", cex=.7)
1047 addHist(dataSet2, breaks=res, density=30)
1048
1049 # Annual rebalancings
1050 dataSet1 = transCost.tc02[,9]
1051 dataSet2 = transCost.tc02.bench[,9]
1052 x.range = range(c(dataSet1, dataSet2))
1053 x.min = min(x.range)
1054 x.max = max(x.range)
1055 res = seq(x.min, 2*x.max, length=breaksLength)
1056 histObject1 = hist(dataSet1, breaks=res, plot=F)
1057 histObject2 = hist(dataSet2, breaks=res, plot=F)
1058 y.lim = range(c(histObject1$counts, histObject2$counts)) * 1.3
1059 nicehist(dataSet1, xTitle=x.title, yTitle=y.title, multiPlot=T, newDev=F, nCol=2, ylim
=y.lim, breaks=res)
1060 legendText = c(expression(paste("(f)", lambda*=".02")), "T.c. strategy : Preceding
", "Reb. strategy : Annual"))
1061 nicelegend("topleft", legendText, bty="n", cex=.7)
1062 addObj = addHist(dataSet2, breaks=res, density=30)
1063
1064 savePlot("images/hist_transCost02_stochVol_SemiAnnual", type="eps")
1065
1066 #
1067 # Plotting transaction cost histograms, lambda = .03
1068 #
1069
1070 # Hourly rebalancings
1071 dataSet1 = transCost.tc03[,1]
1072 dataSet2 = transCost.tc03.bench[,1]
1073 x.range = range(c(dataSet1, dataSet2))
1074 x.min = min(x.range)

```



```

1075 x.max = max(x.range)
1076 res = seq(x.min,x.max,length=breaksLength)
1077 histObject1 = hist(dataSet1,breaks=res,plot=F)
1078 histObject2 = hist(dataSet2,breaks=res,plot=F)
1079 y.lim = range(c(histObject1$counts,histObject2$counts)) * 1.3
1080 nicehist(dataSet1,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
1081 legendText = c(expression(paste("a"),lambda*=".03"),"T.c. strategy : Preceding
", "Reb. strategy : Hourly"))
1082 nicelegend("topleft",legendText,bty="n",cex=.7)
1083 addHist(dataSet2,breaks=res,density=30)
1084
1085 # Daily rebalancings
1086 dataSet1 = transCost.tc03[,3]
1087 dataSet2 = transCost.tc03.bench[,3]
1088 x.range = range(c(dataSet1,dataSet2))
1089 x.min = min(x.range)
1090 x.max = max(x.range)
1091 res = seq(x.min,x.max,length=breaksLength)
1092 histObject1 = hist(dataSet1,breaks=res,plot=F)
1093 histObject2 = hist(dataSet2,breaks=res,plot=F)
1094 y.lim = range(c(histObject1$counts,histObject2$counts)) * 1.3
1095 nicehist(dataSet1,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,nCol=2,ylim
=y.lim,breaks=res)
1096 legendText = c(expression(paste("b"),lambda*=".03"),"T.c. strategy : Preceding
", "Reb. strategy : Daily"))
1097 nicelegend("topleft",legendText,bty="n",cex=.7)
1098 addHist(dataSet2,breaks=res,density=30)
1099
1100 savePlot("images/hist_transCost03_stochVol_HourlyDaily",type="eps")
1101
1102 # 'Every 3rd day' rebalancings
1103 dataSet1 = transCost.tc03[,4]
1104 dataSet2 = transCost.tc03.bench[,4]
1105 x.range = range(c(dataSet1,dataSet2))
1106 x.min = min(x.range)
1107 x.max = max(x.range)
1108 res = seq(x.min,x.max,length=breaksLength)
1109 histObject1 = hist(dataSet1,breaks=res,plot=F)
1110 histObject2 = hist(dataSet2,breaks=res,plot=F)
1111 y.lim = range(c(histObject1$counts,histObject2$counts)) * 1.3
1112 nicehist(dataSet1,xTitle=x.title,yTitle=y.title,nCol=2,ylim=y.lim,breaks=res)
1113 legendText = c(expression(paste("c"),lambda*=".03"),"T.c. strategy : Preceding
", "Reb. strategy : Ev. 3rd day"))
1114 nicelegend("topleft",legendText,bty="n",cex=.7)
1115 addHist(dataSet2,breaks=res,density=30)
1116
1117 # 'Every 12th day' rebalancings
1118 dataSet1 = transCost.tc03[,5]
1119 dataSet2 = transCost.tc03.bench[,5]
1120 x.range = range(c(dataSet1,dataSet2))
1121 x.min = min(x.range)
1122 x.max = max(x.range)
1123 res = seq(x.min,x.max,length=breaksLength)
1124 histObject1 = hist(dataSet1,breaks=res,plot=F)
1125 histObject2 = hist(dataSet2,breaks=res,plot=F)
1126 y.lim = range(c(histObject1$counts,histObject2$counts)) * 1.3
1127 nicehist(dataSet1,xTitle=x.title,yTitle=y.title,multiPlot=T,newDev=F,nCol=2,ylim
=y.lim,breaks=res)
1128 legendText = c(expression(paste("d"),lambda*=".03"),"T.c. strategy : Preceding
", "Reb. strategy : Ev. 12th day"))
1129 nicelegend("topleft",legendText,bty="n",cex=.7)
1130 addHist(dataSet2,breaks=res,density=30)
1131
1132 savePlot("images/hist_transCost03_stochVol_3rd12th",type="eps")
1133

```

```

1134 # Monthly rebalancings
1135 dataSet1 = transCost.tc03[,6]
1136 dataSet2 = transCost.tc03.bench[,6]
1137 x.range = range(c(dataSet1, dataSet2))
1138 x.min = min(x.range)
1139 x.max = max(x.range)
1140 res = seq(x.min, x.max, length=breaksLength)
1141 histObject1 = hist(dataSet1, breaks=res, plot=F)
1142 histObject2 = hist(dataSet2, breaks=res, plot=F)
1143 y.lim = range(c(histObject1$counts, histObject2$counts)) * 1.3
1144 nicehist(dataSet1, xTitle=x.title, yTitle=y.title, nCol=2, ylim=y.lim, breaks=res)
1145 legendText = c(expression(paste("(e)", lambda*=".03")), "T.c. strategy : Preceding
", "Reb. strategy : Monthly"))
1146 nicelegend("topleft", legendText, bty="n", cex=.7)
1147 addHist(dataSet2, breaks=res, density=30)
1148
1149 # Bimonthly rebalancings
1150 dataSet1 = transCost.tc03[,7]
1151 dataSet2 = transCost.tc03.bench[,7]
1152 x.range = range(c(dataSet1, dataSet2))
1153 x.min = min(x.range)
1154 x.max = max(x.range)
1155 res = seq(x.min, x.max, length=breaksLength)
1156 histObject1 = hist(dataSet1, breaks=res, plot=F)
1157 histObject2 = hist(dataSet2, breaks=res, plot=F)
1158 y.lim = range(c(histObject1$counts, histObject2$counts)) * 1.3
1159 nicehist(dataSet1, xTitle=x.title, yTitle=y.title, multiPlot=T, newDev=F, nCol=2, ylim
=y.lim, breaks=res)
1160 legendText = c(expression(paste("(f)", lambda*=".03")), "T.c. strategy : Preceding
", "Reb. strategy : Bimonthly"))
1161 nicelegend("topleft", legendText, bty="n", cex=.7)
1162 addHist(dataSet2, breaks=res, density=30)
1163
1164 savePlot("images/hist_transCost03_stochVol_MonthlyBi", type="eps")
1165
1166 # Semiannual rebalancings
1167 dataSet1 = transCost.tc03[,8]
1168 dataSet2 = transCost.tc03.bench[,8]
1169 x.range = range(c(dataSet1, dataSet2))
1170 x.min = min(x.range)
1171 x.max = max(x.range)
1172 res = seq(x.min, x.max, length=breaksLength)
1173 histObject1 = hist(dataSet1, breaks=res, plot=F)
1174 histObject2 = hist(dataSet2, breaks=res, plot=F)
1175 y.lim = range(c(histObject1$counts, histObject2$counts)) * 1.3
1176 nicehist(dataSet1, xTitle=x.title, yTitle=y.title, nCol=2, ylim=y.lim, breaks=res)
1177 legendText = c(expression(paste("(e)", lambda*=".03")), "T.c. strategy : Preceding
", "Reb. strategy : Semiannual"))
1178 nicelegend("topleft", legendText, bty="n", cex=.7)
1179 addHist(dataSet2, breaks=res, density=30)
1180
1181 # Annual rebalancings
1182 dataSet1 = transCost.tc03[,9]
1183 dataSet2 = transCost.tc03.bench[,9]
1184 x.range = range(c(dataSet1, dataSet2))
1185 x.min = min(x.range)
1186 x.max = max(x.range)
1187 res = seq(x.min, 2*x.max, length=breaksLength)
1188 histObject1 = hist(dataSet1, breaks=res, plot=F)
1189 histObject2 = hist(dataSet2, breaks=res, plot=F)
1190 y.lim = range(c(histObject1$counts, histObject2$counts)) * 1.3
1191 nicehist(dataSet1, xTitle=x.title, yTitle=y.title, multiPlot=T, newDev=F, nCol=2, ylim
=y.lim, breaks=res)
1192 legendText = c(expression(paste("(f)", lambda*=".03")), "T.c. strategy : Preceding
", "Reb. strategy : Annual"))

```

```
1193 nicelegend(" topleft", legendText, bty="n", cex=.7)
1194 addObj = addHist(dataSet2, breaks=res, density=30)
1195
1196 savePlot(" images/hist_transCost03_stochVol_SemiAnnual", type="eps")
```



# Bibliography

- [1] Fred Espen Benth. *Matematisk finans*. 3rd ed. Universitetsforlaget, 2007.
- [2] *CIR process*. URL: [http://en.wikipedia.org/wiki/CIR\\_process](http://en.wikipedia.org/wiki/CIR_process).
- [3] John C. Cox, Jonathan E. Ingersoll Jr., and Stephen A. Ross. “A theory of the term structure of interest rates”. In: *Econometrica* 53 (Mar. 1985), pp. 385–408.
- [4] M. H. A. Davies and A. R. Norman. “Portfolio selection with transaction costs”. In: *Mathematics of Operations Research* 15 (Nov. 1990), pp. 676–713.
- [5] *Discretization*. URL: <http://en.wikipedia.org/wiki/Discretization>.
- [6] Jon Exley, Shyam Mehta, and Andrew Smith. “Mean reversion”. In: (June 2004).
- [7] *Geometric Brownian motion*. URL: [http://en.wikipedia.org/wiki/Geometric\\_Brownian\\_motion](http://en.wikipedia.org/wiki/Geometric_Brownian_motion).
- [8] David Harper. *An introduction to value at risk*. URL: <http://www.investopedia.com/articles/04/092904.asp>.
- [9] Steven Heston. “A closed-form solution for options with stochastic volatility with applications to bond and currency options”. In: *The review of financial studies* 6 (Summer 1993), pp. 327–343.
- [10] Alireza Javaheri. *Inside volatility arbitrage: the secrets of skewness*. 1st ed. John Wiley & sons, inc., 2005.
- [11] Philippe Jorion. *Value at risk*. 3rd ed. The McGraw-Hill companies, 2007.
- [12] Peter E. Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*. 1st ed. Springer-Verlag, 1992.
- [13] Athanasios Koulakiotis, Nicholas Papasyriopoulos, and Phil Molyneux. “More evidence on the relationship between stock price returns and volatility: A note”. In: *International research journal of finance and economics* 1 (Jan. 2006), pp. 21–28.
- [14] Bernt Øksendal. *Stochastic differential equations*. 6th ed. Springer-Verlag, 2007.
- [15] Robert C. Merton. “Lifetime portfolio selection under uncertainty: The continuous-time case”. In: *The Review of Economics and Statistics* 51 (Aug. 1969), pp. 247–257.

- [16] Nimalin Moodley. “The Heston model: a practical approach with Matlab code”. Honours project. University of the Witwatersrand, 2005.
- [17] William F. Sharpe. “The Sharpe ratio”. In: *The Journal of Portfolio Management* 21 (Fall 1994), pp. 49–58.
- [18] *Sharpe ratio*. URL: [http://en.wikipedia.org/wiki/Sharpe\\_ratio](http://en.wikipedia.org/wiki/Sharpe_ratio).
- [19] *Stochastic volatility*. URL: [http://en.wikipedia.org/wiki/Stochastic\\_volatility](http://en.wikipedia.org/wiki/Stochastic_volatility).