UNIVERSITY OF TROMSØ UIT

FACULTY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF MATHEMATICS AND STATISTICS

A threshold cointegration analysis of Norwegian interest rates

Berner Larsen

Sta-3900
Master's Thesis in Statistics
April, 2012

Acknowledgements

First and foremost I would like to thank my supervisors Associate Professor Edvin L. Bredrup and Associate Professor Sigrunn Holbek Sørbye at Department of Mathematics and Statistics, University of Tromsø.

Second, I would like to thank my colleague Associate Professor Svein Oskar Lauvsnes at Bodø Graduate School of Business, University of Nordland, for reading through some versions of this thesis and giving me valuable advice.

Also, I am grateful to University of Tromsø and Bodø Graduate School of Business, University of Nordland for adapting the study progression and the workload, respectively, such that it has been posssible to complete this Master's Degree program in Statistics on a part-time basis.

Finally, I would like to express my deepest gratitude to my wife Lillian for being patient all those weekends and evenings when I have been "married to my books".

Abstract

In this thesis we generalize the Hansen and Seo test in the R package **tsDyn**, which tests a linear cointegration model against a two-regime threshold cointegration model, to the case of three regimes in the alternative hypothesis. As the Lagrange Multiplier (LM) test statistic used in the Hansen and Seo test in **tsDyn** is different from the LM statistic described in Hansen and Seo (2002), we generalize both these LM statistics, and show that they are equal under certain conditions. The Hansen and Seo test uses the SupLM statistic which is the maximum of this LM statistic when the two thresholds vary over the set of all possible threshold values. The grid search algorithm, which is necessary when maximizing this LM statistic, is also extended to the case of three regimes, and it is rewritten such that if the cointegration value β is given, it really maximizes the LM statistic under the constraints specified by the user.

In our empirical studies we have examined thoroughly the bivariate time series consisting of the monthly NIBOR rates of the maturities tomorrow next and 12 months. When modeling this bivariate time series, we find strong evidence for a two-regime TVECM being superior to a linear VECM, and in our out-of-sample forecasting the two-regime SETAR model gives much better prediction of the cointegration relation than a linear AR model. When testing a two-regime SETAR model for the cointegration relation against a three-regime model, the two-regime model cannot be rejected at any reasonable significance level. In addition, we show how influential a few outliers may be by removing them from the time series and rerunning some of the statistical tests. Also, we have tested all the 66 possible pairs of Norwegian interest rates for cointegration, and we have tested the term spread of each pair for threshold effects, i.e., testing a linear model against a two-regime model, as well as testing a two-regime model against a three-regime model. We find a lot of cointegrated pairs, and we find evidence for a two-regime model in approximately 50 % of the cases, and evidence for a three-regime model in some cases in this univariate time series analysis.

At last, we simulate a bivariate time series with a three-regime threshold cointegration model as data generation process, and estimate a three-regime threshold cointegration model from this simulated time series. Thus, we illustrate that the thresholds which our version of the Hansen and Seo test detects as optimal, are close to the original thresholds used in the simulation. As expected, a linear model for this bivariate time series is strongly rejected, and there is strong evidence for a three-regime threshold model for the cointegration relation being superior to both a linear model and a two-regime threshold model .

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cointegration has since it was introduced in Granger (1981), attached much attention among economists because it is a tool for testing the existence of and finding stable long-run relationships between nonstationary variables. For example, time series of interest rates are often nonstationary, but the Expectations Hypothesis which states that a long-term interest rate is an average of expected future short-term rates plus a risk premium, implies that there exists a stable linear long-run relationship between the short-term and the long-term interest rate (Hall, Anderson, and Granger 1992). A lot of papers have investigated the relationship between short-term and long-term interest rates, see e.g., Modigliani and Shiller (1973), Engsted (1996), Campbell and Shiller (1991), Musti and D'Ecclesia (2008), Arize, Malindretos, and Obi (2002), and Buigut and Rao (2010). Some of the results support the Expectations Hypothesis, and some do not, so other theories than the Expectations Hypothesis have been presented to explain the term structure of interest rates, but it is generally accepted that interest rates of different maturities should not deviate too much from each other (Siklos and Wohar 1996). The first tests for cointegration were proposed in Engle and Granger (1987), while Johansen (1988) and Johansen and Juselius (1990) have developed a procedure to test for the number of cointegration relations, i.e., long-run relationships, between the variables, and to find these cointegration relations. In this thesis we consider only bivariate time series, so the number of cointegration relations is either 0 (i.e., no cointegration) or 1.

When modeling a bivariate time series consisting of two interest rates of different maturities, the long-run relationship is typically the term spread, i.e., the difference between the interest rates, or more generally, a linear combination of the interest rates with one coefficient normalized to 1, and the other coefficient nearby 1. If we in our model include an error correction term containing this long-run relationship, we achieve that at each time point adjustments are performed due to deviations from the long-run equilibrium, the larger deviations the larger adjustments. However, in economic applications it is often unrealistic that the adjustments should be done at each time point. For example, there may be transaction costs, so that arbitrage opportunities between two markets only arise when the price difference is large enough to imply net gains to traders (Clements and Galvão 2004). To take into account such nonlinear behavior, Balke and Fomby (1997) introduced the threshold cointegration model, which allows the adjustment to be made only when the deviation from the long-run equilibrium is larger than an upper threshold and/or smaller than a lower threshold. Stigler (2011) gives both an overview of the field threshold cointegration and a description of how such a data analysis may be conducted by using the R package **tsDyn**. With this paper as a starting point, we will analyse NIBOR rates, downloaded from `http://www.norges-bank.no/en/price-stability/interest-rates/`, by using threshold cointegration. We will also analyse the term spread by using nonlinear autoregressive time series models described in Di Narzo, Aznarte, and Stigler (2011).

When a threshold cointegration model is estimated, it is of crucial interest to test whether this nonlinear model is superior to a linear cointegration model. Hansen and Seo (2002) proposed a test which tests a linear cointegration model against a two-regime threshold cointegration model, and this test is implemented in the R package **tsDyn**. In Hansen and Seo (2002) and Seo (2003) monthly U.S. Treasury bond rates are modeled by using two-regime and three-regime threshold cointegration models, respectively. We downloaded these U.S. interest rates from St. Louis Federal Reserve Bank at `http://research.stlouisfed.org/fred2/`, but we were not able to reproduce the results in Seo (2003). Therefore, we have examined the algorithm of the Hansen and Seo test in the package **tsDyn** thoroughly.

Our main contribution is the generalization of the Hansen and Seo test in the R package **tsDyn** to the case of three regimes in the alternative hypothesis. As the Lagrange Multiplier (LM) test statistic used in the Hansen and Seo test in **tsDyn** is different from the LM statistic described in Hansen and Seo (2002), we generalize both these LM statistics, and show that they are equal under certain conditions. The Hansen and Seo test uses the **SupLM** statistic which is the maximum of this LM statistic when the two thresholds γ_1 and γ_2 vary over the set of all possible threshold values. However, the function $\mathbf{LM}(\gamma_1, \gamma_2)$ is a highly irregular function such that we have to perform a grid search when maximizing this function. The global maximum of a function under explicitly given constraints is unique, i.e., the maximum value is unique, but there may be more than one point which give this maximum value. However, neither the implementation of the Hansen and Seo test used in Seo (2003) nor the implementation in the package **tsDyn** gives the user full control over the constraints used when maximizing $\mathbf{LM}(\gamma_1, \gamma_2)$, which may explain why we did not succeed in reproducing the results in Seo (2003). Therefore, we have made a new algorithm for the grid search in Chapter 3, which covers both the case of two regimes and the case of three regimes in the alternative hypothesis. In the case of three regimes, the algorithm is quadratic in the number of possible threshold values, and hence very time consuming as the P-value of the test statistic is estimated by using bootstrapping. Though, it is preferable with an algorithm which maximizes correctly under the given constraints.

In our empirical studies we have examined thoroughly the bivariate time series consisting of the monthly NIBOR rates of the maturities tomorrow next and 12 months, which was the first pair of Norwegian interest rates we found where a two-regime threshold model is significantly better than a linear model. We analyse both this bivariate time series by using functions for multivariate time series analysis, and the cointegration relation by using functions for univariate time series analysis. Our out-of-sample forecasting shows that a threshold model gives much better prediction of the cointegration relation than a linear model. In addition, we analyze the effect of removing 6 outliers from the tomorrow next rates and 2 outliers from the 12 months rates by using interpolation. Thus, we show how influential a few outliers may be.

As there exist NIBOR rates of 9 different maturities and interest rates on Norwegian government bonds of 3 different maturities, we may make 66 pairs of Norwegian interest rates. Due to the fact that our version of the Hansen and Seo test is very time-consuming, we are not able to test all these pairs of interest rates for threshold cointegration. Rather, we have tested all these pairs for cointegration, and we have tested the term spread of each pair for threshold effects, i.e., testing a linear model against a two-regime model, as well as testing a two-regime model against a three-regime model. We find a lot of cointegrated pairs, and we find evidence for a two-regime model in approximately 50 % of the cases, and evidence for a three-regime model in some cases in this univariate time series analysis. However, a threshold cointegration analysis of the corresponding bivariate time series is a topic for further research.

At last, we simulate a bivariate time series with a three-regime threshold cointegration model as data generation process, and estimate a three-regime threshold cointegration model from this simulated time series. Thus, we illustrate that the thresholds which our version of the Hansen and Seo test detects as optimal, are close to the original thresholds used in the simulation. As expected, a linear model for this bivariate time series is strongly rejected, and there is strong evidence for a three-regime threshold model for the cointegration relation being superior to both a linear model and a two-regime threshold model .

The rest of this thesis is organized as follows: In Chapter 2 we give an overview of all time series models and statistical tests used in this thesis. Chapter 3 describes our generalization of the Hansen and Seo test in the R package **tsDyn** to the case of three regimes in the alternative hypothesis. Chapter 4 contains our analysis of Norwegian interest rates. In Chapter 5 we simulate a time series which follows a three-regime vector error correction model, and we analyse this simulated time series by using the same tools as in Chapter 4. In Chapter 6 we summarize the results of this thesis. Appendix A and B contain the R code of the original version in **tsDyn** and our revised version of the Hansen and Seo test, respectively. Appendix C contains all the R code chunks which were run in Chapter 4 and 5 to perform the data analysis.

Chapter 2

The time series models and the statistical tests used in this thesis

This chapter contains all the time series models and the statistical tests used in later chapters of this thesis. All the materials in this chapter are found in the R documentation of the R functions mentioned in the text, and in Shumway and Stoffer (2006), Tsay (2010), Lütkepohl (2007), Pfaff (2008a) and Juselius (2006).

2.1 White noise processes

A univariate time series x_t is an ordered set of random variables $x_1, x_2, x_3, \dots$ where x_i is the value at time point i. Also, we will use the same notation x_t for a realization of this stochastic process. In applications, we usually know only one single realization of finite length T of the data-generating process, such that the task is to find a model for the data-generating process which fits well to the known data $x_1, \dots, x_T$.

A **white noise process** is a simple data-generating process, but it is important as more complicated time series models usually are defined by using a white noise process as error term. It is defined as follows:

Definition 2.1. A white noise process is a data-generating process ϵ_t such that $\mathrm{E}(\epsilon_t) = 0$, $\mathrm{E}(\epsilon_t^2) = \sigma^2$ for all t and $\mathrm{E}(\epsilon_t \epsilon_\tau) = 0$ for all $t \neq \tau$. If in addition $\epsilon_t \sim N(0, \sigma^2)$, we say that ϵ_t is **Gaussian white noise**. If we replace the condition of uncorrelatedness with the stronger assumption of independence, the process is said to be **independent white noise**.

2.2 Autoregressive moving average models

When modeling the term spread in Chapter 4, we will try an ARMA(p,q) model which is defined as follows:

Definition 2.2. The time series x_t is an **ARMA(p,q) process** if

$$x_t = c + \phi_1 x_{t-1} + \cdots + \phi_p x_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} \tag{2.1}$$

where c is a constant, p and q are nonnegative integers, $\phi_p \neq 0$, $\theta_q \neq 0$ and ϵ_t is a white noise process. The integer p is called the **AR order**, while the number q is the **MA order**.

Using the **lag operator** L defined by $Lx_t = x_{t-1}$, we may write the model as:

$$(1 - \phi_1 L - \cdots - \phi_p L^p) x_t = c + (1 + \theta_1 L + \cdots + \theta_q L^q)\epsilon_t.$$

The polynomial $\phi(L) = 1 - \phi_1 L - \cdots - \phi_p L^p$ is the **AR polynomial**, while the polynomial $\theta(L) = 1 + \theta_1 L + \theta_2 L^2 \cdots + \theta_q L^q$ is the **MA polynomial**. We assume that the AR and MA polynomials have no common factors; otherwise we may reduce the AR and MA orders by cancelling these factors.

We estimate an ARMA(p,q) model by using the function `arima` in the R package **stats**.

2.2.1 Model checking

When fitting ARMA(p,q) models it is important to check that the residual series behaves like a white noise series. We do this by plotting the sample autocorrelation function (ACF) and the sample partial autocorrelation function (PACF) of the residuals using the R functions `acf` and `pacf`. The AR-order p and MA-order q are sufficiently large if the sample ACF and PACF are well inside the 95 % confidence bounds of $\pm \frac{2}{\sqrt{T}}$. In addition, we run the Ljung-Box test to test whether the sample autocorrelations

$$\widehat{\rho_{\hat{\epsilon}}}(h) = \frac{\sum_{t=h+1}^{T} \hat{\epsilon}_t \hat{\epsilon}_{t-h}}{\sum\limits_{t=1}^{T} \hat{\epsilon}_t^2} \qquad \text{of the residuals } \hat{\epsilon}_t \text{ when } h = 1, \ldots, H, \tag{2.2}$$

collectively are large. The Ljung-Box statistic is defined by

$$Q = T(T+2) \sum_{h=1}^{H} \frac{(\widehat{\rho_{\hat{\epsilon}}}(h))^2}{T-h} \tag{2.3}$$

Under the null hypothesis H_0 of white noise residuals, Q is asymptotically distributed as χ^2_{H-p-q} when $T \to \infty$. Here, the number of degrees of freedom is decreased from H to $H - p - q$, to get a better approximation of the null hypothesis distribution in the case that $\hat{\epsilon}_t$, $t = 1, \ldots, T$ is the residual series of an ARMA(p,q) model. The R function `Box.test` computes Q and the P-value given the asymptotic distribution of Q under H_0. So, we reject the null hypothesis at level α if the P-value is less than or equal to α.

2.3 Unit root models

The ARMA(p,q) model above is stationary when all the roots of the AR polynomial $\phi(z)$ are outside the unit circle. In this case, we may write the model as

$$x_t = \mu + \sum_{i=0}^{\infty} \psi_i \epsilon_{t-i}$$

However, if at least one of the roots of $\phi(z)$ is inside the unit circle, the process is explosive, i.e., $x_t \to \infty$ rapidly when $t \to \infty$, so this case is not interesting. On the contrary, the case when $\phi(z)$ has the **unit root** $z = 1$ and all the other roots outside the unit circle, has many interesting applications in spite of the fact that the model is nonstationary in this particular case. Suppose that $\phi(L) = F(L)(1 - L)^d$ where all the roots of $F(L)$ are outside the unit circle, and let **the first order difference** be $\Delta x_t = (1 - L)x_t = x_t - x_{t-1}$. Then, the ARMA(p,q) model $\phi(L)x_t = \theta(L)\epsilon_t$ may be written as $F(L)\Delta^d x_t = \theta(L)\epsilon_t$ which shows that the time series $\Delta^d x_t$ is a stationary ARMA$(p-d, q)$ model. The following definitions are useful:

Definition 2.3. A time series with no deterministic component that has a stationary ARMA representation after being differenced d times is said to be **integrated of order *d***, which is denoted by $x_t \sim I(d)$.

Definition 2.4. A time series which is $I(1)$, is also called **difference-stationary**, as we get a stationary series when differencing once.

Definition 2.5. A time series x_t is **trend-stationary** if $x_t = \beta_0 + \beta_1 t + z_t$ where β_0 and β_1 are constants, and z_t is a stationary time series. If $\beta_1 = 0$, the time series x_t is said to be **level-stationary**.

By these definitions, we have extended the class of time series which may be analyzed, considerably from the class of stationary time series. This is important because a lot of the time series in applications are nonstationary. For example, many of the financial time series may be made stationary either by differencing one or more times, or by detrending, i.e., removing the deterministic trend such that the residuals constitute a stationary time series.

In applications, we usually know only a single realization of length T of the data-generating process, which means that we have to estimate the parameters. As we cannot draw a reliable conclusion about unit roots only by finding the roots of the estimated AR polynomial, we have to perform tests for unit roots. In Section 2.3.1 and 2.3.2 we will describe the ADF test and the KPSS test.

2.3.1 The ADF test

The augmentet Dickey-Fuller test (ADF test) is designed for the case when the time series possibly contains a linear trend $\beta_1 + \beta_2 t$, but we do not know a priori the values of β_1 and β_2. Hence, we have to estimate β_1 and β_2, test whether these parameters are significantly different from zero, and test for unit roots. The test procedure consists of test regressions with three different combinations of the deterministic component:

$$\Delta y_t = \beta_1 + \beta_2 t + \pi y_{t-1} + \sum_{j=1}^{k} \gamma_j \Delta y_{t-j} + u_t, \tag{2.4}$$

$$\Delta y_t = \beta_1 + \pi y_{t-1} + \sum_{j=1}^{k} \gamma_j \Delta y_{t-j} + u_t, \tag{2.5}$$

$$\Delta y_t = \pi y_{t-1} + \sum_{j=1}^{k} \gamma_j \Delta y_{t-j} + u_t. \tag{2.6}$$

Note that $\pi = 0$ in these equations is equivalent to the existence of a unit root. The function `ur.df` in the R package **urca** (Pfaff 2008a) performs the test regressions (2.4), (2.5) and (2.6) when the parameter `type` is equal to `"trend"`, `"drift"` and `"none"`, respectively. The number k of lagged differences in these regression equations may be selected according to Akaikes or Bayes information criteria, or may be set to a fixed number. The ADF tests for these three regression equations contain the following test statistics:

- τ_3: A "t statistic" for testing $H_0 : \pi = 0$ against $\pi < 0$ in Equation (2.4).
- ϕ_2: A likelihood ratio statistic for testing $H_0 : \beta_1 = \beta_2 = \pi = 0$ against H_0 not true in Equation (2.4).
- ϕ_3: A likelihood ratio statistic for testing $H_0 : \beta_2 = \pi = 0$ against H_0 not true in Equation (2.4).
- τ_2: A "t statistic" for testing $H_0 : \pi = 0$ against $\pi < 0$ in Equation (2.5).
- ϕ_1: A likelihood ratio statistic for testing $H_0 : \beta_1 = \pi = 0$ against H_0 not true in Equation (2.5).
- τ_1: A "t statistic" for testing $H_0 : \pi = 0$ against $\pi < 0$ in Equation (2.6).

These statistics are constructed in the same way as ordinary t and F statistics in regression analysis, but they do not follow the ordinary Student t and F distributions, so these distributions have to be estimated. Fortunately, the function `ur.df` reports both the values of the test statistics and their critical values at the 1%, 5% and 10% significance level.

The test procedure is as follows:

1. First, we estimate Equation (2.4) and test for the presence of a unit root by using the test statistic τ_3. If $H_0 : \pi = 0$ is rejected, we are finished and conclude that the time series y_t does not contain a unit root.
2. If not, we test for the presence of a trend by using the test statistic ϕ_3. If $H_0 : \beta_2 = \pi = 0$ is rejected, we test again for a unit root by using the standardized normal. If this test cannot be rejected, we are finished and conclude that the time series y_t contains a unit root with a non-zero trend. If this test is rejected, we are finished and conclude that the time series y_t does not contain a unit root.
3. If $H_0 : \beta_2 = \pi = 0$ cannot be rejected, we conclude that the time series y_t does

not contain a trend, we estimate Equation (2.5) (where the trend term is removed), and test for the presence of a unit root by using the test statistic τ_2. If $H_0 : \pi = 0$ is rejected, we are finished and conclude that the time series y_t does not contain a unit root.

4. If not, we test for the presence of a drift term by using the test statistic ϕ_1. If $H_0 : \beta_1 = \pi = 0$ is rejected, we test again for a unit root by using the standardized normal. If this test cannot be rejected, we are finished and conclude that the time series y_t contains a unit root with a non-zero drift. If this test is rejected, we are finished and conclude that the time series y_t does not contain a unit root.

5. If $H_0 : \beta_1 = \pi = 0$ cannot be rejected, we conclude that the time series y_t has neither a drift nor a trend term, we estimate Equation (2.6) (where the trend and drift term are removed) and test for the presence of a unit root by using the test statistic τ_1. If $H_0 : \pi = 0$ is rejected, we conclude that the time series y_t does not contain a unit root. If $H_0 : \pi = 0$ cannot be rejected, we are finished and conclude that the time series y_t contains a unit root without trend and drift. If $H_0 : \pi = 0$ is rejected, we are finished and conclude that the time series y_t does not contain a unit root.

If the result of the test procedure is that y_t contains a unit root, we know that y_t is $I(d)$ where $d \geq 1$. Therefore, we apply the test procedure on the series Δy_t. If the conclusion is that Δy_t does not contain a unit root, we conclude that y_t is $I(1)$. If not, we conclude that y_t is $I(d)$ where $d \geq 2$, and apply the test procedure on the series $\Delta^2 y_t$, and so on.

2.3.2 The KPSS test

In many applications we want to conclude that the time series contains a unit root. Then, it may be dangerous to solely rely on the ADF test which has a unit root process as null hypothesis. The Kwiatkowski-Phillips-Schmidt-Shin test (KPSS test) has on the other hand a stationary process as the null hypothesis and a unit root process as the alternative hypothesis. Thus, the probability of mistakenly concluding with a unit root process is fully controlled by the significance level.

In the KPSS test (Kwiatkowski et al. 1992) we assume that the time series y_t may be decomposed into a sum of a deterministic trend, a random walk and a stationary error:

$$y_t = \xi t + r_t + \epsilon_t$$

where ξ is a constant, r_t is a random walk defined by $r_t = r_{t-1} + u_t$ with u_t i.i.d $(0,\sigma_u^2)$ and r_0 fixed, and ϵ_t is the stationary error term. The null hypothesis is simply $\sigma_u^2 = 0$, which implies that the random walk degenerates to the constant r_0, such that y_t is trend-stationary under H_0 when $\xi \neq 0$, and level-stationary under H_0 when $\xi = 0$. The test statistic is constructed as follows: First, we regress y_t on a constant or on a constant and a trend, depending on whether we want to test level-stationarity or trend-stationarity. Let $\hat{\epsilon}_t, t = 1, \ldots, T$ be the residuals of this regression, and define the partial sums of these residuals as $S_t = \sum_{i=1}^{t} \hat{\epsilon}_i$, $t = 1, \ldots, T$. Then the test statistic for the KPSS test is

$$LM = \frac{\sum_{i=1}^{T} S_t^2}{\hat{\sigma}_\epsilon^2}$$

where the estimate $\hat{\sigma}_\epsilon^2$ of the error variance is defined by

$$\hat{\sigma}_\epsilon^2 = T^{-1} \sum_{t=1}^{T} \hat{\epsilon}_t^2 + 2T^{-1} \sum_{s=1}^{l} \left(1 - \frac{s}{l+1}\right) \sum_{t=s+1}^{T} \hat{\epsilon}_t \hat{\epsilon}_{t-s},$$

where the number l of lags used when computing the error variance, has to be specified. This test is implemented in the function ***ur.kpss*** in the R package **urca**. By letting the parameter ***lags="long"***, the number l is set to $l = \left[12\left(\frac{T}{100}\right)^{\frac{1}{4}}\right]$. The test statistics $\hat{\eta}_\mu$ and $\hat{\eta}_\tau$ for testing level-stationarity and trend-stationarity, respectively, are reported, together with the critical values for the significance levels 1%, 2.5%, 5% and 10%.

2.4 SETAR(p) models

Generally, we get a better approximation of a nonlinear function by using a piecewise linear function, rather than a global linear function. Similarly, we may get a better approximation of a nonlinear data-generation process by introducing a piecewise linear time series model. A simple nonlinear time series model is the self-exciting threshold autoregressive (SETAR) model which is defined as follows:

Definition 2.6. A time series x_t is said to follow a k-regime **SETAR(p) model** if it satisfies

$$x_t = c_j + \phi_{1j} x_{t-1} + \cdots + \phi_{pj} x_{t-p} + \epsilon_{tj}, \qquad \text{if} \qquad \gamma_{j-1} < x_{t-d-1} \leq \gamma_j, \tag{2.7}$$

where p is a nonnegative integer, k is a positive integer, $d \in \{0, 1, \ldots, (p-1)\}$, $j = 1, \ldots, k$ and γ_i are real numbers such that $-\infty = \gamma_0 < \gamma_1 < \cdots < \gamma_k = \infty$, and ϵ_{tj} is a white noise process for each $j = 1, \ldots, k$. The number p is the **AR order**, the number d is the **threshold delay**, i.e., the time delay of the threshold variable x_{t-d-1} compared with x_{t-1}, the integer k denotes the **regime number**, while the numbers $\gamma_1, \ldots, \gamma_{k-1}$ denote the $k-1$ **thresholds** which divide the threshold space into k regimes.

Note that, in each of the k regimes, the time series x_t is a linear AR(p) model. So, when $k = 1$, x_t is an AR(p) model, However, when $k \geq 2$, all these k linear models are different, and which one of these k models we actually use when computing x_t, is governed by the value of the threshold variable x_{t-d-1}. As a result, the SETAR(p) model is nonlinear when $k > 1$.

We estimate a SETAR(p) model by using the function ***setar*** in the R package **tsDyn**. Note that the definition of the SETAR model in **tsDyn** is a bit more general than the definition above which corresponds to always using the default value 1 of the parameters ***steps*** and ***d*** in the ***setar*** function, and never using the parameters ***mTh*** and ***thVar*** to define the threshold variable.

2.4.1 Testing for the number of regimes in SETAR models

When we have estimated a SETAR(p) model with $k > 1$, it is important to decide whether this nonlinear model is superior to a linear model (with only one regime). Hansen (1999) has proposed such a test which is implemented in the function ***setarTest*** in the R package **tsDyn**. By using this test, we are able to test a 1-regime model against a 2-regime model, test a 1-regime model against a 3-regime model, and test a 2-regime model against a 3-regime model. Let S_j be the minimum SSR when fitting a j-regime SETAR model to the data $y_1, y_2, \ldots, y_T$. Then, S_1 is the SSR from the estimation of the linear AR(p) model. If $j = 2$, the threshold value γ which minimizes the SSR, is computed by performing a grid search for γ, thus S_2 is the minimum value of SSR in this grid search. If $j = 3$, a new grid search for the second threshold value is performed conditional on the value of the first threshold, thus S_3 is the minimum value of SSR in this new grid search.

The test statistic for testing a j-regime SETAR model against a k-regime SETAR model where $k > j$ is given by

$$F_{jk} = T\frac{S_j - S_k}{S_k}.$$

However, the distribution of F_{jk} is nonstandard due to the presence of nuisance parameter(s) γ which are only defined under the alternative hypothesis. Therefore, the distribution of F_{jk} is estimated by bootstrapping, i.e., resampling of the residuals under the null hypothesis, estimating the threshold parameter(s) and computing F_{jk} for each bootstrap replication. The function ***setarTest*** returns the value of the test statistic F_{jk}, the bootstrap P-value and the critical values at the 90%, 95%, 97.5% and 99% level.

2.5 Vector autoregressive models

A good starting point in multivariate time series analysis is the vector autoregressive model of order p (VAR(p) model), which is defined as follows:

Definition 2.7. The K-dimensional time series y_t is said to follow a **VAR(p) model** if it satisfies

$$y_t = \mu + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + u_t \quad t = 1, \ldots, T \tag{2.8}$$

where $y_t = [y_{1t}, \ldots, y_{Kt}]'$ is a $(K \times 1)$ random vector, the A_i are fixed $(K \times K)$ coefficient matrices, μ is an intercept term, and u_t is a K-dimensional white noise process, i.e., $E(u_t) = 0$, $E(u_t u_s') = 0$ when $t \neq s$ and $E(u_t u_t') = \Sigma_u$ where Σ_u is nonsingular. The process y_t is said to be **stable** if its **reverse characteristic polynomial** has no roots in and on the unit circle, i.e.,

$$\det(I_K - A_1 z - \cdots - A_p z) \neq 0 \quad \text{for } |z| \leq 1. \tag{2.9}$$

We estimate a VAR(p) model by using the R function ***lineVar*** in the package **tsDyn** which does linear regression on Equation (2.8). When a model is estimated, it is of crucial importance to test whether the residuals obey the model's assumptions. So, we test for serial correlation, ARCH effects and nonnormality in the residuals.

2.5.1 The asymptotic Portmanteau test

Suppose that u_t, $t = 1, \dots, T$ are the residuals of a VAR(p) process defined by Equation (2.8). Then, the autocovariance matrices C_i and the autocorrelation matrices R_i of the residuals u_t are estimated as

$$C_i = \frac{1}{T} \sum_{t=i+1}^{T} u_t u'_{t-i}, \quad i = 0, 1, \dots, h < T$$
$$R_i = D^{-1} C_i D^{-1}, \quad i = 0, 1, \dots, h < T,$$

where D is the diagonal matrix whose diagonal elements are equal to the square root of the diagonal elements of C_0. The asymptotic Portmanteau test is a test for the overall significance of the residual autocorrelation up to lag h, i.e., it tests

$$H_0 : \boldsymbol{R}_h = (R_1, \dots, R_h) = 0 \quad \text{against} \quad H_1 : \boldsymbol{R}_h \neq 0.$$

The test statistic of the Portmanteau test is

$$Q_h = T \sum_{i=1}^{h} \operatorname{tr}(\widehat{C}_i' \widehat{C}_0^{-1} \widehat{C}_i \widehat{C}_0^{-1})$$

where $\widehat{C}_i = \frac{1}{T} \sum_{t=i+1}^{T} \hat{u}_t \hat{u}'_{t-i}, \quad i = 0, 1, \dots, h < T$ and $\hat{u}_t$, $t = 1, \dots, T$ are the estimated residuals.

For large T and h the test statistic Q_h is approximately distributed as $\chi^2(K^2(h-p)$. We perform the asymptotic Portmanteau test with $h = 16$ by using the R function `serial.test` in the R package **vars** with the default parameters `type="PT.asymptotic"` and `lags.pt=16`. Also, the P-value is included in the output from this function, such that the null hypothesis of no autocorrelation in the residuals is rejected if the P-value is less than the selected significance level.

2.5.2 The ARCH-LM test

We test for ARCH effects in the residuals by using the ARCH-LM test which is defined as follows: Consider the auxiliary model

$$\text{vech}(u_t u_t') = \beta_0 + B_1 \text{vech}(u_{t-1} u'_{t-1}) + \cdots + B_q \text{vech}(u_{t-q} u'_{t-q}) + \epsilon_t, \qquad t = q+1, \dots, T \quad (2.10)$$

where vech() is the column stacking operator for symmetric matrices which stacks the elements on and below the main diagonal, and ϵ_t is an error term. Note that when u_t is K-dimensional, β_0 and ϵ_t are $\frac{1}{2}K(K+1)$- dimensional vectors, while B_i, $i = 1, \dots, q$ are $\left(\frac{1}{2}K(K+1) \times \frac{1}{2}K(K+1)\right)$-dimensional matrices. The hypothesis in the ARCH-LM test are $H_0 : B_1 = \cdots = B_q = 0$ and $H_1 : B_1 \neq 0$ or $\cdots$ or $B_q \neq 0$. Let $\widehat{\Sigma}_{\text{vech}}$ be the residual covariance estimator of the model (2.10), i.e., $\widehat{\Sigma}_{\text{vech}} = \frac{1}{T-q} \sum_{t=q+1}^{T} \hat{\epsilon}_t \hat{\epsilon}_t'$ where $\hat{\epsilon}_t$, $t = 1, \dots, T$ are the estimated residuals in (2.10), and let $\widehat{\Sigma}_0$ be the residual covariance estimator of the model (2.10) under H_0, i.e., when $q = 0$. Then, the test statistic is defined as:

$$LM_{MARCH}(q) = \frac{TK(K+1)}{2} - T \operatorname{tr}\left(\widehat{\Sigma}_{\text{vech}} \widehat{\Sigma}_0^{-1}\right).$$

Under H_0, the statistic has an asymptotic $\chi^2(qK^2(K+1)^2/4)$-distribution. Also, we get the univariate ARCH-LM test by letting $K = 1$ in the above formulas. The regression model (2.10) reduces in this case to

$$u_t^2 = \beta_0 + B_1 u_{t-1}^2 + \cdots + B_q u_{t-q}^2 + \epsilon_t \quad \text{where } \beta_0, \epsilon_t, B_1, \ldots, B_q \text{ now are numbers.}$$

The function `arch.test` in the R package **vars** computes by default only the multivariate ARCH-LM test statistic for the multivariate time series u_t, but if we let the parameter `multivariate.only` be `FALSE`, it also computes the univariate ARCH-LM test statistic for each of the components of u_t. We use the default numbers of lags in the ARCH-LM tests, which are 5 and 16 in the univariate and multivariate case, respectively, but the number of lags may be changed by setting the parameters `lags.single` and `lags.multi`. The P-value is included in the output from `arch.test`, so the null hypothesis of no ARCH effect in the residuals is rejected if the P-value is less than the selected significance level.

2.5.3 Normality tests

The following normality tests are based on the third and fourth central moment, i.e., the skewness and kurtosis of the normal distribution. If $x \sim N(0,1)$, we know that $E(x^3) = 0$ and $E(x^4) = 3$. Let $\widehat{u}_t$ be the estimated residuals of a VAR(p) model based on a sample $y_1, \ldots, y_T$ with the presample values $y_{-p+1}, \ldots, y_0$, and suppose that u_t is zero mean white noise with nonsingular covariance matrix Σ_u which may be Cholesky decomposed as $\Sigma_u = PP'$. The residuals $\widehat{u}_t$ are standardized by letting $\widehat{w}_t = \widehat{P}^{-1}(\widehat{u}_t - \overline{\widehat{u}})$ where $\overline{\widehat{u}} = \frac{1}{T}\sum_{t=1}^{T}\widehat{u}_t$ and $\widehat{P}$ is a matrix satisfying $\widehat{P}\widehat{P}' = \frac{1}{T-Kp-1}\sum_{t=1}^{T}(\widehat{u}_t - \overline{\widehat{u}})(\widehat{u}_t' - \overline{\widehat{u}}')$ such that $\text{plim}(\widehat{P} - P) = 0$. By these definitions, $w_t \sim N_K(0, I_K)$ if the residuals are Gaussian white noise. Let

$$\widehat{b}_1 = (\widehat{b}_{11}, \ldots, \widehat{b}_{K1})' \quad \text{with} \quad \widehat{b}_{k1} = \frac{1}{T}\sum_{t=1}^{T}\widehat{w}_{kt}^3, \quad k = 1, \ldots, K$$

$$\widehat{b}_2 = (\widehat{b}_{12}, \ldots, \widehat{b}_{K2})' \quad \text{with} \quad \widehat{b}_{k2} = \frac{1}{T}\sum_{t=1}^{T}\widehat{w}_{kt}^4, \quad k = 1, \ldots, K$$

Then, the test statistic $\lambda_s = T\widehat{b}_1\widehat{b}_1'/6$, which has asymptotical distribution $\chi^2(K)$, may be used to test skewness, i.e., to test $H_0 : E[w_{1t}^3, \ldots, w_{Kt}^3]' = 0$ against $H_1 : E[w_{1t}^3, \ldots, w_{Kt}^3]' \neq 0$. Similarly, the test statistic $\lambda_k = T(\widehat{b}_2 - \mathbf{3}_K)(\widehat{b}_2 - \mathbf{3}_K)'/24$, which has asymptotical distribution $\chi^2(K)$, may be used to test kurtosis, i.e., to test $H_0 : E[w_{1t}^4, \ldots, w_{Kt}^4]' = \mathbf{3}_K$ against $H_1 : E[w_{1t}^4, \ldots, w_{Kt}^4]' \neq \mathbf{3}_K$. Finally, the test statistic $\lambda_{sk} = \lambda_s + \lambda_k$, which has asymptotical distribution $\chi^2(2K)$, may be used for a joint test of the null hypotheses for skewness and kurtosis (the Jarque-Bera normality test).

The function `normality.test` in the R package **vars** performs these three multivariate tests. If the parameter `multivariate.only` is `FALSE`, also the univariate Jarque-Bera tests for each of the K components are performed. For each of these tests, both the value of the test statistic and the P-value are reported, so we reject the null hypothesis if the P-value is less than the selected significance level.

2.6 Vector error correction models

Suppose that each component of a K-dimensional time-series y_t is $I(1)$. Then, Equation (2.8) on page 11 is not an appropriate formulation of its model because the terms $y_t, y_{t-1}, \ldots, y_{t-p}$ all are nonstationary. However, by substituting

$$A_1 = I_K + \mathbf{\Gamma}_1$$
$$A_i = \mathbf{\Gamma}_i - \mathbf{\Gamma}_{i-1} \quad i = 1, \ldots, p-1$$
$$A_p = -\mathbf{\Gamma}_{p-1}$$

in Equation (2.8), rearranging the terms and using that $\Delta y_i = y_i - y_{i-1}$ for all i, we may rewrite this equation as

$$\Delta y_t = \mu + \mathbf{\Gamma}_1 \Delta y_{t-1} + \mathbf{\Gamma}_2 \Delta y_{t-2} + \cdots + \mathbf{\Gamma}_{p-1} \Delta y_{t-p+1} + u_t. \tag{2.11}$$

Naturally, both equations describe the same model, but we prefer to use Equation (2.11) when y_t is $I(1)$, because each term in Equation (2.11) is stationary in this case. So, when y_t is $I(1)$, we may find an appropriate model for y_t by differencing each component of y_t once, and performing the regression based on Equation (2.11). However, then we have not taken into account that there may be dependencies between some of the components of y_t, e.g., two of the components may have a common trend, or there may exist a linear combination of the components of y_t which is stationary. For examle, if y_t consists of two interest rates of different maturities, the difference between the two interest rates often is stationary, while each of the interest rates is $I(1)$. This problem is generally solved by including an **error correction term** $\mathbf{\Pi} y_{t-1}$ in Equation (2.11) where $\mathbf{\Pi}$ is a $(K \times K)$ matrix which has rank $\text{rk}(\mathbf{\Pi}) < K$, because if $\mathbf{\Pi}$ has full rank K, then $\mathbf{\Pi}$ is invertible, such that the nonstationary variable y_{t-1} may be written as a sum of stationary terms by using this equation, which is an inconsistency. So, $\text{rk}(\mathbf{\Pi}) = r < K$ which implies that there exist $(K \times r)$-matrices $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ of rank r such that $\mathbf{\Pi} = \boldsymbol{\alpha}\boldsymbol{\beta}'$. Then, each of the r rows of $\boldsymbol{\beta}' y_{t-1}$ is a stationary linear combination of the components of y_t, and is called a **cointegration relation**. The number r, which is equal to the number of cointegration relations, is called the **cointegration rank**. As the matrix $\boldsymbol{\beta}$ contains all the coefficients of the cointegration relations, it is called the **cointegration matrix**. The matrix $\boldsymbol{\alpha}$, which is the coefficient matrix of the stationary term $\boldsymbol{\beta}' y_{t-1}$ in Equation (2.12), is called the **loading matrix**. Thus, we have arrived at the **vector error correction model** (VECM):

Definition 2.8. A VECM of order p is defined by

$$\Delta y_t = \mu + \boldsymbol{\alpha}\boldsymbol{\beta}' y_{t-1} + \mathbf{\Gamma}_1 \Delta y_{t-1} + \mathbf{\Gamma}_2 \Delta y_{t-2} + \cdots + \mathbf{\Gamma}_{p-1} \Delta y_{t-p+1} + u_t \quad t = 1, \ldots, T \tag{2.12}$$

where $y_t = [y_{1t}, \ldots, y_{Kt}]'$ is a $(K \times 1)$ random vector, μ is a $(K \times 1)$ constant vector, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are $(K \times r)$ matrices such that $\text{rk}(\boldsymbol{\alpha}) = \text{rk}(\boldsymbol{\beta}) = r$ with $0 < r < K$, the $\mathbf{\Gamma}_i$ are fixed $(K \times K)$ coefficient matrices, and u_t is a K-dimensional white noise process, i.e., $E(u_t) = 0$, $E(u_t u_s') = 0$ when $t \neq s$ and $E(u_t u_t') = \Sigma_u$ where Σ_u is nonsingular.

Remark 1. Note that the case $r = 0$ is excluded in this definition. If $r = 0$, there does not exist any cointegration relation, so we do not need an error correction term such that the VAR model in diffences in Equation 2.11 is approriate.

Remark 2. When we rewrite the VAR(p) model (2.8) to the VECM form (2.12), the matrix $\mathbf{\Pi}$ is uniquely determined as $\mathbf{\Pi} = I - A_1 - A_2 - \cdots - A_p$, but the $(K \times r)$-matrices $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ such that $\mathbf{\Pi} = \boldsymbol{\alpha}\boldsymbol{\beta}'$ are not uniquely determined. Therefore, common practice is to normalize $\boldsymbol{\beta}$ such that the upper $(r \times r)$ submatrix of $\boldsymbol{\beta}$ is I_r, the identity matrix of order r. When $K = 2$, as later in this thesis, the cointegration rank is $r = 1$ if there exists cointegration at all. Thus, the cointegration matrix is in this case the vector $\boldsymbol{\beta}' = [1 \ \ -\beta]$ where β is a number, which we call the **cointegration value**.

Remark 3. Also, note that the VECM of order p always may be rewritten as a VAR(p) model by substituting all the Δ-operators in Equation (2.12) with its definition (e.g., Δy_t replaced by $y_t - y_{t-1}$), and simplifying. However, this VAR(p) model is no longer **unrestricted** as the model (2.8), because there is at least one cointegration relation in this VECM of order p as the number r of cointegration relations should satisfy $0 < r < K$.

2.6.1 The Johansen cointegration rank test

The cointegration rank r has to be determined before we estimate the VECM. Therefore, we perform a likelihood ratio test to find the cointegration rank, before we estimate the restricted VECM with this specific cointegration rank. A likelihood ratio statistic for testing

$$H_0: \ r = r_0 \quad \text{against} \quad H_1: \ r_0 < r \leq r_1$$

is given by

$$\lambda_{LR}(r_0, r_1) = -T \sum_{i=r_0+1}^{r_1} \ln(1 - \lambda_i),$$

where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_K$ are the eigenvalues of the matrix $S_{11}^{-\frac{1}{2}} S_{10} S_{00}^{-1} S_{01} S_{11}^{-\frac{1}{2}}$ which is computed by using the following matrix formulas:

$$\Delta Y = [\Delta y_1, \ldots, \Delta y_T]$$
$$Y_{-1} = [y_0, \ldots, y_{T-1}]$$
$$\Delta X_{t-1} = [\Delta y'_{t-1}, \ldots, \Delta y'_{t-p+1}, \mu']'$$
$$\Delta X = [\Delta X_0, \ldots, \Delta X_{T-1}]$$
$$M = I_T - \Delta X'(\Delta X \Delta X')^{-1} \Delta X$$
$$R_0 = \Delta Y M$$
$$R_1 = Y_{-1} M$$
$$S_{ij} = R_i R'_j / T, \quad i = 0, 1, \ j = 0, 1.$$

By letting r_1 have different values in the interval $[r_0 + 1, K]$, we get different LR tests for the cointegration rank. If $r_1 = r_0 + 1$, we get the **trace test**, and if $r_1 = K$, we get the **maximum eigenvalue test**. The function `ca.jo` in the R package **urca** performs the trace test and the maximum eigenvalue test when the parameter `type` is equal to `"trace"` and `"eigen"`, respectively. However, the limiting distributions of these LR statistics are nonstandard, but they may be simulated by considering multivariate random walks. Conveniently, the function `ca.jo` reports the critical values at the 1%, 5% and 10% significance level, so we reject H_0 when the LR statistic has larger value than the critical value for the chosen significance level.

When the cointegration rank r is determined, we use the function `cajorls` in the R package **urca** to compute the least squares estimate of the restricted VECM with cointegration rank r.

2.7 Threshold vector error correction models

In Section 2.4 on page 10 a nonlinear SETAR(p) model was made from linear AR(p) models by introducing threshold(s). Similary, we may make a nonlinear threshold vector error correction model (TVECM) from a linear VECM:

Definition 2.9. A K-dimensional time series y_t is said to follow a k-regime TVECM of order p if it satisfies

$$\Delta y_t = c_j + \mathbf{\Pi}_j y_{t-1} + \mathbf{\Gamma}_{1j} y_{t-1} + \cdots + \mathbf{\Gamma}_{(p-1),j} y_{t-p+1} + u_{tj}, \qquad \text{if} \qquad \gamma_{j-1} < y_{t-d-1} \le \gamma_j, \tag{2.13}$$

where $p \ge 0$, $k > 1$ and $0 \le d < p$ are integers, $j = 1, \dots, k$ and γ_i are real numbers such that $-\infty = \gamma_0 < \gamma_1 < \cdots < \gamma_k = \infty$, and u_{tj} is a K-dimensional white noise process for each $j = 1, \dots, k$. The number p is the **AR order**, d is the **threshold delay**, i.e., the time delay of the threshold variable y_{t-d-1} compared with y_{t-1}, the integer j denotes the **regime number**, while the numbers $\gamma_1, \dots, \gamma_{k-1}$ denote the $k-1$ **thresholds** which divide the threshold space into k regimes.

Note that, in each of the k regimes, the time series y_t is a linear VECM. However, all these k linear models are different (otherwise, we may reduce k by merging some of the regimes), and which one of these k models we actually use when computing y_t, is governed by the value of the threshold variable y_{t-d-1}. As a result, the TVECM is nonlinear.

We estimate a TVECM by using the function *`TVECM`* in the R package **tsDyn**. Note that in **tsDyn** only TVECMs with threshold delay equal to 0 may be estimated and tested.

2.7.1 Checking the residuals of an estimated TVECM

Unfortunately, the functions in the R package **vars** for checking the residuals may only be applied to estimated VAR(p) models. Therefore, we have made an R function (included in R chunk 40 in Appendix C) which performs the ARCH-LM test described in Section 2.5.2 on a general data set. When applying this function on the residuals from the estimated VAR(3) model, the results are similar, but not exactly equal to the results from the funcion *`arch.test`*. In this chunk we also perform a Doornik-Hansen univariate and multivariate normality test (Doornik and Hansen 1994) on the residuals of the estimated TVECM by using the function *`DH.test`* in the R package **asbio**. This test is designed to deal with small samples rather than the asymptotic Jarque-Bera test which is implemented in the function *`normality.test`*. But more important for us, this function performs the normality test on whatever data set, not only on the residuals of an estimated VAR(p) model as the function *`normality.test`*.

2.7.2 The Hansen and Seo test

When we have estimated a TVECM, it is important to decide whether this nonlinear model is superior to a linear VECM. Such a test was proposed in Hansen and Seo (2002), and is implemented in the function *`TVECM.HStest`* in the R package **tsDyn**. By using this function, we are able to test a linear VECM against a two-regime TVECM. The idea is to test whether the difference between the parameter matrices in the two regimes, is significantly different from zero. If so, the TVECM catches some dynamics of the given time series which the linear VECM does not, and accordingly, the TVECM is superior to a linear VECM. However, in Chapter 3 we extend this test to cover the case of a three-regime TVECM as the alternative hypothesis, so we postpone the details of the test statistic until this chapter. In our data analysis in Chapter 4 and 5 we use this extended version of the function ***TVECM.HStest***.

Chapter 3

Testing for three-regime threshold cointegration

In this chapter we extend the function ***`TVECM.HStest`*** to the case of a three-regime TVECM in the alternative hypothesis. In Section 3.1 we generalize both the LM statistic described in Hansen and Seo (2002), and the LM statistic implemented in the function ***`TVECM.HStest`***, and we show that these two LM statistics are equivalent under certain conditions. When computing the test statistic in the Hansen and Seo test, it is necessary to perform a grid search to find the largest LM statistic when the thresholds vary over the set of all possible threshold values. In Section 3.2 we extend this grid search to the case of two thresholds, and we improve this grid search such that it really finds the largest LM statistic under the constraints specified by the user, when the cointegration value β is given. Throughout this chapter we assume that $K = 2$, i.e., only bivariate models are considered.

Appendix A contains the source code of the function ***`TVECM.HSTest`*** in the version 0.7-40 of the package **tsDyn** which was the starting point for our work on the Hansen and Seo test, while Appendix B contains the source code of our new version of this function.

3.1 The derivation of the SupLM statistic in the case of three regimes

3.1.1 Some matrix formulas

This section contains the vector and matrix formulas, which are necessary when deriving the formulas for the LM statistics, but not usually covered in an introductory linear algebra course.

Matrix rule 1. ***Product of partitioned matrices.***
Let the $(m \times n)$ matrix A be partitioned into submatrices $A_{11}, A_{12}, A_{21}, A_{22}$ with dimensions $(p{\times}q)$, $p{\times}(n{-}q))$, $((m{-}p){\times}q)$, and $((m{-}p){\times}(n{-}q))$, respectively, such that $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$.

Let the $(n \times t)$ *matrix* B *be partitioned as* $B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$. *Then, we have that:*

$$AB = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}.$$

provided that A *and* B *are partitioned such that all the products of the submatrices are defined.*

Matrix rule 2. ***Inverse of a partitioned matrix.***

Suppose that the invertible matrix A *is partitioned as* $A = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix}$ *where the matrices* A_{11} *and* A_{22} *both are invertible. Then, we have that*

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} & 0 \\ 0 & A_{22}^{-1} \end{bmatrix}.$$

Matrix rule 3. ***Transpose of a partitioned matrix.***

Let A' *denote the transpose of* A*, and let* A *be partitioned as* $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$. *Then we have that* $A' = \begin{bmatrix} A'_{11} & A'_{21} \\ A'_{12} & A'_{22} \end{bmatrix}$.

Matrix definition 1. Orthogonal matrices.

An $(m \times k)$ matrix B is orthogonal to the $(m \times n)$ matrix A if $A'B = 0$.

Matrix definition 2. The vec and vech operator.

Let A be an $(m \times n)$ matrix with the columns $A_1, A_2, \ldots, A_n$. Then the **vec** operator stacks the columns of A into an $(mn \times 1)$ vector, i.e.,

$$\text{vec}(A) = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix}.$$

While the vec operator stacks all the elements in the columns, the **vech** operator stacks only the elements on and below the main diagonal of an $(m \times m)$ matrix A into an $((\frac{1}{2}m(m+1)) \times 1)$ vector.

Matrix definition 3. Kronecker product

Let $A = (a_{ij})$ and $B = (b_{ij})$ be $(m \times n)$ and $(p \times q)$ matrices, respectively. The $(mp \times nq)$ matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}$$

is the **Kronecker product** of A and B.

Matrix rule 4. ***Rules for the Kronecker product and the vec operator.***
Let A, B and C be matrices of appropriate dimensions. Then we have the following rules:

$$(A \otimes B)' = A' \otimes B'$$
$$A \otimes (B + C) = A \otimes B + A \otimes C$$
$$(A \otimes B)(C \otimes D) = AC \otimes BD$$
$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} \quad \textit{when } A \textit{ and } B \textit{ are invertible,}$$
$$\text{vec}(A + B) = \text{vec}(A) + \text{vec}(B)$$
$$\text{vec}(ABC) = (C' \otimes A)\text{vec}(B)$$
$$\text{vec}(AB) = (I_n \otimes A)\text{vec}(B) \quad \textit{where } n \textit{ is the number of columns in } B$$

3.1.2 The three-regime TVECM written in matrix form

Let x_t be a two-dimensional $I(1)$ time series with normalized cointegration vector $\boldsymbol{\beta} = [1, \ -\beta]'$ where β is the cointegration value, and let $w_t(\beta) = \boldsymbol{\beta}' x_t$ denote the cointegration relation. A linear VECM of order p may then be written as

$$\Delta x_t = A' X_{t-1}(\beta) + u_t$$

where the regressor

$$X_{t-1}(\beta) = \begin{bmatrix} 1 & w_{t-1}(\beta)' & \Delta x'_{t-1} & \Delta x'_{t-2} & \cdots & \Delta x'_{t-p+1} \end{bmatrix}'$$

is of format $((2p) \times 1)$, the parameter matrix A is of format $((2p) \times 2)$, and the error u_t is of format (2×1) and is assumed to have finite covariance matrix $\Sigma = \text{E}(u_t u_t')$.

Let $Y = \begin{bmatrix} \Delta x_1 & \Delta x_2 & \cdots & \Delta x_T \end{bmatrix}'$, $X = \begin{bmatrix} X_0(\beta) & X_1(\beta) & \cdots & X_{T-1}(\beta) \end{bmatrix}'$, and $U = \begin{bmatrix} u_1 & u_2 & \cdots & u_T \end{bmatrix}'$. Then the linear VECM above may be written compactly as

$$Y = XA + U \tag{3.1}$$

where Y is of format $(T \times 2)$, X is of format $(T \times (2p))$, A is of format $((2p) \times 2)$, and U is of format $(T \times 2)$. When the errors u_t are Gaussian, and the cointegration value β is fixed, the maximum likelihood estimator $\tilde{A}$ is equal to the ordinary least squares estimator which is given by $\widehat{A} = (X'X)^{-1}X'Y$.

We extend this linear VECM to a TVECM with three regimes and threshold delay 0: Let the threshold parameter be $\gamma = (\gamma_1, \gamma_2)$ where $\gamma_1 < \gamma_2$, and let

$$d_{1t}(\beta, \gamma) = 1(w_{t-1}(\beta) \le \gamma_1)$$
$$d_{2t}(\beta, \gamma) = 1(\gamma_1 < w_{t-1}(\beta) \le \gamma_2)$$
$$d_{3t}(\beta, \gamma) = 1(w_{t-1}(\beta) > \gamma_2)$$

where $1(\cdot)$ denotes the **indicator function**, i.e., $1(P) = 1$ if the logical condition P is TRUE and $1(P) = 0$ otherwise. Then the TVECM with three regimes may be written as:

$$\Delta x_t = A_1' X_{t-1}(\beta) d_{1t}(\beta, \gamma) + A_2' X_{t-1}(\beta) d_{2t}(\beta, \gamma) + A_3' X_{t-1}(\beta) d_{3t}(\beta, \gamma) + u_t.$$

If we let Y and U be as above, and let $Z_i = \begin{bmatrix} X_0(\beta)'d_{i1}(\beta,\gamma) \\ X_1(\beta)'d_{i2}(\beta,\gamma) \\ \vdots \\ X_{T-1}(\beta)'d_{iT}(\beta,\gamma) \end{bmatrix}$ for $i = 1,2,3$, the model may be more compactly written as

$$Y = Z_1A_1 + Z_2A_2 + Z_3A_3 + U = ZA + U \tag{3.2}$$

where $Z = \begin{bmatrix} Z_1 & Z_2 & Z_3 \end{bmatrix}$ and $A = \begin{bmatrix} A_1' & A_2' & A_3' \end{bmatrix}'$.

So, the T observations of the bivariate time series are now divided into three regimes. If the t-th observation is in regime i, the t-th row of the matrix Z_i is equal to the t-th row of the matrix X, while the t-th row of the matrices Z_j, $j \neq i$ is equal to 0. An important consequense of this fact is that $X = Z_1 + Z_2 + Z_3$ and $Z_i'Z_j = 0$ when $i \neq j$, i.e., Z_i is orthogonal to Z_j when $i \neq j$.

Typically, the middle regime in a three-regime TVECM contains a large part of the observations of the time series. So, when testing the hypothesis H_1 of a three-regime TVECM against the hypothesis H_0 of a linear VECM, we actually test whether there are significant differences between the parameters in the middle regime and the parameters in the lower and upper regime. If the differences are not significant, then we may use the same parameter matrix in all three regimes, so

$$Y = Z_1A_1 + Z_2A_2 + Z_3A_3 + U = Z_1A + Z_2A + Z_3A + U = (Z_1 + Z_2 + Z_3)A + U = XA + U$$

which is the linear VECM (3.1) above.

Therefore, we rewrite the three-regime TVECM as follows:

$$\begin{aligned} Y &= Z_1A_1 + Z_2A_2 + Z_3A_3 + U = Z_1A_1 + (X - Z_1 - Z_3)A_2 + Z_3A_3 + U \\ &= Z_1(A_1 - A_2) + XA_2 + Z_3(A_3 - A_2) + U = XA_2 + Z_1B_1 + Z_3B_3 + U \end{aligned} \tag{3.3}$$

So, testing for three-regime threshold cointegration implies testing whether the parameter matrix $\begin{bmatrix} B_1 & B_3 \end{bmatrix}$ is significantly different from zero.

3.1.3 The least squares estimators of the parameters

In this section we compute the least squares estimators of the parameter matrices of the three-regime TVECM.

Proposition 3.1. *The least squares estimators of the parameter matrices A_2, B_1 and B_3 in the three-regime TVECM $Y = XA_2 + Z_1B_1 + Z_3B_3 + U$ are given by:*

$$\widehat{A}_2 = (Z_2'Z_2)^{-1}Z_2'Y,\ \widehat{B}_1 = \left[(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2'\right]Y,\ \textit{and}\ \widehat{B}_3 = \left[(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2'\right]Y.$$

Proof. First, we find the least squares estimator of A in model (3.2) by using the rules for partitioned matrices in Section 3.1.1 and the fact that Z_i is orthogonal to Z_j when $i \neq j$.

$$\widehat{A} = \begin{bmatrix} \widehat{A}_1 \\ \widehat{A}_2 \\ \widehat{A}_3 \end{bmatrix} = (Z'Z)^{-1}Z'Y = \left(\begin{bmatrix} Z_1' \\ Z_2' \\ Z_3' \end{bmatrix} \begin{bmatrix} Z_1 & Z_2 & Z_3 \end{bmatrix} \right)^{-1} \begin{bmatrix} Z_1' \\ Z_2' \\ Z_3' \end{bmatrix} Y$$

$$= \begin{bmatrix} Z_1'Z_1 & 0 & 0 \\ 0 & Z_2'Z_2 & 0 \\ 0 & 0 & Z_3'Z_3 \end{bmatrix}^{-1} \begin{bmatrix} Z_1' \\ Z_2' \\ Z_3' \end{bmatrix} Y = \begin{bmatrix} (Z_1'Z_1)^{-1} & 0 & 0 \\ 0 & (Z_2'Z_2)^{-1} & 0 \\ 0 & 0 & (Z_3'Z_3)^{-1} \end{bmatrix} \begin{bmatrix} Z_1' \\ Z_2' \\ Z_3' \end{bmatrix} Y = \begin{bmatrix} (Z_1'Z_1)^{-1}Z_1'Y \\ (Z_2'Z_2)^{-1}Z_2'Y \\ (Z_3'Z_3)^{-1}Z_3'Y \end{bmatrix}$$

By using $B_1 = A_1 - A_2$ and $B_3 = A_3 - A_2$ from (3.3), we get

$$\widehat{A}_2 = (Z_2'Z_2)^{-1}Z_2'Y$$

$$\widehat{B}_1 = \widehat{A}_1 - \widehat{A}_2 = (Z_1'Z_1)^{-1}Z_1'Y - (Z_2'Z_2)^{-1}Z_2'Y = \left[(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2'\right] Y$$

$$\widehat{B}_3 = \widehat{A}_3 - \widehat{A}_2 = (Z_3'Z_3)^{-1}Z_3'Y - (Z_2'Z_2)^{-1}Z_2'Y = \left[(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2'\right] Y$$

□

3.1.4 The LM statistics

As in Seo (2003) we use a Lagrange Multiplier test with the following LM statistic:

$$\text{LM}(\beta, \gamma) = \hat{s}' \left(\widehat{\text{Var}(\hat{s})}\right)^{-1} \hat{s} \tag{3.4}$$

where $\hat{s} = \hat{s}(\beta, \gamma) = \begin{bmatrix} \text{vec}\left(\widehat{B}_1\right) \\ \text{vec}\left(\widehat{B}_3\right) \end{bmatrix}$ is the least squares estimator of the parameter vector, i.e., the parameter matrix vectorized by using the $\text{vec}(\cdot)$ operator defined in Section 3.1.1, and $\widehat{\text{Var}(\hat{s})}$ is an estimate of the covariance matrix of $\hat{s}$. Because the conditional heteroscedasticity is commonly used in time series analysis of interest rates, we use the White-Eicker heteroscedasticity consistent covariance estimator, which is defined as follows: Let $U = \begin{bmatrix} U_1 & U_2 \end{bmatrix}$ be the residuals of Equation (3.3), and let $\widehat{U} = \begin{bmatrix} \widehat{U}_1 & \widehat{U}_2 \end{bmatrix}$ be the corresponding estimated residuals. Then, the **White-Eicker heteroscedasticity consistent estimator** of $\text{Var}(\text{vec}(U)) = \text{Var}\left(\begin{bmatrix} U_1 \\ U_2 \end{bmatrix}\right)$ is defined as the matrix $D = \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix}$ where $D_{ij} = \text{diag}(\widehat{U}_{1i}\widehat{U}_{1j}, \widehat{U}_{2i}\widehat{U}_{2j}, \ldots \widehat{U}_{Ti}\widehat{U}_{Tj})$, i.e., the diagonal matrix with $\widehat{U}_{1i}\widehat{U}_{1j}, \widehat{U}_{2i}\widehat{U}_{2j}, \ldots, \widehat{U}_{Ti}\widehat{U}_{Tj}$ on the diagonal.

If we, on the contrary, suppose that the error process u_t is homoscedastic, i.e., $\Sigma_u = \text{E}(u_t u_t') = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}$ for all $t = 1, 2, \ldots, T$, we have that

$$\text{Var}(\text{vec}(U)) = \text{Var}\left(\begin{bmatrix} U_1 \\ U_2 \end{bmatrix}\right) = \text{E}\left(\begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \begin{bmatrix} U_1' & U_2' \end{bmatrix}\right) = \begin{bmatrix} \text{E}(U_1U_1') & \text{E}(U_1U_2' \\ \text{E}(U_2U_1') & \text{E}(U_2U_2') \end{bmatrix}$$

$$= \begin{bmatrix} \sigma_{11}I_T & \sigma_{12}I_T \\ \sigma_{21}I_T & \sigma_{22}I_T \end{bmatrix} = \Sigma_u \otimes I_T.$$

In this case we may use the estimator $\widetilde{\Sigma}_u = \frac{1}{T}\widehat{U}\widehat{U}'$.

Proposition 3.2. *With the notation above, the least squares estimator $\hat{s}$ of the parameter vector is given by*

$$\hat{s} = \begin{bmatrix} I_2 \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2'] \\ I_2 \otimes [(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2'] \end{bmatrix} \mathrm{vec}(Y) = s + \begin{bmatrix} I_2 \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2'] \\ I_2 \otimes [(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2'] \end{bmatrix} \mathrm{vec}(U) \tag{3.5}$$

The White-Eicker heteroscedastic consistent estimate of the covariance matrix of $\hat{s}$ is given by

$$\begin{bmatrix} I_2 \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2'] \\ I_2 \otimes [(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2'] \end{bmatrix} D \begin{bmatrix} I_2 \otimes [Z_1(Z_1'Z_1)^{-1} - Z_2(Z_2'Z_2)^{-1}] & I_2 \otimes [Z_3(Z_3'Z_3)^{-1} - Z_2(Z_2'Z_2)^{-1}] \end{bmatrix} \tag{3.6}$$

If we suppose that the error process u_t is standard white noise, we may estimate the covarianse matrix of $\hat{s}$ by

$$\frac{1}{T}\begin{bmatrix} \widehat{U}'\widehat{U} \otimes [(Z_1'Z_1)^{-1} + (Z_2'Z_2)^{-1}] & \widehat{U}'\widehat{U} \otimes [(Z_2'Z_2)^{-1}] \\ \widehat{U}'\widehat{U} \otimes [(Z_2'Z_2)^{-1}] & \widehat{U}'\widehat{U} \otimes [(Z_2'Z_2)^{-1} + (Z_3'Z_3)^{-1}] \end{bmatrix} \tag{3.7}$$

Proof. We apply the rules for the vec(·) operator (see Section 3.1.1) to rewrite the expression for $\hat{s}$:

$$\begin{aligned} \hat{s} &= \begin{bmatrix} \mathrm{vec}(\widehat{B}_1) \\ \mathrm{vec}(\widehat{B}_3) \end{bmatrix} = \begin{bmatrix} \mathrm{vec}\{[(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2']Y\} \\ \mathrm{vec}\{[(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2']Y\} \end{bmatrix} \\ &= \begin{bmatrix} \left(I_2 \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2']\right)\mathrm{vec}(Y) \\ \left(I_2 \otimes [(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2']\right)\mathrm{vec}(Y) \end{bmatrix} = \begin{bmatrix} I_2 \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2'] \\ I_2 \otimes [(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2'] \end{bmatrix}\mathrm{vec}(Y) \end{aligned}$$

which proves the first expression for $\hat{s}$. To prove the second expression we rewrite vec(Y):

$$\begin{aligned} \mathrm{vec}(Y) &= \mathrm{vec}(XA_2 + Z_1B_1 + Z_3B_3 + U) = \mathrm{vec}(XA_2) + \mathrm{vec}(Z_1B_1) + \mathrm{vec}(Z_3B_3) + \mathrm{vec}(U) \\ &= (I_2 \otimes X)\mathrm{vec}(A_2) + (I_2 \otimes Z_1)\mathrm{vec}(B_1) + (I_2 \otimes Z_3)\mathrm{vec}(B_3) + \mathrm{vec}(U) \end{aligned}$$

In addition, note that $X = Z_1 + Z_2 + Z_3$, that $Z_i'Z_j = 0$ when $i \neq j$, and that $(Z_i'Z_i)^{-1}Z_i'Z_i = I_N$ where $N = 2p$ is the number of columns in Z_i. By using these facts, we get

$$\begin{aligned} \hat{s} &= \begin{bmatrix} I_2 \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2'] \\ I_2 \otimes [(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2'] \end{bmatrix}\mathrm{vec}(Y) \\ &= \begin{bmatrix} I_2 \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2'] \\ I_2 \otimes [(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2'] \end{bmatrix}[(I_2 \otimes X)\mathrm{vec}(A_2) + (I_2 \otimes Z_1)\mathrm{vec}(B_1) + (I_2 \otimes Z_3)\mathrm{vec}(B_3) + \mathrm{vec}(U)] \\ &= \begin{bmatrix} [I_2 \otimes (I_N - I_N)]\mathrm{vec}(A_2) + (I_2 \otimes I_N)\mathrm{vec}(B_1) + (I_2 \otimes 0)\mathrm{vec}(B_3) + (I_2 \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2])\mathrm{vec}(U) \\ [I_2 \otimes (I_N - I_N)]\mathrm{vec}(A_2) + (I_2 \otimes 0)\mathrm{vec}(B_1) + (I_2 \otimes I_N)\mathrm{vec}(B_3) + (I_2 \otimes [(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2])\mathrm{vec}(U) \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{vec}(B_1) + (I_2 \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2])\mathrm{vec}(U) \\ \mathrm{vec}(B_3) + (I_2 \otimes [(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2])\mathrm{vec}(U) \end{bmatrix} \\ &= s + \begin{bmatrix} I_2 \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2] \\ I_2 \otimes [(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2] \end{bmatrix}\mathrm{vec}(U) \end{aligned}$$

Next, we prove formula (3.6). Note that

$$\hat{s} - s = C\text{vec}(U) \quad \text{where} \quad C = \begin{bmatrix} I_2 \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2] \\ I_2 \otimes [(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2] \end{bmatrix}$$

Thus, we get

$$\text{Cov}(\hat{s}) = \text{Cov}(\hat{s} - s) = C\text{Cov}(\text{vec}(U))C' = CDC'$$

which is equivalent to formula (3.6).

To prove formula (3.7), we substitute $\Sigma_u \otimes I_T$ for D in expression (3.6) and simplify:

$$\begin{bmatrix} I_2 \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2'] \\ I_2 \otimes [(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2'] \end{bmatrix} (\Sigma_u \otimes I_T) \begin{bmatrix} I_2 \otimes [Z_1(Z_1'Z_1)^{-1} - Z_2(Z_2'Z_2)^{-1}] & I_2 \otimes [Z_3(Z_3'Z_3)^{-1} - Z_2(Z_2'Z_2)^{-1}] \end{bmatrix}$$

$$= \begin{bmatrix} \Sigma_u \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2'] \\ \Sigma_u \otimes [(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2'] \end{bmatrix} \begin{bmatrix} I_2 \otimes [Z_1(Z_1'Z_1)^{-1} - Z_2(Z_2'Z_2)^{-1}] & I_2 \otimes [Z_3(Z_3'Z_3)^{-1} - Z_2(Z_2'Z_2)^{-1}] \end{bmatrix}$$

We expand the first of the 4 entries in this matrix product:

$$\left(\Sigma_u \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2']\right)\left(I_2 \otimes [Z_1(Z_1'Z_1)^{-1} - Z_2(Z_2'Z_2)^{-1}]\right) =$$
$$\Sigma_u \otimes [(Z_1'Z_1)^{-1}Z_1'Z_1(Z_1'Z_1)^{-1} - (Z_1'Z_1)^{-1}Z_1'Z_2(Z_2'Z_2)^{-1} - (Z_2'Z_2)^{-1}Z_2'Z_1(Z_1'Z_1)^{-1} + (Z_2'Z_2)^{-1}Z_2'Z_2(Z_2'Z_2)^{-1}] =$$
$$\Sigma_u \otimes [(Z_1'Z_1)^{-1} - 0 - 0 + (Z_2'Z_2)^{-1}] = \Sigma_u \otimes [(Z_1'Z_1)^{-1} + (Z_2'Z_2)^{-1}]$$

By expanding the three other entries similarly, and replacing the covariance matrix Σ_u by its estimate $\frac{1}{T}\widehat{U}'\widehat{U}$, we get formula (3.7). □

The formulas in Hansen and Seo (2002) for the LM statistic may now be generalized to the three-regime TVECM:

Proposition 3.3. *We use the following notation:*

$D_i = diag(U_{1j}, U_{2j}, \ldots, U_{Tj})$ *for* $j = 1, 2$
$M_i = I_2 \otimes Z_i'Z_i$ *for* $i = 1, 2, 3$
$\xi_i = \begin{bmatrix} D_1 Z_i & D_2 Z_i \end{bmatrix}$ *for* $i = 1, 2, 3$
$\Omega_i = \xi_i'\xi_i$ *for* $i = 1, 2, 3$
$V_i = M_i^{-1}\Omega_i M_i^{-1}$ *for* $i = 1, 2, 3$
$R_i = \left(I_2 \otimes (Z_i'Z_i)^{-1}Z_i'\right)$ *for* $i = 1, 2, 3$.

Then, the matrix $V = \begin{bmatrix} V_1 + V_2 & V_2 \\ V_2 & V_2 + V_3 \end{bmatrix}$ *is equal to the White-Eicker heteroscedastic consistent estimator (3.6) of the covariance matrix of* $\hat{s}$*, and the LM statistic is given by*

$$LM_1 = \hat{s}'V^{-1}s = \text{vec}(Y)' \begin{bmatrix} R_1' - R_2' & R_3' - R_2' \end{bmatrix} V^{-1} \begin{bmatrix} R_1 - R_2 \\ R_3 - R_2 \end{bmatrix} \text{vec}(Y).$$

Remark 4. For a two-regime TVECM, the only changes from the formulas in the proposition are $V = V_1 + V_2$, $\quad s = [R_1 - R_2]\text{vec}(Y)$, and $i = 1, 2$.

Proof. We compute V_i:

$$V_i = M_i^{-1}\Omega_i M_i^{-1} = (I_2 \otimes (Z_i'Z_i)^{-1})\begin{bmatrix} D_1Z_i & D_2Z_i \end{bmatrix}'\begin{bmatrix} D_1Z_i & D_2Z_i \end{bmatrix}(I_2 \otimes (Z_i'Z_i)^{-1})$$
$$= (I_2 \otimes (Z_i'Z_i)^{-1})\begin{bmatrix} Z_i'D_1 \\ Z_i'D_2 \end{bmatrix}\begin{bmatrix} D_1Z_i & D_2Z_i \end{bmatrix}(I_2 \otimes (Z_i'Z_i)^{-1})$$

$$= (I_2 \otimes (Z_i'Z_i)^{-1})(I_2 \otimes Z_i')\begin{bmatrix} D_1 \\ D_2 \end{bmatrix}\begin{bmatrix} D_1 & D_2 \end{bmatrix}(I_2 \otimes Z_i)(I_2 \otimes (Z_i'Z_i)^{-1})$$
$$= (I_2 \otimes (Z_i'Z_i)^{-1})(I_2 \otimes Z_i')\begin{bmatrix} D_1D_1 & D_1D_2 \\ D_2D_1 & D_2D_2 \end{bmatrix}(I_2 \otimes Z_i)(I_2 \otimes (Z_i'Z_i)^{-1})$$
$$= (I_2 \otimes (Z_i'Z_i)^{-1}Z_i')\begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix}(I_2 \otimes Z_i(Z_i'Z_i)^{-1}) = \left(I_2 \otimes (Z_i'Z_i)^{-1}Z_i'\right)D\left(I_2 \otimes Z_i(Z_i'Z_i)^{-1}\right)$$

The heteroscedastic consistent estimator (3.6) of the covariance matrix of $\hat{s}$ may be written as:

$$\begin{bmatrix} R_1 - R_2 \\ R_3 - R_2 \end{bmatrix} D\begin{bmatrix} R_1' - R_2' & R_3' - R_2' \end{bmatrix} = \begin{bmatrix} R_1D - R_2D \\ R_3D - R_2D \end{bmatrix}\begin{bmatrix} R_1' - R_2' & R_3' - R_2' \end{bmatrix} =$$
$$\begin{bmatrix} R_1DR_1' - R_1DR_2' - R_2DR_1' + R_2DR_2' & R_1DR_3' - R_1DR_2' - R_2DR_3' + R_2DR_2' \\ R_3DR_1' - R_3DR_2' - R_2DR_1' + R_2DR_2' & R_3DR_3' - R_3DR_2' - R_2DR_3' + R_2DR_2' \end{bmatrix}$$

Now we need the following Lemma to simplify this expression further:

Lemma 3.4. $R_iDR_j' = 0$ *for all* i *and* j *such that* $i \neq j$, $i = 1, 2, 3$ *and* $j = 1, 2, 3$.

Proof. Note that if a column in Z_i' is equal to 0, then the corresponding column in $Z_i'D_{kl}$ is also equal to 0 because the matrices D_{kl} are diagonal matrices. Thus, if an element of a row in $Z_i'D_{kl}$ is $\neq 0$, then the corresponding element in each column of Z_j is $= 0$ when $j \neq i$, so $Z_i'D_{kl}Z_j = 0$ when $i \neq j$. Consequently, we get

$$(I_2 \otimes Z_i')D(I_2 \otimes Z_j) = \begin{bmatrix} Z_i' & 0 \\ 0 & Z_i' \end{bmatrix}\begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix}\begin{bmatrix} Z_j & 0 \\ 0 & Z_j \end{bmatrix} = \begin{bmatrix} Z_i'D_{11}Z_j & Z_i'D_{12}Z_j \\ Z_i'D_{21}Z_j & Z_i'D_{22}Z_j \end{bmatrix} = 0$$

Thus,we get

$$R_iDR_j' = \left(I_2 \otimes (Z_i'Z_i)^{-1}Z_i'\right)D\left(I_2 \otimes Z_j(Z_j'Z_j)^{-1}\right) = \left(I_2 \otimes (Z_i'Z_i)^{-1}\right)(I_2 \otimes Z_i')D(I_2 \otimes Z_j)\left(I_2 \otimes (Z_j'Z_j)^{-1}\right)$$
$$= \left(I_2 \otimes (Z_i'Z_i)^{-1}\right) \cdot 0 \cdot \left(I_2 \otimes Z_j(Z_j'Z_j)^{-1}\right) = 0$$

□

Now, we use the lemma to simplify:

$$\begin{bmatrix} R_1DR_1' - R_1DR_2' - R_2DR_1' + R_2DR_2' & R_1DR_3' - R_1DR_2' - R_2DR_3' + R_2DR_2' \\ R_3DR_1' - R_3DR_2' - R_2DR_1' + R_2DR_2' & R_3DR_3' - R_3DR_2' - R_2DR_3' + R_2DR_2' \end{bmatrix}$$
$$= \begin{bmatrix} V_1 - 0 - 0 + V_2 & 0 - 0 - 0 + V_2 \\ 0 - 0 - 0 + V_2 & V_3 - 0 - 0 + V_2 \end{bmatrix} = \begin{bmatrix} V_1 + V_2 & V_2 \\ V_2 & V_3 + V_2 \end{bmatrix}$$

□

In the function ***TVECM.HStest*** in the R package **tsDyn** (Di Narzo, Aznarte, and Stigler 2011), the LM statistic for a two-regime TVECM is computed. Now we generalize this formula to the three-regime TVECM and show that this LM statistic is identical to $\mathbf{LM}_1$ in Proposition 3.3.

Proposition 3.5. *Let $Y = XA_2 + U$ be the linear VECM defined above, let $Z = \begin{bmatrix} Z_1 & Z_3 \end{bmatrix}$ and let $H = I_T - X(X'X)^{-1}X$. Let $s_e = \mathrm{vec}((HZ)'Y)$ and $z_e = \begin{bmatrix} D_1HZ & D_2HZ \end{bmatrix}$ where $D_i = diag(U_{1i}, U_{2i}, \ldots, U_{Ti})$ as before. Then $z_e'z_e$ is the White-Eicker heteroscedastic consistent estimate of the covariance matrix of s_e, and the LM statistic $LM_2 = s_e'(z_e'z_e)^{-1}s_e$ is equal to the statistic LM_1 in Proposition 3.3, when $Z'HZ$ is invertible.*

Proof. When regressing Y on X, the residuals are given by $\widehat{U} = HY$. As the matrix H is symmetric and idempotent, i.e., $H^2 = H$, we have that

$$s_e = \mathrm{vec}((HZ)'Y) = \mathrm{vec}(Z'HY) = \mathrm{vec}(Z'HHY) = \mathrm{vec}(Z'H\widehat{U}) = (I_2 \otimes (HZ)')\mathrm{vec}(\widehat{U})$$

Thus, we get

$$\mathrm{Cov}(s_e) = \mathrm{Cov}((I_2 \otimes (HZ)')\mathrm{vec}(\widehat{U})) = (I_2 \otimes (HZ)')\mathrm{Cov}(\mathrm{vec}(\widehat{U}))(I_2 \otimes HZ),$$

while

$$z_e'z_e = \begin{bmatrix} Z'HD_1 \\ Z'HD_2 \end{bmatrix} \begin{bmatrix} D_1HZ & D_2HZ \end{bmatrix} = \begin{bmatrix} Z'HD_1^2HZ & Z'HD_1D_2HZ \\ Z'HD_2D_1HZ & Z'HD_2^2HZ \end{bmatrix} \quad (H \text{ and } D_i \text{ are symmetric})$$

$$= (I_2 \otimes Z'H) \begin{bmatrix} D_1^2 & D_1D_2 \\ D_2D_1 & D_2^2 \end{bmatrix} (I_2 \otimes HZ) = (I_2 \otimes (HZ)')D(I_2 \otimes HZ),$$

where $D = \begin{bmatrix} D_1^2 & D_1D_2 \\ D_2D_1 & D_2^2 \end{bmatrix}$ is the White-Eicker estimate of $\mathrm{Cov}(\mathrm{vec}(U))$.

To prove the last assertion, note that both LM_1 and LM_2 may be written as $w'\left(\widehat{\mathrm{Cov}(w)}\right)^{-1} w$, i.e., $w = \hat{s}$ in LM_1 and $w = s_e$ in LM_2. In addition, $\mathrm{Cov}(w)$ is in both cases estimated by using the White-Eicker heteroscedastic consistent estimator of $\mathrm{Cov}(U)$. Suppose that there exists an invertible matrix K such that $s_e = K\hat{s}$. Then, we have that

$$LM_2 = s_e'\left(\widehat{\mathrm{Cov}(s_e)}\right)^{-1} s_e = (K\hat{s})'\left(\widehat{\mathrm{Cov}(K\hat{s})}\right)^{-1} K\hat{s} = \hat{s}'K'\left(K\widehat{\mathrm{Cov}(\hat{s})}K'\right)^{-1} K\hat{s}$$
$$= \hat{s}'K'(K')^{-1}\left(\widehat{\mathrm{Cov}(\hat{s})}\right)^{-1} K^{-1}K\hat{s} = \hat{s}'\left(\widehat{\mathrm{Cov}(\hat{s})}\right)^{-1}\hat{s} = LM_1$$

Therefore, $LM_1 = LM_2$ if there exists such a matrix K. Note that the order of the estimated parameters is different in s_e and $\hat{s}$. In $\hat{s}$, all the elements of $\widehat{B_1}$ precede all the elements of $\widehat{B_3}$, while in s_e all the elements connected to the regression of Y_1 precede all the elements connected to the regression of Y_2. However, we may reorder the elements of $\hat{s}$ by leftmultiplying with the matrix $P = \begin{bmatrix} I_N & 0 & 0 & 0 \\ 0 & 0 & I_N & 0 \\ 0 & I_N & 0 & 0 \\ 0 & 0 & 0 & I_N \end{bmatrix}$ where $N = 2p$ is the number of columns in the X matrix:

$$P\hat{s} = \begin{bmatrix} I_N & 0 & 0 & 0 \\ 0 & 0 & I_N & 0 \\ 0 & I_N & 0 & 0 \\ 0 & 0 & 0 & I_N \end{bmatrix} \begin{bmatrix} I_2 \otimes [(Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2'] \\ I_2 \otimes [(Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2'] \end{bmatrix} \mathrm{vec}(Y)$$

$$= \left[I_2 \otimes \begin{bmatrix} (Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2' \\ (Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2' \end{bmatrix} \right] \mathrm{vec}(Y),$$

so the order of the parameters in $P\hat{s}$ and s_e is equal. The matrix P is invertible as $P^2 = I$. Now, we prove that $s_e = (I_2 \otimes Z'HZ)P\hat{s}$. This will complete the proof of the assertion that $LM_1 = LM_2$ as $(I_2 \otimes Z'HZ)P$ is invertible when $Z'HZ$ is invertible. We rewrite this equation:

$$s_e = (I_2 \otimes Z'HZ)P\hat{s}$$

$$(I_2 \otimes (HZ)')\mathrm{vec}(Y) = (I_2 \otimes Z'HZ) \left[I_2 \otimes \begin{bmatrix} (Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2' \\ (Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2' \end{bmatrix} \right] \mathrm{vec}(Y)$$

$$(I_2 \otimes Z'H)\mathrm{vec}(Y) = \left(I_2 \otimes Z'H \begin{bmatrix} Z_1 & Z_3 \end{bmatrix} \begin{bmatrix} (Z_1'Z_1)^{-1}Z_1' - (Z_2'Z_2)^{-1}Z_2' \\ (Z_3'Z_3)^{-1}Z_3' - (Z_2'Z_2)^{-1}Z_2' \end{bmatrix} \right) \mathrm{vec}(Y)$$

$$\left(I_2 \otimes \begin{bmatrix} Z_1'H \\ Z_3'H \end{bmatrix} \right) \mathrm{vec}(Y) = \left(I_2 \otimes \begin{bmatrix} Z_1'H \left(Z_1(Z_1'Z_1)^{-1}Z_1' + Z_3(Z_3'Z_3)^{-1}Z_3' - (Z_1 + Z_3)(Z_2'Z_2)^{-1}Z_2' \right) \\ Z_3'H \left(Z_1(Z_1'Z_1)^{-1}Z_1' + Z_3(Z_3'Z_3)^{-1}Z_3' - (Z_1 + Z_3)(Z_2'Z_2)^{-1}Z_2' \right) \end{bmatrix} \right) \mathrm{vec}(Y)$$

so it suffices to prove that

$$\begin{bmatrix} Z_1'H \left(Z_1(Z_1'Z_1)^{-1}Z_1' + Z_3(Z_3'Z_3)^{-1}Z_3' - (Z_1 + Z_3)(Z_2'Z_2)^{-1}Z_2' \right) \\ Z_3'H \left(Z_1(Z_1'Z_1)^{-1}Z_1' + Z_3(Z_3'Z_3)^{-1}Z_3' - (Z_1 + Z_3)(Z_2'Z_2)^{-1}Z_2' \right) \end{bmatrix} - \begin{bmatrix} Z_1'H \\ Z_3'H \end{bmatrix} = 0$$

We show the equality in detail only for the first component, as the proof for the second component is similar. (Interchanging Z_1 and Z_3 in the proof for the first component gives the proof for the second component.) First we simplify $Z_1'H$:

$$\begin{aligned} Z_1'H &= Z_1'(I_T - X(X'X)^{-1}X') = Z_1'(I_t - (Z_1 + Z_2 + Z_3)\left[(Z_1' + Z_2' + Z_3')(Z_1 + Z_2 + Z_3)\right]^{-1}(Z_1' + Z_2' + Z_3') \\ &= Z_1' - Z_1'Z_1\,(Z_1'Z_1 + Z_2'Z_2 + Z_3'Z_3)^{-1}\,(Z_1' + Z_2' + Z_3') \end{aligned}$$

where we have used that $Z_i'Z_j = 0$ when $i \neq j$. Thus, we get

$$\begin{aligned} & Z_1'H \left(Z_1(Z_1'Z_1)^{-1}Z_1' + Z_3(Z_3'Z_3)^{-1}Z_3' - (Z_1 + Z_3)(Z_2'Z_2)^{-1}Z_2' \right) - Z_1'H \\ &= \left[Z_1' - Z_1'Z_1\,(Z_1'Z_1 + Z_2'Z_2 + Z_3'Z_3)^{-1}\,(Z_1' + Z_2' + Z_3') \right] \left(Z_1(Z_1'Z_1)^{-1}Z_1' + Z_3(Z_3'Z_3)^{-1}Z_3' - (Z_1 + Z_3)(Z_2'Z_2)^{-1}Z_2' \right) \\ &\quad - Z_1' + Z_1'Z_1\,(Z_1'Z_1 + Z_2'Z_2 + Z_3'Z_3)^{-1}\,(Z_1' + Z_2' + Z_3') \\ &= Z_1' + 0 - (Z_1'Z_1)(Z_2'Z_2)^{-1}Z_2' - 0 - \left[Z_1'Z_1\,(Z_1'Z_1 + Z_2'Z_2 + Z_3'Z_3)^{-1} \right] \left(Z_1' + Z_3' - (Z_1'Z_1 + Z_3'Z_3)(Z_2'Z_2)^{-1}Z_2' \right) \\ &\quad - Z_1' + Z_1'Z_1\,(Z_1'Z_1 + Z_2'Z_2 + Z_3'Z_3)^{-1}\,(Z_1' + Z_2' + Z_3') \\ &= Z_1'Z_1\,(Z_1'Z_1 + Z_2'Z_2 + Z_3'Z_3)^{-1}\,Z_2' + \left[Z_1'Z_1\,(Z_1'Z_1 + Z_2'Z_2 + Z_3'Z_3)^{-1} \right] \left(Z_1'Z_1 + Z_3'Z_3)(Z_2'Z_2)^{-1}Z_2' \right) \\ &\quad - (Z_1'Z_1)(Z_2'Z_2)^{-1}Z_2' \end{aligned}$$

$$
\begin{aligned}
&= Z_1'Z_1\left((Z_1'Z_1 + Z_2'Z_2 + Z_3'Z_3)^{-1}\right)(Z_2'Z_2)(Z_2'Z_2)^{-1}Z_2' \\
&\quad + \left[Z_1'Z_1\,(Z_1'Z_1 + Z_2'Z_2 + Z_3'Z_3)^{-1}\right]\left((Z_1'Z_1 + Z_3'Z_3)(Z_2'Z_2)^{-1}Z_2'\right) - (Z_1'Z_1)(Z_2'Z_2)^{-1}Z_2' \\
&= \left[Z_1'Z_1\,(Z_1'Z_1 + Z_2'Z_2 + Z_3'Z_3)^{-1}\right]\left((Z_1'Z_1 + Z_2'Z_2 + Z_3'Z_3)(Z_2'Z_2)^{-1}Z_2'\right) - (Z_1'Z_1)(Z_2'Z_2)^{-1}Z_2' \\
&= (Z_1'Z_1)\left[(Z_1'Z_1 + Z_2'Z_2 + Z_3'Z_3)^{-1}\,(Z_1'Z_1 + Z_2'Z_2 + Z_3'Z_3) - I_N\right](Z_2'Z_2)^{-1}Z_2' \\
&= (Z_1'Z_1)(I_N - I_N)(Z_2'Z_2)^{-1}Z_2' = 0
\end{aligned}
$$

which completes the proof. □

Remark 5. Note that the left multiplication by $D_i = \mathrm{diag}(U_i)$ in z_e is equivalent to element-wise multiplication by the vector U_i. When LM_2 is computed, the element-wise multiplication should be used as in **tsDyn**, because this is more efficient than leftmultiplying with D_i. However, mathematically, it is preferable to use leftmultiplying with D_i instead of introducing a new symbol for the element-wise multiplication.

Remark 6. According to Engle (1984), the statistic LM_1 in Proposition 3.3 is the Wald statistic, and the statistic LM_2 in Proposition 3.5 is the Lagrange Multiplier statistic, while in Hansen and Seo (2002) LM_1 is called the LM statistic. In Engle (1984), it is proved that the Wald statistic, the LM statistic and the likelihood ratio statistic are identical when the loglikelihood is a quadratic function, which is the case when multinormal distribution is presupposed. As $LM_1 = LM_2$, it does not matter which name we use, but in computations it is more effective to use LM_2 than LM_1. This is important as we have to perform a grid search to find the most optimal value of $\gamma = (\gamma_1, \gamma_2)$.

3.1.5 The SupLM statistic

From now on, we denote the LM statistic by LM. Note that LM depends on both the cointegration value β and the threshold parameter $\gamma = (\gamma_1, \gamma_2)$, i.e., $LM = LM(\beta, \gamma)$. If β is known a priori, we use this known value β_0. For example, when we are testing for nonlinear mean reversion in interest rates, we may let $\beta_0 = 1$ such that the error correction term becomes $w_{t-1}(\beta_0) = \beta_0' y_t = \begin{bmatrix}1 & -1\end{bmatrix} y_t = y_{t1} - y_{t2}$, i.e., the difference between the two interest rates. The LM statistic is computed under H_0 such that when β is not a priori known, we use the estimate of β under H_0, i.e., the least squares estimate $\hat{\beta}$ in the linear VECM (3.1). However, we have not any value of γ under H_0. In addition, the least squares estimation of the parameters in the TVECM (3.2) is valid only for fixed γ. Therefore, we can't use $LM(\hat{\beta}, \gamma)$ as test statistic when testing H_0 against H_1. As in Hansen and Seo (2002) and Seo (2003), we use the SupLM statistic which does not depend on the nuisance parameter γ:

$$
\mathrm{SupLM} = \sup_{\gamma \in \Gamma_p} \mathrm{LM}(\hat{\beta}, \gamma)
$$

where $\Gamma_p = \{(\gamma_1, \gamma_2) \mid P(w_{t-1}(\hat{\beta}) \le \gamma_1) \ge p, P(\gamma_1 < w_{t-1}(\hat{\beta}) \le \gamma_2) \ge p, P(w_{t-1}(\hat{\beta}) > \gamma_2) \ge p\}$, and p is a trimming parameter which typically is set to 0.05, 0.10 or 0.15. As $\mathrm{LM}(\hat{\beta}, \gamma)$ is a highly irregular function, this maximization has to be performed by using a grid search.

3.2 The implementation of the SupLM test in the case of three regimes

In the function `TVECM.HStest` in the R package **tsDyn**, the SupLM statistic is implemented for the case of a two-regime TVECM. We extend this to the case of a three-regime TVECM. This involves using the formula for LM_2 in Proposition 3.5, and a new algorithm for searching for the largest $LM_2(\beta, \gamma)$ when $\gamma \in \Gamma_p$. The function `TVECM.XHStest` is our extension and improvement of the function `TVECM.HStest`. It computes the set of possible threshold values in the following way:

1. As in `TVECM.HStest`, the error correction term vector $[w_0(\beta), w_1(\beta), \dots, w_{T-1}(\beta)]$ is sorted in ascending order and stored in the vector `allgammas`.

2. The minimum number of w_t's in each regime is computed by `Ttrim`= $\lceil Tp \rceil$ where T is the size of the dataset as before, and p is the trimming parameter. Then, `Min1=allgammas[Ttrim]` is the smallest possible value of the largest w_t in the lower regime if the lower regime should contain at least $100p$ % of the data. Correspondingly, `Max1=allgammas[T-Ttrim+1]` is the largest possible value of the smallest w_t in the upper regime if the upper regime should contain at least `Ttrim` w_t's. Here we use the `ceiling` function to assure that the lower and upper regime contain at least $100p$ % of the w_t's, while in `TVECM.HStest` the `round` function is used such that the lower and upper regime may contain slightly less than $100p$ % of the w_t's.

3. Next, we remove all duplicates in `allgammas`. In `TVECM.HStest`, this is done by using the R function `unique`. However, `unique` removes an element of the vector only if it is exactly equal to another element in the vector. This means that if two elements in the vector are different only because of inaccuracies in the representation of floating point numbers, then `unique` does not remove such a duplicate. Usually, there are such duplicates in `allgammas` because the error correction term $w_t(\beta) = \beta' x_t$ is computed by doing arithmetic operations on the floating point numbers β and x_t. We remove duplicates in the following way: Let `gammas=allgammas`, i.e., we remove duplicates from the vector `gammas`, while we keep the vector `allgammas` unchanged because we need it later to ensure that the middle regime has enough data. Let `tolerance` be an upper limit for inaccuracies due to computations with floating point numbers. The default value is set to `1e-12`, but the user may specify another value by `tolerance=<value>`. If `gammas[i]-gammas[i-1]`$\leq$`tolerance`, then `gammas[i]` is recognized as a duplicate of `gammas[i-1]`, and hence is removed from the vector `gammas`. When this is done for each $i = 2, 3, \dots, T$, the difference between any two consequtive elements in `gammas` is $\geq$`tolerance`, i.e., all duplicates (according to our definition of duplicates) are removed from the vector `gammas`.

4. Next, we compute the index `iMin` in `gammas` of `Min1` and the index `iMax` in `gammas` of `Max1`. Then, we have that `gammas[iMin]` is the smallest possible value of the largest w_t in the lower regime, and that `gammas[iMax]` is the largest possible value of the smallest w_t in the upper regime. Thus, the lower threshold γ_1 has to be $\geq$ `gammas[iMin]`, and the upper threshold γ_2 has to be $\leq$ `gammas[iMax-1]`. (We have to subtract 1 in the index because the upper regime is defined by a strong inequality, while the lower regime is defined by an $\leq$ inequality.)

5. Now, the vector `gammas=gammas[iMin:(iMax-1)]` contains all the possible threshold values. If the number of possible threshold values, i.e., the length of the vector `gammas`, is less than `ngridTh`, the user specified number of different threshold values in the grid, then `ngridTh` is set equal to the length of `gammas`, and a warning with the new value of `ngridTh` is displayed. Then, each slot in the grid contains one element. If we had used the user specified value of `ngridTh`, then some of the slots in the grid would have been empty, so this refinement of the grid does not change SupLM.

6. If the length of `gammas` is larger than `ngridTh`, then `ngridTh` elements of `gammas` are selected as in `TVEM.HStest` by the R command

   ```
   gammas=gammas[round(seq(from=1,to=length(gammas),length.out=ngridTh))]
   ```
   (3.8)

   The function `seq` makes a sequence of `ngridTh` elements equally spaced between 1 `length(gammas)`, and the function `round` rounds this sequence of decimal numbers to integers. Note that if the parameter `ngridTh` is chosen smaller than the length of the `gammas` vector, then a further constraint is introduced when maximizing $LM = LM(\beta, \gamma)$, and this constraint depends not only on the value of `ngridTh`, but also on how the `ngridTh` possible threshold values are selected from the `gammas` vector, i.e., on the R command (3.8). Naturally, this constraint is removed if `ngridTh` is set equal to the length of the time series, though, this may be very time-consuming if combined with bootstrapping.

7. We add `tolerance` to `gammas` to ensure that w_t is moved to the correct regime when w_t is slightly larger than the current threshold value.

Now, `gammas` contains all the threshold values which should be used in the grid when searching for the SupLM value. If the user has specified `type="2Reg"`, i.e., the model has two regimes, this is a single grid, i.e., as in `TVECM.HStest`, we compute $LM(\beta, \gamma)$ for each γ in `gammas`, and SupLM is the largest of these values. (If the user has specified a fixed β by `fixed.beta=<value>`, this value is used when $LM(\beta, \gamma)$ is computed. If the user has not specified a fixed β, we use the least squares estimate $\hat{\beta}$ of β from the linear VECM model, computed by the function `VECM` in package **tsDyn**.

If the user has specified `type="3Reg"`, i.e., the model have three regimes, we use a double `for` loop to set up the grid: The outer `for` loop controls the index `i` of the lower threshold γ_1 in `gammas`. The index `i` runs from 1 to `iMax` which is determined below. The inner `for` loop controls the index `j` of the upper threshold γ_2 in `gammas`. This index runs from `jMin` to `ngridTh`. We require that the middle regime contains at least $100p$ % of the data when `iMax` and `jMin` is determined. We compute `iMax` in the following way:

1. We find the index in `allgammas` of the smallest element in `allgammas` equal to `allgammas[T-Ttrim]` except for floating point error by using the R command `i1=which(abs(allgammas-allgammas[T-Ttrim])<tolerance,arr.ind=TRUE)[1]`. Then, `allgammas[i1]` is the largest w_t in the middle regime when the upper regime contains at least `Ttrim` elements. If also the middle regime contains at least `Ttrim` elements, then the the largest w_t in the lower regime is `Max2=allgammas[i1-Ttrim]`.

2. The upper limit for the index *i* is now computed by the R command `iMax=max(which(gammas<=Max2,arr.ind=TRUE))`.
   Note that we have to select the largest element in `gammas` less than or equal to `Max2` as there may be no element in `gammas` equal to `Max2` because we have removed elements from `gammas`.

The lower limit `jMin` of *j* has to be determined when the index *i* and γ_1 have got their values. The aim is to choose the starting point of γ_2 so large that the middle regime contains at least `Ttrim` w_t's when the lower threshold γ_1 is fixed to `gammas[i]`. We compute `jMin` in the following way.

1. First, we compute the set of indices of all the elements in `allgammas` which are approximately equal to `gammas[i]` by using the following R command: `help=which(abs(allgammas-gammas[i])<tolerance,arr.ind=TRUE)`.
   Then `help[length[help]]` is the largest element in `allgammas` which is approximately equal to `gammas[i]`, i.e., the largest w_t moved to the lower regime for this value of γ_1.

2. Next, we compute the least possible value of the the largest w_t moved to the middle regime for this value of γ_1 by using the R command `Min2=allgammas[help[length(help)]+Ttrim]-tolerance`.
   The term `+Ttrim` assures that the middle regime has at least `Ttrim` elements. The term `-tolerance` is added to get the correct answer in the following test `gammas>Min2` when `Min2` is approximately equal to an element in `gammas`. This is necessary because by construction, `gammas` contains only the smallest element of a group of elements in `allgammas` which at most differ with `tolerance` in absolute value.

3. Next, we test whether `gammas(ngridTh)>Min2`. If not, the value `gammas[i]` of γ_1 is so large that there is no possible choices of γ_2 such that the middle regime contains at least `Ttrim` w_t's, so we skip this value of `i`.

4. If `gammas(ngridTh)>Min2`, we compute `jMin` by the R command

   ```
   jMin<-min(which(gammas >= Min2,arr.ind=TRUE))
   ```

   which is the index of the least element in `gammas` greater than or equal to `Min2`

3.3 Summary and concluding remarks

In this chapter we have generalized the LM statistic described in Hansen and Seo (2002) and the LM statistic used in the function `TVECM.HStest` to the case of three regimes in the alternative hypothesis, and we have shown that these LM statistics are equal under certain conditions. Correspondingly, we have generalized the grid search for maximizing this LM statistic to the case of three regimes, and the grid search is improved such that in the case of known cointegration value β, it really finds the global maximum of $\mathbf{LM}(\gamma_1, \gamma_2)$ when the two thresholds γ_1 and γ_2 vary among all the possible threshold values if the parameter `ngridTh` is greater than or equal to the length of the time series. However, our implementation, the function `TVECM.XHStest`, is very time consuming, especially when the number of bootstrap replications is e.g., 1000 to get an accurate estimate of the P-value of the test. Fortunately, the time consumption is considerably reduced when modeling long time series of interest rates, if we use the cointegration value $\beta = 1$ instead of the estimated value of β. This is due to the fact that the number of different values of the threshold variable, which in this case is the lagged value of the difference between the two interest rates, is considerably smaller than the length of the time series.

It is important to be aware of the fact that even when ***ngridTh*** is chosen large such that the whole vector of possible threshold values is used when maximizing $\mathrm{LM}(\gamma_1, \gamma_2)$, we have maximized $\mathrm{LM}(\gamma_1, \gamma_2)$ under the constraint that each group with w_t values which do not differ more than ***tolerance*** in absolute value, are moved as a whole between the regimes. If we drop this constraint, we normally get an even larger value of **SupLM** (and an even smaller value of SSR). But then we regard numbers which differ in value only due to inaccuracies in the floating point number repesentation, as different, which is wrong from an applied perspective. So, we keep this constraint throughout our analysis.

Also, the requirement of at least $100p$ % of the w_t's in each regime is a constraint under the maximization. If we weaken this constraint by reducing p, we get an even larger value of the **SupLM**. However, the value of p should not be chosen too small, because we need enough w_t's in each regime to get a reasonable estimate of the coefficients in each regime.

In addition, the selection of threshold values from ***gammas*** when ***ngridTh*** is chosen less than ***length(gammas)***, involves a further constraint. Fewer threshold values in the grid imply that all w_t's between two consequtive elements in ***gammas*** are grouped together and moved as a whole between the regimes.

The asymptotic distribution of the **SupLM** statistic is nonstandard, so we use bootstrapping to estimate the P-value. In the function ***TVECM.HStest*** two methods of bootstrapping are implemented: residual bootstrapping and fixed regressor bootstrapping. In residual bootstrapping a completely new data set is drawn for each bootstrap replication by using the function ***TVECM.sim***, while in fixed regressor bootstrapping only new Y-values are drawn, i.e., the regressor X is fixed for all bootstrap replications. The fixed regressor bootstrapping is strictly speaking not bootstrapping, but an approximation of the asymptotic distribution of the **SupLM** statistic using the White-Eicker heteroscedastic consistent covariance estimator. On the contrary, the residual bootstrapping is real bootstrapping, but presupposes homoscedasticity. In ***TVECM.XHStest*** we have kept all the bootstrapping capabilities in ***TVECM.HStest*** unchanged.

A consequence of the double grid to search for the largest $\mathrm{LM}(\gamma_1, \gamma_2)$ is that even when the number of possible threshold values is only 200, the elapse time to compute the **SupLM** is about half a minute. As we have to repeat the computation of **SupLM** statistic for each bootstrap replication, the elapse time of the function ***TVECM.XHStest*** is several hours when the number of bootstrap replications is e.g., 1000, which is often recommended when estimating P-values by bootstrapping. Above, we have seen that leaving out w_t-values from the set of possible threshold values implies introducing a further constraint when maximizing $\mathrm{LM}(\gamma_1, \gamma_2)$. Therefore, we should use exactly the same constraints , i.e., exactly the same parameters ***ngridTh***, ***trim*** and ***tolerance***, when computing the **SupLM** statistic, and when bootstrapping. Preferably, ***ngridTh*** greater than or equal to the length of the time series should be combined with at least ***nboot*** equal to 1000, but if this is infeasible, a test run should be done with such a large ***ngridTh*** and a small value of ***nboot*** to measure the influence on the **SupLM** statistic of including all the w_t-values in the set of possible threshold values.

Chapter 4

Analysis of the NIBOR rates of maturities tomorrow next and 12 months

In this chapter we analyse the bivariate time series consisting of the monthly averages of NIBOR rates of the two maturities tomorrow next and 12 months. We analyse both this bivariate time series by using functions for multivariate time series analysis, and the term spread of these NIBOR rates by using functions for univariate time series analysis. In addition, we test the influence of some outliers by removing them from these time series and rerunning some of the statistical tests. At last, we run the Johansen cointegration rank test on all the possible pairs of interest rates which may be made from the NIBOR rates of 9 different maturities and rates on Norwegian government bonds of 3 maturities, and we test the term spread of these pairs for threshold effects.

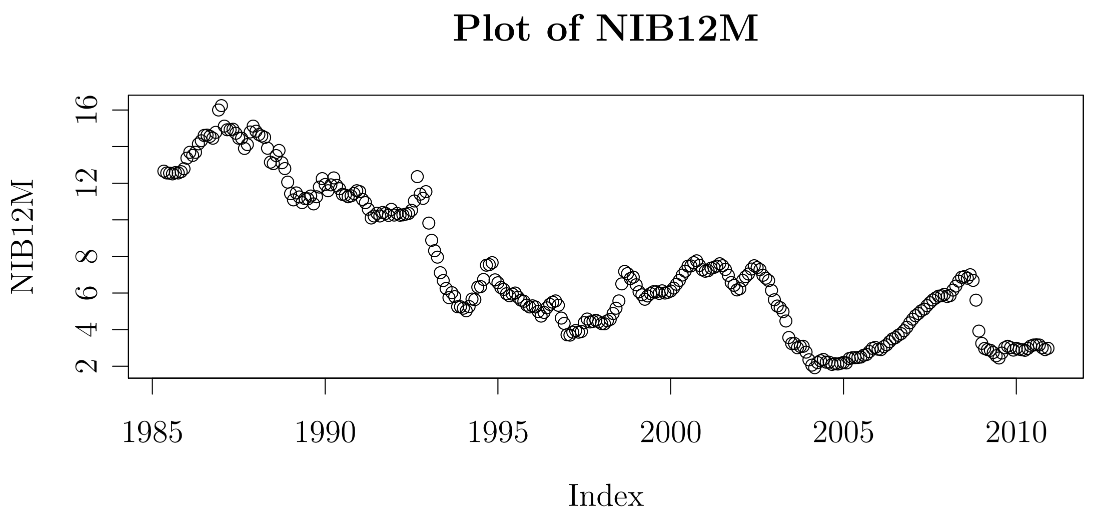
All the models and tests we have used in the data analysis in this chapter, are described in Chapter 2 and 3. The data analysis is performed by using `R`, and all the `R` code chunks needed for this data analysis are included in Appendix C.

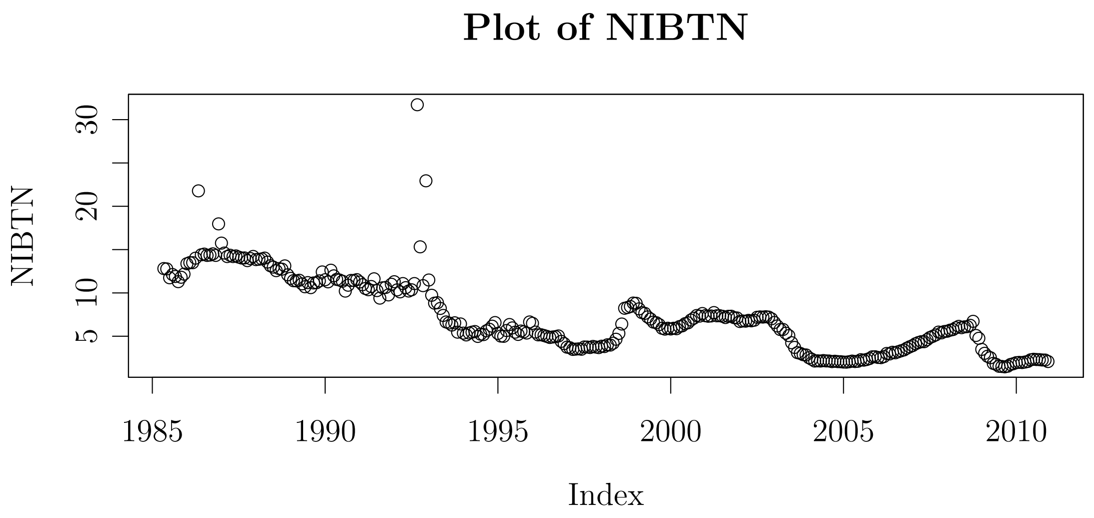
4.1 The data set

In our analysis we use the `NIBOR TN` (Norwegian Inter Bank Offered Rate tomorrow next, monthly averages of daily observations of the nominal interest rate) as the short rate, and `NIBOR 12M` (Norwegian Inter Bank Offered Rate 12 months, monthly averages of daily observations of the nominal interest rate) as the long rate. We create the time series objects *`NIBTN`* and *`NIB12M`* of the `NIBOR TN` and `NIBOR 12M` rates downloaded from www.norges-bank.no. These time series contain all the monthly observations from May 1985 to December 2010, i.e., 308 observations in each time series.

First of all, we plot these two time series. In Figure 4.1 on the next page we see that *`NIBTN`* and *`NIB12M`* have 6 and 2 outliers, respectively. With the purpose of being able to test the influence of these outliers, we make new time series objects *`TestNIBTN`* and *`TestNIB12M`* where these outliers are replaced by interpolated values between adjacent

Figure 4.1. Plots of the time series *NIB12M* and *NIBTN*.



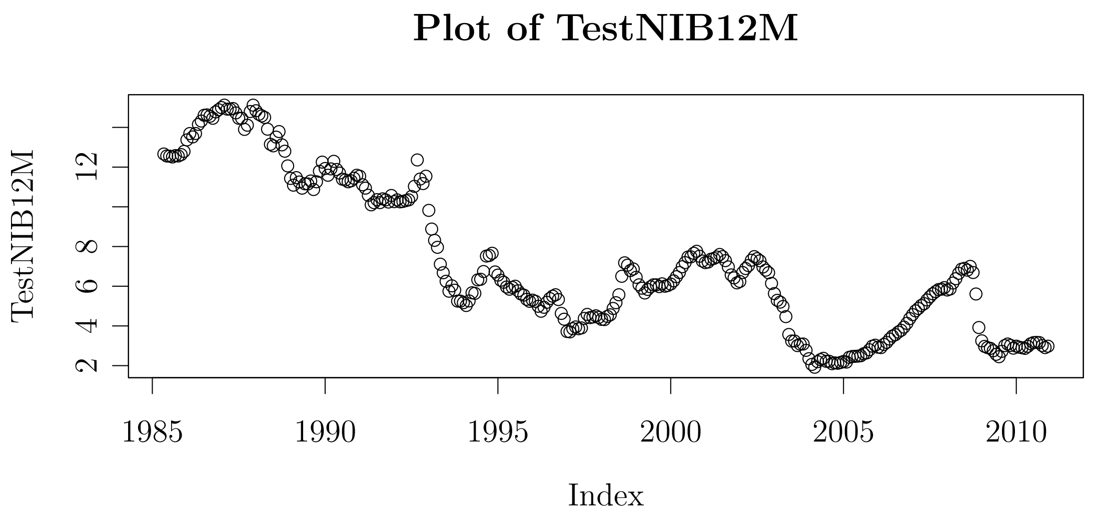


values of the outliers. Figure 4.2 shows the plots of these time series where the outliers are removed.

4.2 Estimation of an unrestricted VAR model

As a starting point, we estimate an unrestricted VAR model. Table 4.1 on the next page shows the coefficients of the estimated VAR model. Note that the intercept term in each equation is insignificant. So, we could have estimated an unrestricted VAR model without constant term. However, in the TVECM estimated in Table 4.16 on page 46, the constant term is significant. Therefore, we have kept the constant term throughout our analysis.

Figure 4.2. Plots of the time series *`TestNIB12M`* and *`TestNIBTN`* where the outliers are removed.



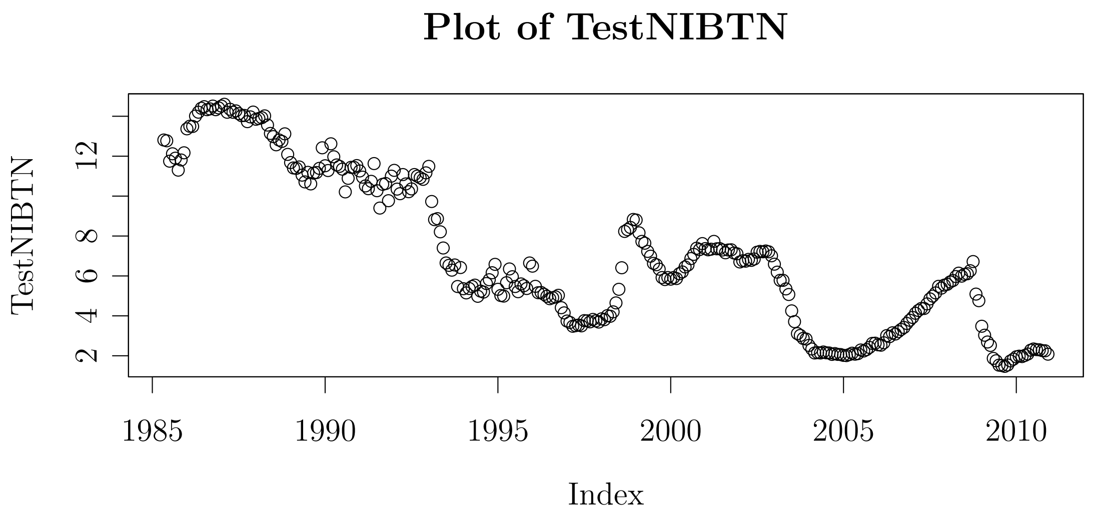


Table 4.1. The coefficients of the estimated VAR model. The standard errors of the coefficients are in the parenthesis. The symbols '***', '*' and '.' denote significance at the 0.1 %, 1 % and 10 % level, respectively.

	Equation NIB12M	Equation NIBTN
Intercept	0.0044(0.0370)	-0.2158(0.1816)
NIB12M -1	1.5871(0.0621)***	1.5928(0.3049)***
NIBTN -1	-0.0820(0.0118)***	0.1077(0.0578).
NIB12M -2	-0.6357(0.1062)***	-1.1195(0.5214)*
NIBTN -2	0.0296(0.0130)*	0.0731(0.0640)
NIB12M -3	0.0942(0.0638)	0.0059(0.3133)
NIBTN -3	0.0036(0.0123)	0.3659(0.0602)***

When a model is estimated, it is of crucial importance to test whether the residuals obey the model's assumptions. So, we test for ARCH effects, nonnormality and serial correlation in the residuals. Table 4.2 shows the results of the ARCH and normality tests. We see that in the multivariate ARCH test the null hypothesis of no ARCH effects is strongly rejected. However, in the univariate ARCH tests for the time series ***NIBTN*** and ***NIB12M***, the null hypothesis of no ARCH effects cannot be rejected at any reasonable significance level. We also see that the null hypothesis of Gaussian error terms is strongly rejected; there are both skewness and kurtosis.

Table 4.3 shows the results of the Portmanteau test of the estimated VAR model for two different values of the parameter *K*, the lag order of the VAR model. We see that the null hypothesis of no serial correlation is rejected even at the 1 % level when *K=2*, but when *K=3*, the null hypothesis of no serial correlation is no longer rejected at any reasonable significance level. This shows that *K=3* lags in the VAR model are sufficient to remove serial correction in the residuals.

Table 4.2. The ARCH and normality tests of the estimated VAR model of ***NIB12MTN***.

	Statistic	df	p-value
Multivariate ARCH-LM test	131.7	45	1.96e-10
ARCH-LM test of NIB12M	23.4	16	0.105
ARCH-LM test of NIBTN	1.1	16	1
Multivariate JB test	190005.2	4	0
Multivariate Skewness test	4247.7	2	0
Multivariate Kurtosis test	185757.4	2	0
JB test of NIB12M	133.6	2	0
JB test of NIBTN	244304.2	2	0

Table 4.3. The Portmanteau test of the estimated VAR model of ***NIB12MTN*** when the parameter *K* has value 2 and 3.

	Statistic	df	p-value
K=2	105.6	56	6.98e-05
K=3	58.9	52	0.237

4.3 Testing for unit roots

The largest root of the characteristic polynomial of the estimated VAR model above is 0.9879 , i.e., very close to 1, so it is necessary to test whether there are unit roots in this model. Hence, we apply the test procedure described in Section 2.3.1 on page 7. We start with testing whether the time series ***NIBTN*** contains a unit root or not. Table 4.4 shows the values of the test statistics and their critical values belonging to the test regression (2.4). The value $\tau_3 = -3.18$ means that the hypothesis $H_0 : \pi = 0$ cannot be rejected at the 5% significance level. The value $\phi_3 = 5.08$ means that the hypothesis $H_0 : \beta_2 = \pi = 0$ cannot be rejected at the 10% significance level. Further, $\phi_2 = 3.63$ means that the hypothesis $H_0 : \beta_1 = \beta_2 = \pi = 0$ cannot be rejected at the 10% significance level. So, the time series ***NIBTN*** contains a unit root without trend. Plots of the residuals, the autocorrelations of the residuals and the partial autocorrelations of the residuals in this test regression are in Figure 4.3 on the next page. We see that the number of lags selected by the AIC criterium is sufficient to remove autocorrelation in the residuals.

Next, we test whether this model has a drift term by performing the test regression (2.5). Table 4.5 shows the values of the test statistics and their critical values belonging to this test regression. We see that for each of these two tests, the null hypotheis cannot be rejected at any reasonable significance level. Consequently, we conclude that the ***NIBTN*** time series does contain a unit root, but neither a linear trend nor a drift term is present, i.e., the time series ***NIBTN*** is a pure random walk.

Finally, we test whether the series is $I(2)$ by using the ADF test on the differenced series. Table 4.6 on the next page shows the values of the test statistics and their critical values in the test regression (2.4) performed on the differenced series. We see that the null hypothesis of a unit root is strongly rejected. Consequently, the original series ***NIBTN*** is not $I(2)$, i.e., it is $I(1)$.

As an alternative, we may test whether the ***NIBTN*** series is level-stationary or trend-stationary by using the KPSS test described in Section 2.3.2 on page 9. Table 4.7 on the next page shows the values of the test statistics and their critical values. The null hypothesis of a level-stationary series is strongly rejected by the first line in this table, while the null hypothesis of a trend-stationary series is rejected at the 2.5 % significance level by the second line in this table. So, the series ***NIBTN*** is a unit root process.

Table 4.4. ADF test of ***NIBTN***: τ_3, ϕ_2 and ϕ_3 tests

	statistic	1%	5%	10%
τ_3	-3.18	-3.98	-3.42	-3.13
ϕ_2	3.63	6.15	4.71	4.05
ϕ_3	5.08	8.34	6.30	5.36

Table 4.5. ADF test of ***NIBTN***: τ_2 and ϕ_1 tests.

	statistic	1%	5%	10%
τ_2	-1.83	-3.44	-2.87	-2.57
ϕ_1	2.00	6.47	4.61	3.79

Figure 4.3. Diagnostic plots for the ADF test of the time series *`NIBTN`*.

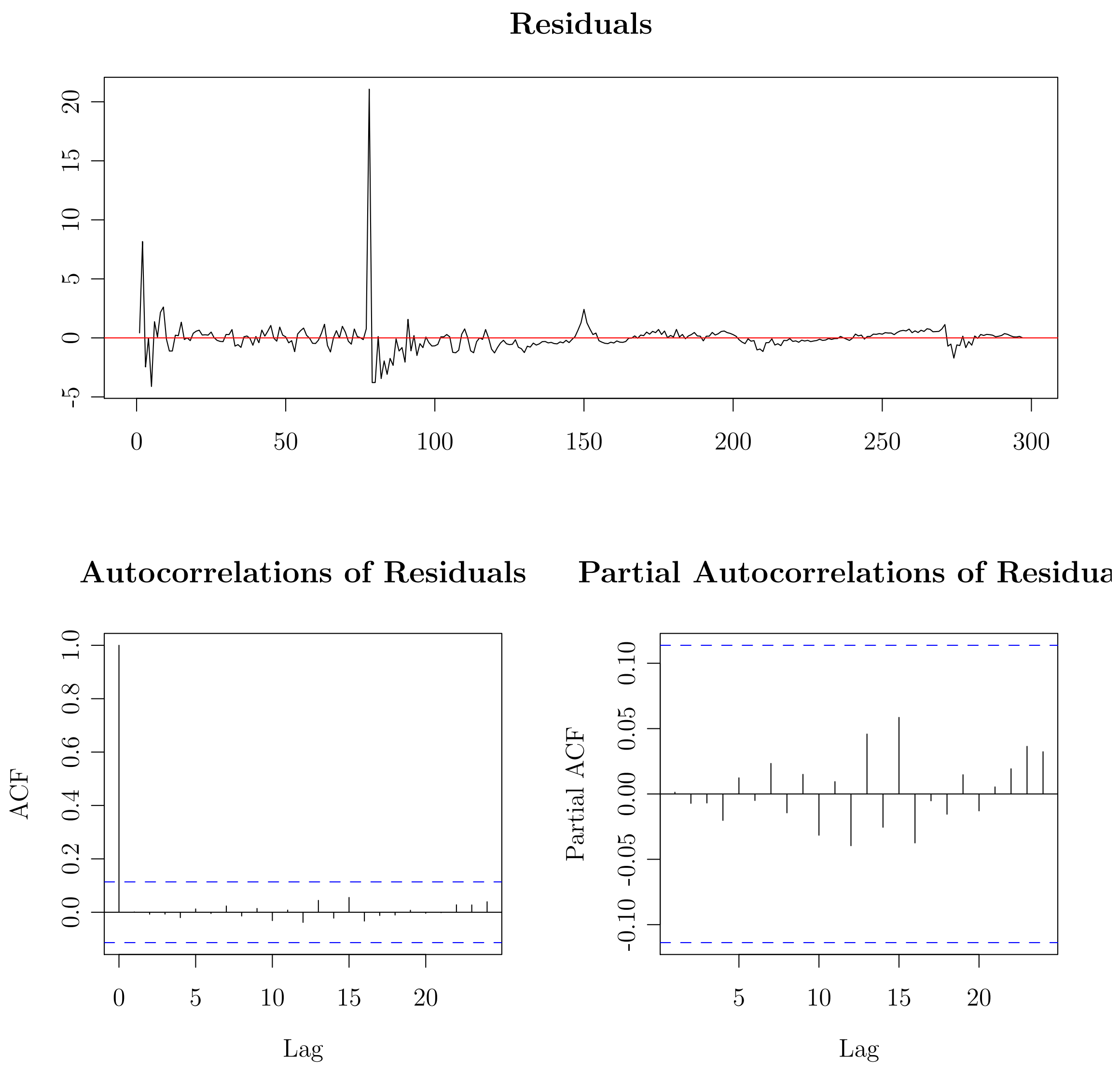


Table 4.6. ADF test of *`diff(NIBTN)`*: τ_3, ϕ_2 and ϕ_3 tests.

	statistic	1%	5%	10%
τ_3	-8.48	-3.98	-3.42	-3.13
ϕ_2	23.96	6.15	4.71	4.05
ϕ_3	35.93	8.34	6.30	5.36

Table 4.7. KPSS test of *`NIBTN`*: $\hat{\eta}_\mu$ and $\hat{\eta}_\tau$ tests.

	statistic	10%	5%	2.5%	1%
$\hat{\eta}_\mu$	1.55	0.35	0.46	0.57	0.74
$\hat{\eta}_\tau$	0.18	0.12	0.15	0.18	0.22

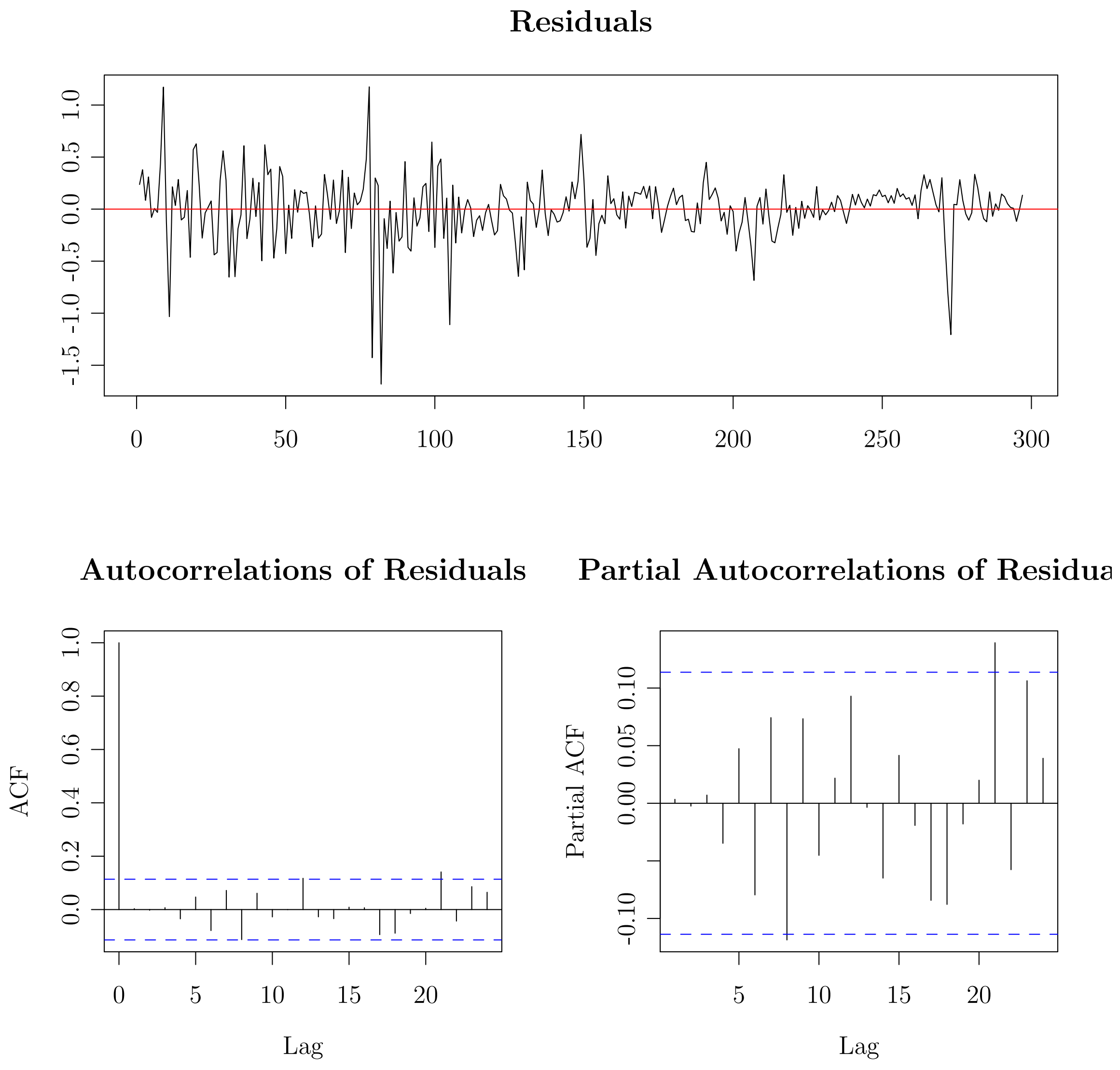
Next, we show in the same way that the time series *NIB12M* is $I(1)$. Table 4.8 shows the values of the three test statistics and their critical values in the test regression (2.4). We see that the null hypothesis in each of the three tests cannot be rejected at any reasonable significance level. So, the time series *NIB12M* contains a unit root without trend.

Next, we test whether this model has a drift term by performing the test regression (2.5). Table 4.9 on the next page shows the values of the test statistics and their critical values. The value $\phi_1 = 1.97$ means that the null hypothesis of a unit root without trend and drift

Table 4.8. ADF test of *NIB12M*: τ_3, ϕ_2 and ϕ_3 tests.

	statistic	1%	5%	10%
τ_3	-2.50	-3.98	-3.42	-3.13
ϕ_2	2.51	6.15	4.71	4.05
ϕ_3	3.22	8.34	6.30	5.36

Figure 4.4. Diagnostic plots for the ADF test of the time series *NIB12M*.



cannot be rejected at any reasonable significance level.

Next, we test whether the series is $I(2)$ by using the ADF test on the differenced series. Table 4.10 shows the values of the test statistics and their critical values. The value $\phi_3 = 27.71$ means that the null hypothesis of a unit root with drift but without trend is rejected at any reasonable significance level. So, the original series `NIB12M` is not $I(2)$, i.e., it is $I(1)$.

As an alternative, we may test whether the `NIB12M` series is level-stationary or trend-stationary by using the KPSS test. Table 4.11 shows the values of the test statistics and their critical values. The null hypothesis of a level-stationary series is strongly rejected by the first line in this table, while the null hypothesis of a trend-stationary series is strongly rejected by the second line in this table. Consequently, the series `NIB12M` is a unit root process.

4.4 Estimation of a VECM

Table 4.1 on page 38 shows that a lot of the coefficients of the unrestricted VAR model are not significant, which means that this model is overparametrized. As we know that the variables `NIBTN` and `NIB12M` both are $I(1)$, we run the cointegration rank test described in Section 2.6.1 on page 15. As a result, we get a VECM which has fewer parameters, and hence, is less overparametrized than the original VAR model. Table 4.12 on the facing page shows the values of the trace statistic and its critical values, while Table 4.13 on the next page shows the values of the $\lambda_{\max}$ statistic and its critical values. Both the trace statistic and the $\lambda_{\max}$ statistic show that the H_0 hypothesis $r = 0$ is strongly rejected. So, we conclude that the cointegration rank is $r = 1$.

Table 4.9. ADF test of `NIB12M`: τ_2 and ϕ_1 tests.

	statistic	1%	5%	10%
τ_2	-1.69	-3.44	-2.87	-2.57
ϕ_1	1.97	6.47	4.61	3.79

Table 4.10. ADF test of `diff(NIB12M)`: τ_3, ϕ_2 and ϕ_3 tests.

	statistic	1%	5%	10%
τ_3	-7.44	-3.98	-3.42	-3.13
ϕ_2	18.47	6.15	4.71	4.05
ϕ_3	27.71	8.34	6.30	5.36

Table 4.11. KPSS test of `NIB12M`: $\hat{\eta}_\mu$ and $\hat{\eta}_\tau$ tests.

	statistic	10%	5%	2.5%	1%
$\hat{\eta}_\mu$	1.55	0.35	0.46	0.57	0.74
$\hat{\eta}_\tau$	0.26	0.12	0.15	0.18	0.22

As an alternative, the cointegration rank may be determined by plotting the cointegration relations as in Section 8.1.3 of Pfaff (2008a). In Figure 4.5 on the following page we see that the first cointegration relation seems to be stationary, while the second relation is undoubtedly nonstationary, which is in agreement with cointegration rank $r = 1$. It does not matter whether we look at the plots of the relations with or without correction for the short-term influences.

Next, we compute the restricted VECM when $r = 1$, and the normalized cointegration relation $\boldsymbol{\beta} = [1 \quad -\beta]'$ by using the function `VECM` in the package **tsDyn** (Di Narzo, Aznarte, and Stigler 2011) or by using the functions `ca.jo` and `cajorls` in the package **urca** (Pfaff 2008a). Table 4.14 shows the coefficients of the estimated VECM with $r = 1$ cointegration relation. We see that a much larger part of these coefficients are significant than the coefficients of the unrestricted VAR model in Table 4.1 on page 38.

4.5 Estimation of a TVECM

In Figure 4.6 on page 45 we have plotted the response of the long rate R_t, the short rate r_t and the term spread $s_t = R_t - r_t$ to the past term spread s_{t-1} as in Seo (2003). (Here, the long rate is `NIB12M`, and the short rate is `NIBTN`.) We see that the current changes in R_t, r_t and s_t are small when $|s_{t-1}|$ is small. However, when $|s_{t-1}|$ is large, the current changes Δr_t and Δs_t are also large. A small proportion of the observations are in this

Table 4.12. The values and the critical values of the trace statistic.

	statistic	10%	5%	1%
r <= 1 |	2.12	6.50	8.18	11.65
r = 0 |	35.39	15.66	17.95	23.52

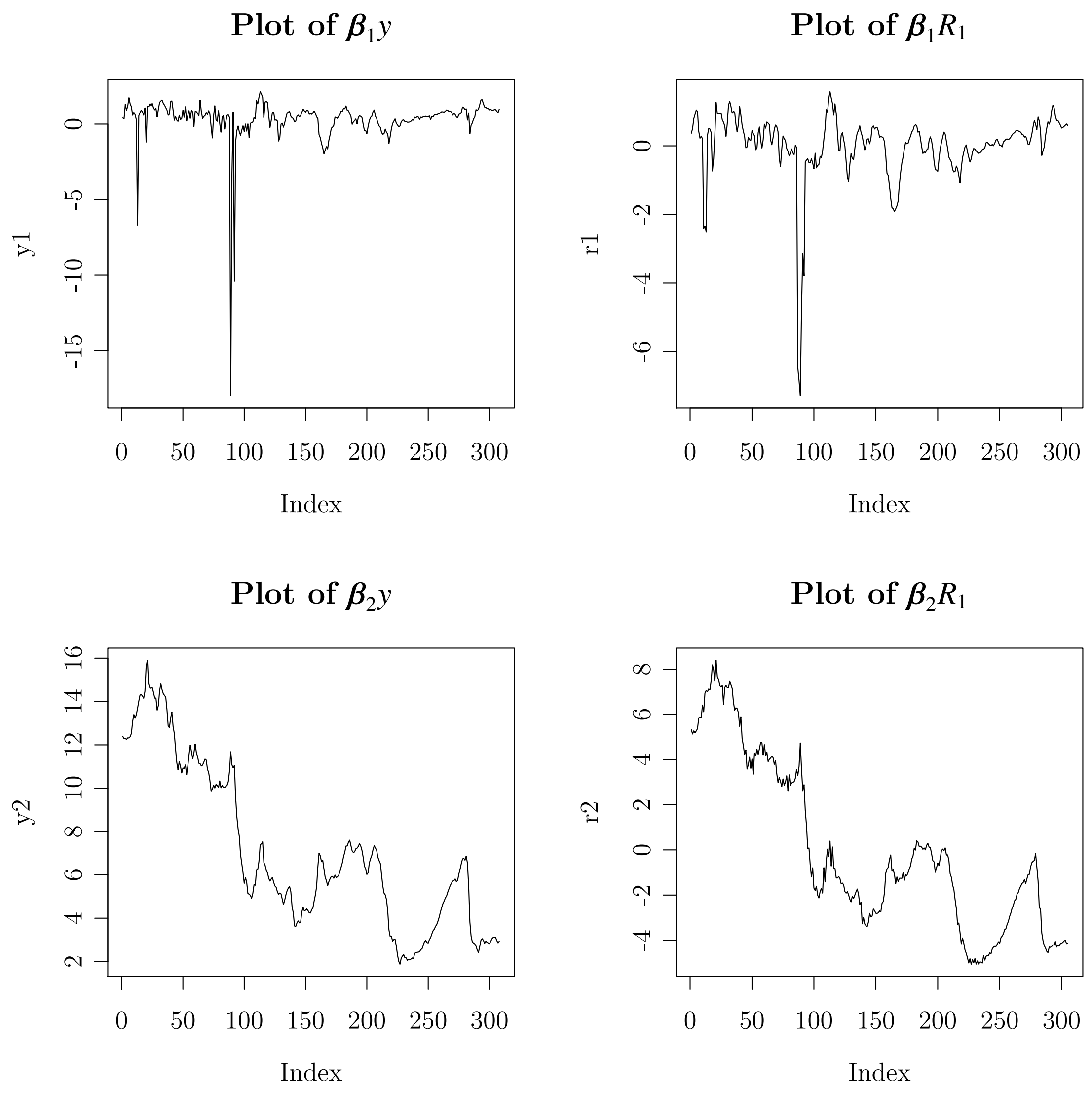
Table 4.13. The values and the critical values of the $\lambda_{\max}$ statistic.

	statistic	10%	5%	1%
r <= 1 |	2.12	6.50	8.18	11.65
r = 0 |	33.27	12.91	14.90	19.19

Table 4.14. The coefficients of the estimated VECM. The standard errors of the coefficients are in the parenthesis. The symbols '***', '**', '*' and '.' denote significance at the 0.1 %, 1 %, 5 % and 10 % level, respectively.

	Equation NIB12M	Equation NIBTN
ECT	0.0512(0.0167)**	0.4739(0.0818)***
Intercept	-0.0353(0.0179)*	-0.1787(0.0878)*
NIB12M -1	0.5420(0.0644)***	1.1132(0.3154)***
NIBTN -1	-0.0334(0.0151)*	-0.4387(0.0741)***
NIB12M -2	-0.0986(0.0638)	-0.0018(0.3123)
NIBTN -2	-0.0035(0.0123)	-0.3659(0.0601)***
Cointegration relation	1	-0.9565

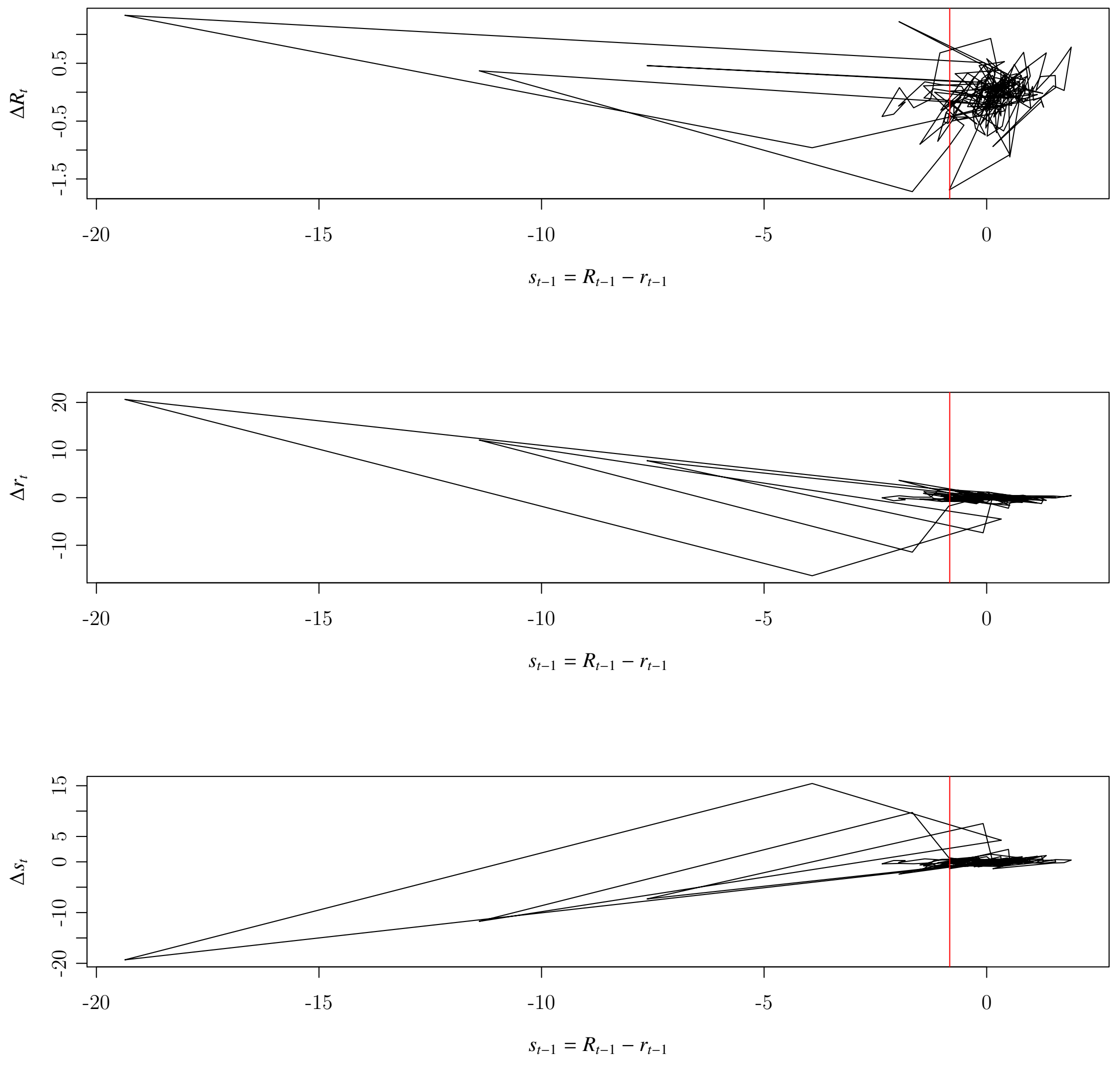
Figure 4.5. Plots of the cointegration relations $\boldsymbol{\beta}_i y$, and those that are corrected for short-term influences, $\boldsymbol{\beta}_i R_1$.



unstable area, but the large current changes in r_t and s_t in the unstable area cause the system to return rapidly into the stable area. So, these time series have a signficant mean reverting property only in the unstable area where s_{t-1} is large negative. Due to this typical nonlinear property, the linear VECM is probably not the best choice for this data set. Therefore, we try a TVECM as in Seo (2003).

When estimating a TVECM, we have to select values of the parameters ***nthresh***, ***lag*** and ***beta***. According to the Expectations Hypothesis, the term spread, i.e., the difference between a long and a short interest rate, is stationary, see e.g., Seo (2003) and Buigut and Rao (2010). In accordance with this, we may fix ***beta=1*** in the estimation of the TVECM. As an alternative, we may find the optimal value of ***beta*** by using the grid search of the ***TVECM*** function in the package **tsDyn**. So, we estimate TVECMs for a small set of choices for these parameters. The results are summarized in Table 4.15 on the next page. Note that these results support the Expectations Hypothesis as for each

Figure 4.6. Plots of ΔR_t, Δr_t and Δs_t against s_{t-1}. The vertical red line in each plot denotes the threshold γ in the estimated TVECM in Table 4.16 on the following page.



pair of values of the parameters ***nthresh*** and ***lag*** the value ***beta=1*** derived from the Expectations Hypothesis, gives AIC and BIC numbers almost as small as the ***beta*** value which we got from the grid search.

However, we see that ***nthresh=1***, ***lag=2*** and ***beta=***1.071 give the smallest value of both AIC and BIC, so we estimate this TVECM once again to get more information about this specific model. Also, we estimate the corresponding TVECM for the data set ***TestNIB12MTN***. Figure 4.7 on the facing page shows the plots of the grid search for β and γ when estimating the model. The plots of SSR as a function of γ and β support the result that $\beta = 1.071$ and $\gamma = -0.83$ give the smallest value of SSR, but there are also other values of β and γ which give almost as small values of SSR. For this TVECM, the parts of the observations in each regime are 0.25 and 0.75 which both are well beyond the trimming parameter 0.1, as it should be. Table 4.16 shows the coefficients of this TVECM. Given the short-run dynamics, i.e., the lagged terms of ΔR_t and Δr_t, the time series ΔR_t and Δr_t and Δs_t are linear functions of s_{t-1} in each regime. Figure 4.8 on page 49 shows the graph of these response functions. We see that when s_{t-1} is large negative, Δr_t

Table 4.15. AIC, BIC and SSR for different TVECMs. The three rightmost columns show the parts of the data in each of the two (three) regimes.

nthresh	lag	β	Parameters	AIC	BIC	SSR	ndown	nmiddle	nup
1	1	1.000	16	-573.8	-510.5	642.2	0.320		0.680
1	1	1.040	16	-571.6	-508.3	623.9	0.451		0.549
1	2	1.000	24	-606.5	-513.5	565.3	0.423		0.577
1	2	1.071	24	-614.7	-521.7	541.0	0.246		0.754
1	3	1.000	32	-598.0	-475.3	554.4	0.319		0.681
1	3	1.105	32	-612.1	-489.4	526.3	0.230		0.770
2	1	1.000	24	-578.7	-481.9	599.9	0.412	0.248	0.340
2	1	1.040	24	-573.4	-476.6	593.1	0.448	0.193	0.359
2	2	1.000	36	-612.6	-471.2	516.3	0.269	0.190	0.541
2	2	1.071	36	-575.9	-434.6	551.2	0.354	0.111	0.534
2	3	1.000	48	-608.9	-423.0	490.7	0.293	0.224	0.484
2	3	1.105	48	-591.8	-405.9	523.1	0.230	0.368	0.401

Table 4.16. The coefficients of the TVECM with $\beta = 1.071$, *lag=2* and threshold value $\gamma = -0.83$. The standard errors of the coefficients are in the parenthesis. The symbols '***', '**', '*' and '.' denote significance at the 0.1 %, 1 %, 5 % and 10 % level, respectively.

	Lower regime		**Upper regime**	
	Equation NIB12M	Equation NIBTN	Equation NIB12M	Equation NIBTN
ECT	0.0932(3.2e-05)***	0.9704(5.1e-18)***	-0.0247(0.5276)	0.0205(0.9124)
Const	0.1112(0.0269)*	1.6485(2.6e-11)***	0.0073(0.7020)	0.0159(0.8605)
NIB12M t -1	0.4983(3.0e-06)***	1.6058(0.0014)**	0.5291(7.5e-11)***	0.8748(0.0194)*
NIBTN t -1	0.0061(0.7892)	-0.0033(0.9759)	0.1867(8.4e-05)***	-0.2750(0.2177)
NIB12M t -2	-0.0902(0.4384)	-0.2677(0.6287)	-0.1092(0.1339)	0.2341(0.4983)
NIBTN t -2	0.0344(0.0975).	0.0518(0.5995)	-0.0801(2.8e-05)***	-0.6300(1.4e-11)***

Figure 4.7. Plot of the grid search for β and the threshold γ in *TVECM(NIB12MTN,nthresh=1,lag=2,...)*.

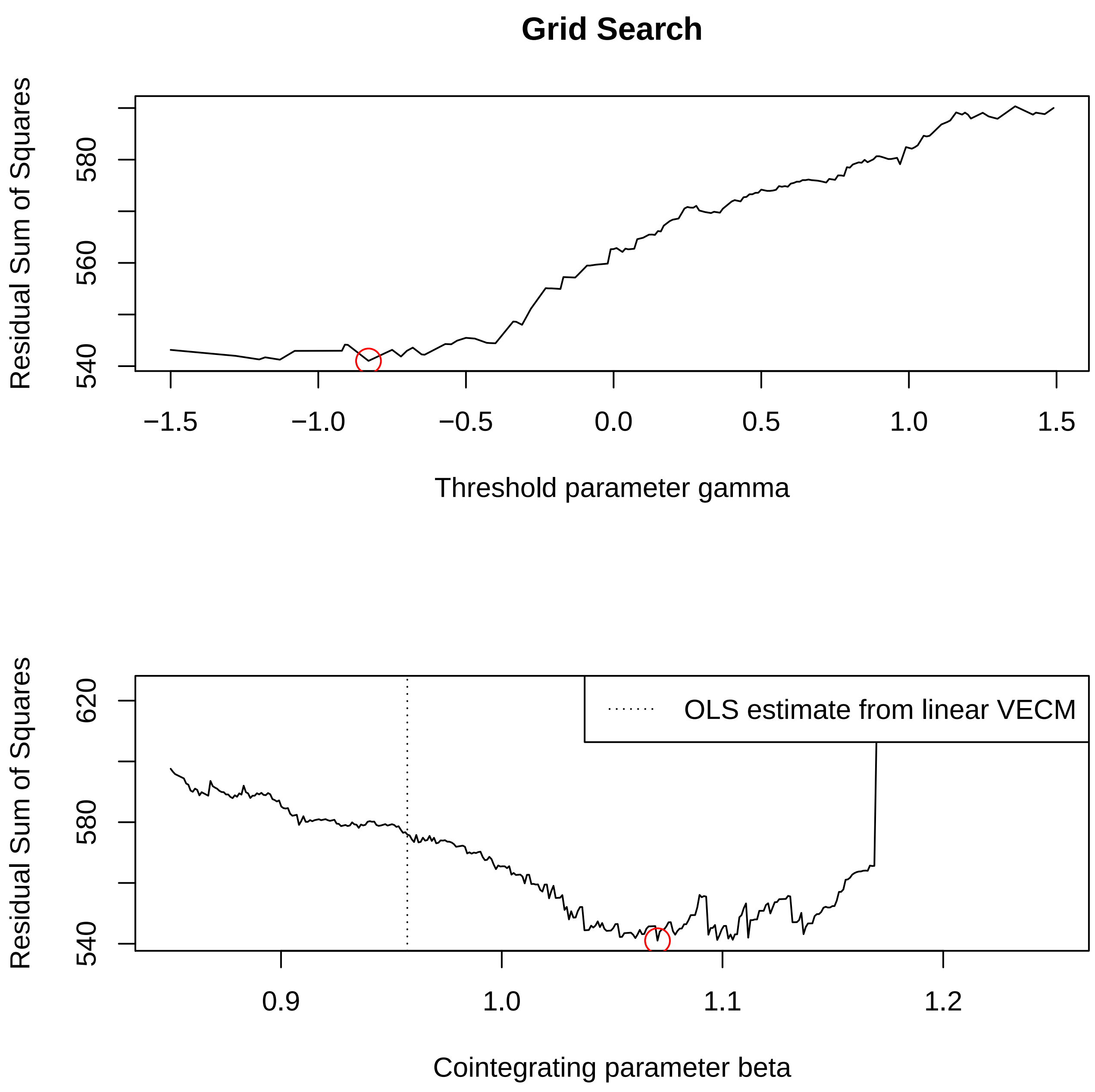


Table 4.17. The results of the ARCH-LM test for the residuals of the estimated VAR(3) model and the estimated TVECM.

	Test statistic	df	P-value
VAR model	121.74	45	0.00000
TVECM	32.71	45	0.91380

Table 4.18. The results of the Doornick-Hansen test for multivariate normality in the residuals of the estimated VAR(3) model and the estimated TVECM.

	VAR model			**TVECM**		
	E	df	P(Chi > E)	E	df	P(Chi > E)
Multivariate	8012.76	4	0.00000	10034.45	4	0.00000
NIB12M	44.78	2	0.00000	32.29	2	0.00000
NIBTN	7967.98	2	0.00000	10002.16	2	0.00000

is large negative and ΔR_t is small negative, such that $s_t = s_{t-1} + \Delta s_t = s_{t-1} + \Delta R_t - \beta \Delta r_t$ is much larger than s_{t-1}. Thus, if we are in the lower regime, which is the unstable area, the error correction term causes a large step towards the upper regime. However, in the upper regime, the coefficients of the constant term and the error correction term are not significant, while most of the coefficients of the lagged terms of ΔR_t and Δr_t are significant, see Table 4.16 on page 46. So, in the upper regime, the bivariate time series $[\Delta R_t \ \Delta r_t]'$ is approximately a stable VAR(2) process.

Similarly, we compute the responses of r_t, R_t and $s_t = R_t - r_t$ from the TVECM model with two thresholds to changes in s_{t-1}. Figure 4.9 on page 50 shows the graph of these response functions. We see that the main difference from Figure 4.8 on the next page is the new narrow middle regime with approximately the same slope for Δr_t as in the lower regime. The upper regime is the stable area as for the TVECM with one threshold. However, it is highly questionable whether we need two unstable areas to the left of the stable regime. This indicates that the TVECM with only one threshold is the best one, but we will investigate this further later on.

Next, we run the Hansen and Seo test by using our function ***`TVECM.XHStest`***. Both fixed regressor bootstrap and residual bootstrap are used in these tests. Table 4.19 shows the result of the Hansen and Seo test of the data set ***`NIB12MTN`*** and ***`TestNIB12MTN`***. We see that for the original data set ***`NIB12MTN`***, the null of a linear cointegration model is rejected even at the 1% significance level irrespective of the bootstrap type and irrespective of whether the alternative model has two or three regimes. However, for the data set ***`TestNIB12MTN`*** where the outliers are removed, we cannot reject the null of linear cointegration at any reasonable significance level. So, for this example it seems like a TVECM may be considered as a tool to take care of outliers. A possible explanation of this phenomenon is as follows:

- The mean reverting property shown in Figure 4.6 on page 45 is much better modeled with a TVECM than with a VECM. By having a large coefficient of the error correction term in the unstable area (the lower regime), we achieve that the time series returns rapidly to the stable area (the upper regime). However, in the upper regime the coefficient of the error correction term is small such that the probably of staying in this regime is large. Table 4.16 on page 46 shows the significant differences between the coefficients of the error correction terms in the two regimes.
- Generally in regression, large outliers influence heavily on the estimated coeffients. So, when we move the outliers to the lower regime, we may expect that we get a better fit in the upper regime in the TVECM than for the corresponding observations in the VECM. Probably, this explains the large differences between the coefficients

Figure 4.8. Plots of the responses of Δr_t, ΔR_t and Δs_t to s_{t-1} in the TVECM with one threshold.

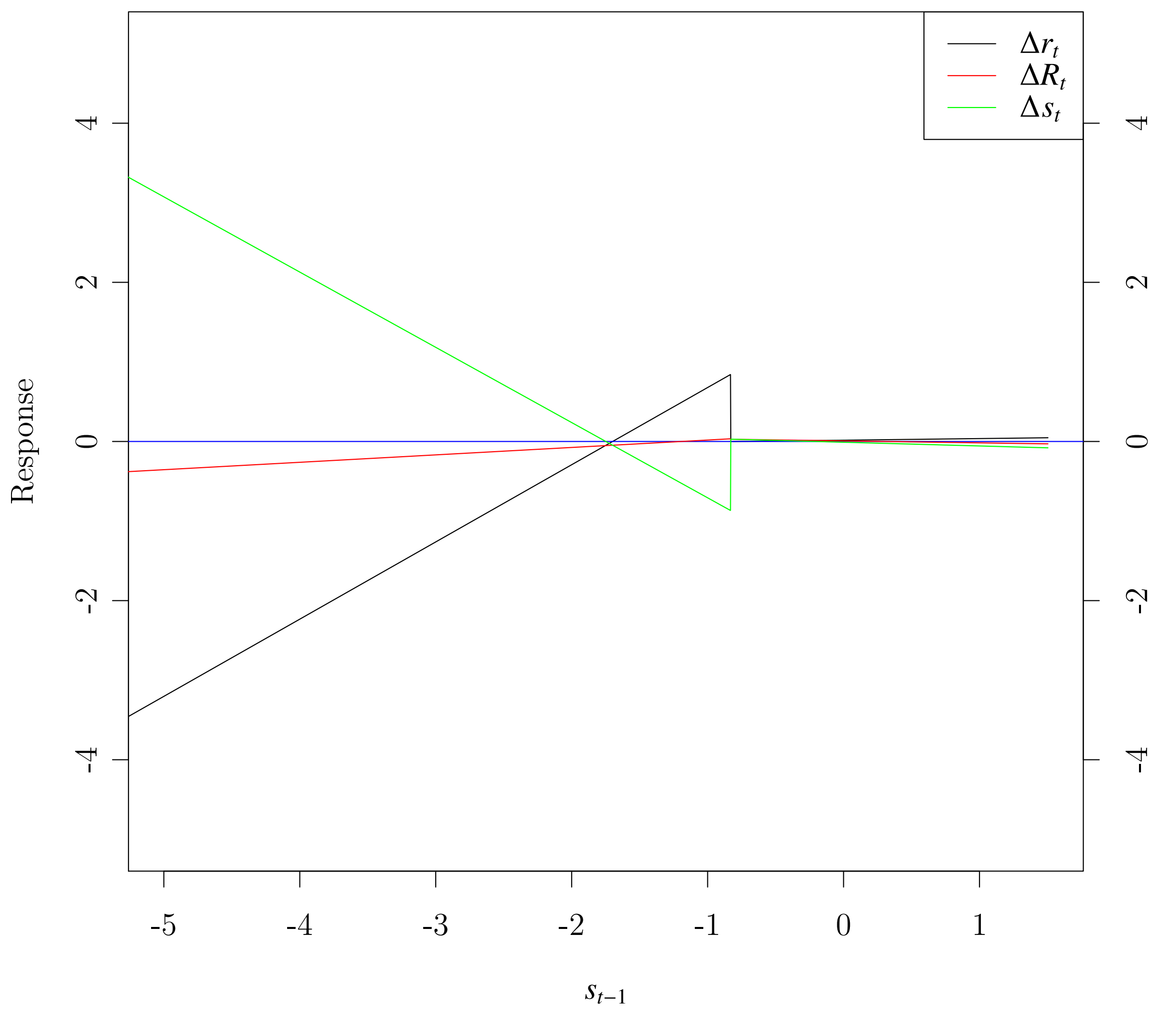
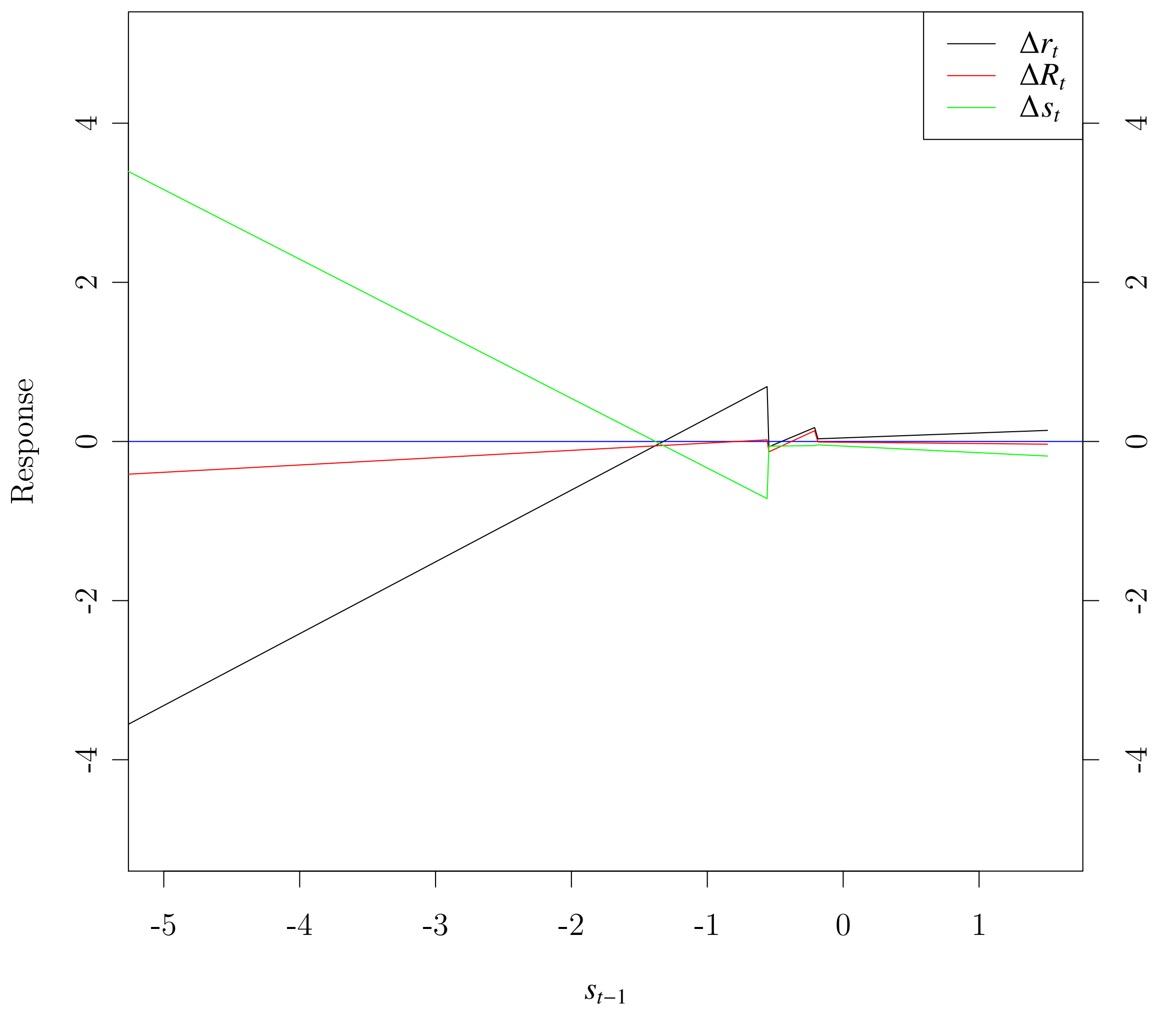


Figure 4.9. Plots of the responses of Δr_t, ΔR_t and Δs_t to s_{t-1} in the TVECM with two thresholds.



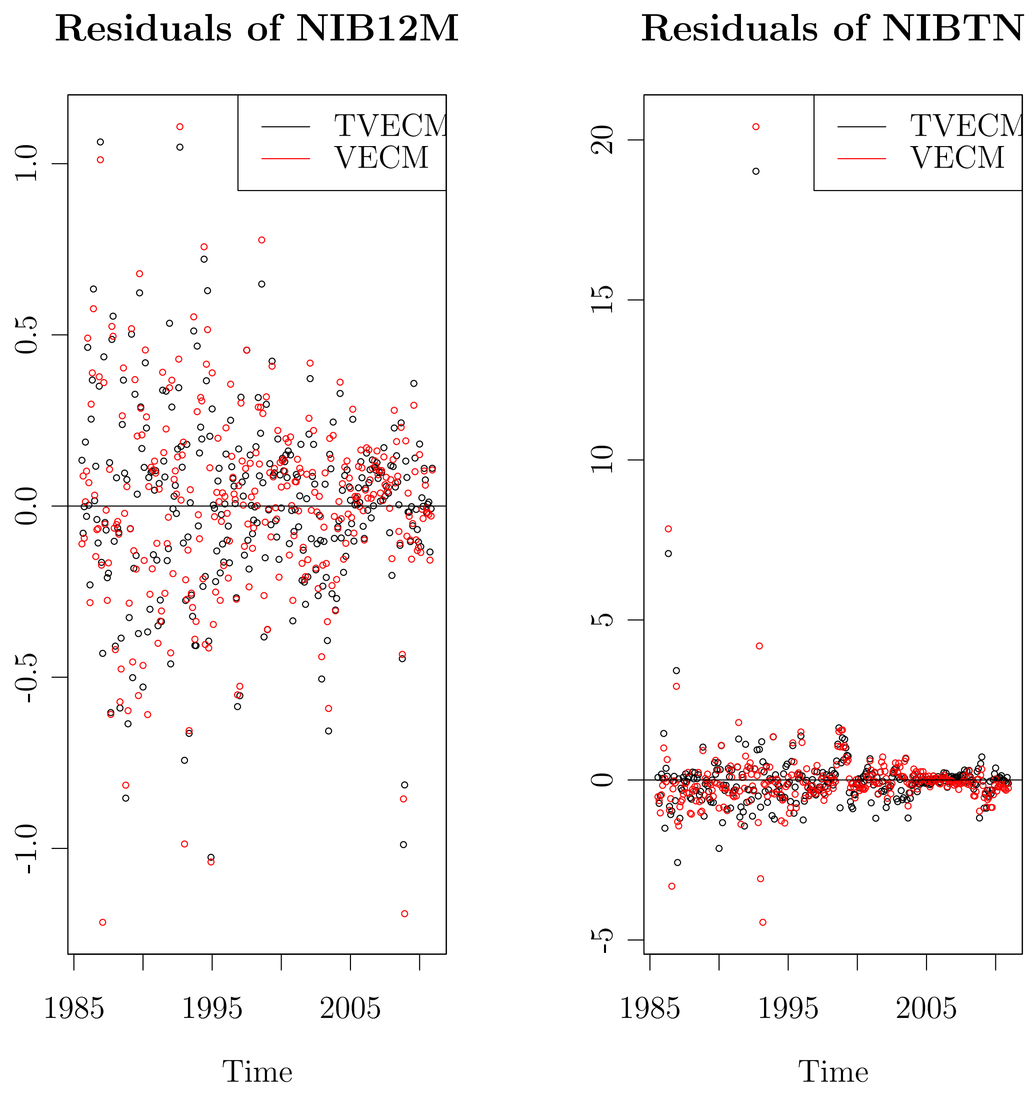
of the VECM in Table 4.14 on page 43 and the coefficients in the upper regime of the TVECM in Table 4.16 on page 46. In addition, we also expect to get a better fit for the observations in the lower regime, when we do not take into account all the observations in the upper regime. This is illustrated in Figure 4.10 on the following page where we have plotted the residuals of the time series `NIB12M` and `NIBTN` for the estimated VECM and TVECM above. We see that the outliers in the VECM are mostly farther away from zero than the outliers in the TVECM. As a result, the SSR of the estimated TVECM is 541 while the SSR of the estimated VECM is 648. Also, there is an improvement in the test of ARCH effects: Table 4.17 on page 47 shows the test statistic for multivariate ARCH effects in the residuals of the estimated VAR model and the estimated TVECM. We see that the null hypothesis of no multivariate ARCH effects in the residuals of the estimated TVECM cannot be rejected at any reasonable significance level, while the null hypothesis of no multivariate ARCH effects in the residuals of the estimated VAR model is strongly rejected as pointed out earlier. However, there is no improvement in the normality test. Table 4.18 on page 48 shows that the null hypothesis of normally distributed errors is strongly rejected both for the residuals of the estimated VAR model and the estimated TVECM.

The function `TVECM.XHStest` also contains a new grid search for the threshold values (γ_1, γ_2) which minimize the SSR. The grid search for the case of two thresholds in the function `TVECM` is a conditional search, i.e., first the function searches for the best value of γ_1, and then it searches for the best value of γ_2 conditional on the value of γ_1. This algorithm is linear in the number of possible threshold values, but it does not necessarily find the global minimum of SSR. On the contrary, the grid search algorithm in `TVECM.XHStest` is quadratic in the number of possible threshold values, and it finds the global minimum of SSR and the global maximum of the LM statistic under the constraints that β is fixed and each of the regimes contain at least as large proportion of the observations as the trimming parameter. We illustrate the improvement of the grid search by gathering the SSR, the threshold values and the percents of observations in each regime from the return data of the `TVECM.XHStest` calls above. The results are shown in Table 4.20 on page 53. Comparing this with Table 4.15 on page 46, we see that for the data set `NIB12MTN` the functions `TVECM` and `TVECM.XHStest` give the minimum SSR 541 and 541.0, respectively, when $\beta = 1.071$, `lag=2` and `nthresh=1`. So, the `TVECM` function has really found the threshold value which minimizes SSR, although the upper plot in Figure 4.7 on page 47

Table 4.19. The results of the Hansen and Seo test for the two data sets `NIB12MTN` and `TestNIB12MTN`.

Data set	nthresh	boot type	nboot	supLM	P-value	Seconds
NIB12MTN	1	FixedReg	1000	29.0	0.020	259.6
	2	FixedReg	1000	56.1	0.001	33496.2
	1	ResBoot	1000	29.0	0.008	360.6
	2	ResBoot	1000	56.1	0.000	33441.0
TestNIB12MTN	1	FixedReg	1000	18.3	0.414	259.9
	2	FixedReg	1000	33.3	0.730	33980.9
	1	ResBoot	1000	18.3	0.530	359.7
	2	ResBoot	1000	33.3	0.796	33738.7

Figure 4.10. Plots of the residuals of *NIB12M* and *NIBTN* in the estimated VECM and TVECM.



does not support this properly. However, in the case of two thresholds, the corresponding minimum SSRs are 551.2 and 458.2, respectively, which clearly shows that the ***TVECM*** function does not find the global minimum of SSR under the given constraints. We also see from Table 4.20 that SSR is minimized when only 11.1 % of the observations are in the middle regime in the TVECM with two thresholds for the data set *NIB12MTN*. This percentage is so close to the trimming parameter, that it is highly questionable whether a TVECM with two thresholds should be used for this data set. We investigate this further in the next section.

4.6 Estimation of a TAR model for the cointegration relation

We know from Seo (2003) that the long-run relationship, i.e., the cointegration relation, in a TVECM may be specified as a threshold autoregressive model (TAR model). If the TVECM has two (three) regimes, then the TAR model for the cointegration relation has two (three) regimes. So, we compute the long-run relationship from the estimated TVECM above, and run the function ***setarTest*** in the package **tsDyn**, which have an option for testing a two-regime TAR model against a three-regime TAR model. Table 4.21 and Figure 4.11 on the next page show the results and the plots, respectively of the ***setarTest*** of ***sNIB12MTN***. We see that the null hypothesis of one regime is rejected at the 5 % significance level when we test against two regimes. On the contrary, when we test two regimes against three regimes, the null hypothesis of two regimes cannot be rejected at any reasonable significance level. So, we conclude that two regimes, i.e., one threshold, is enough when modeling the cointegration relation ***sNIB12MTN***. Hence, also the TVECM for modeling the bivariate time series $[\Delta R_t \; \Delta r_t]'$ should have only one threshold. This is in accordance with Figure 4.9 on page 50 which shows an unstable and too narrow middle regime.

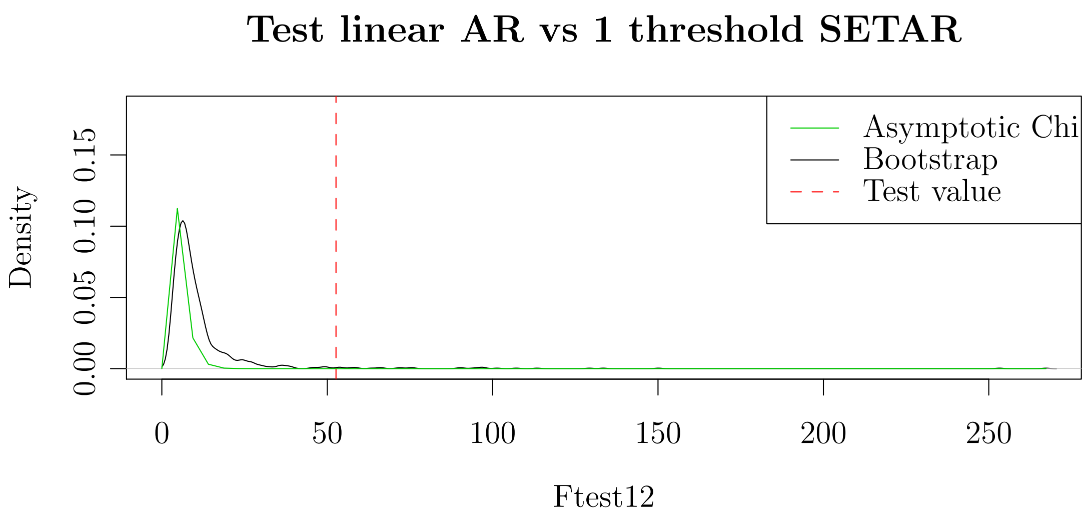
Table 4.22 on page 56 and Figure 4.12 on page 55 show the results and the plots, respectively, of the ***setarTest*** of ***sTestNIB12MTN***. We see that the null hypothesis of one regime cannot be rejected at any reasonable significance level when we test against two and three regimes.

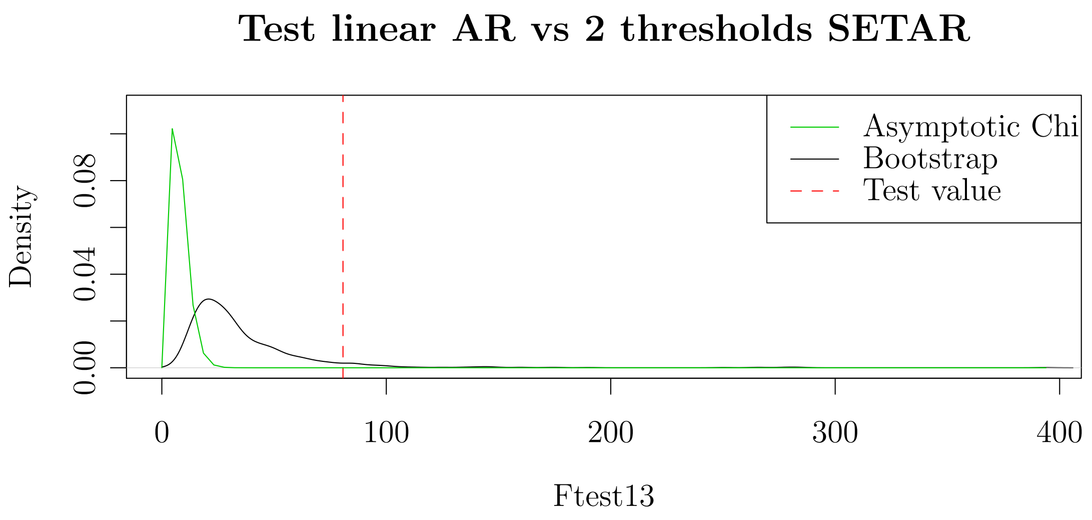
However, a linear autoregressive model with low AR order is not a good choice for this data set because such a model gives a lot of autocorrelation in the residuals. Therefore,

Table 4.20. The minimum SSR, the threshold values and the percents of observations in each regime for the two data sets *NIB12MTN* and *TestNIB12MTN*. The parameter β is in this computation fixed to the values we got from the *TVECM* call above, i.e., $\beta = 1.071$ for *NIB12MTN*, and $\beta = 0.933$ for *TestNIB12MTN*.

Data set	nthresh	SSR	γ_1	γ_2	ndown	nmiddle	nup
NIB12MTN	1	541.0	-0.832		24.6		75.4
	2	458.2	-0.869	-0.561	23.9	11.1	64.9
TestNIB12MTN	1	64.1	1.053		75.7		24.3
	2	61.5	0.678	1.053	48.5	27.2	24.3

Figure 4.11. Plots of the *SETAR* tests of *sNIB12MTN*.





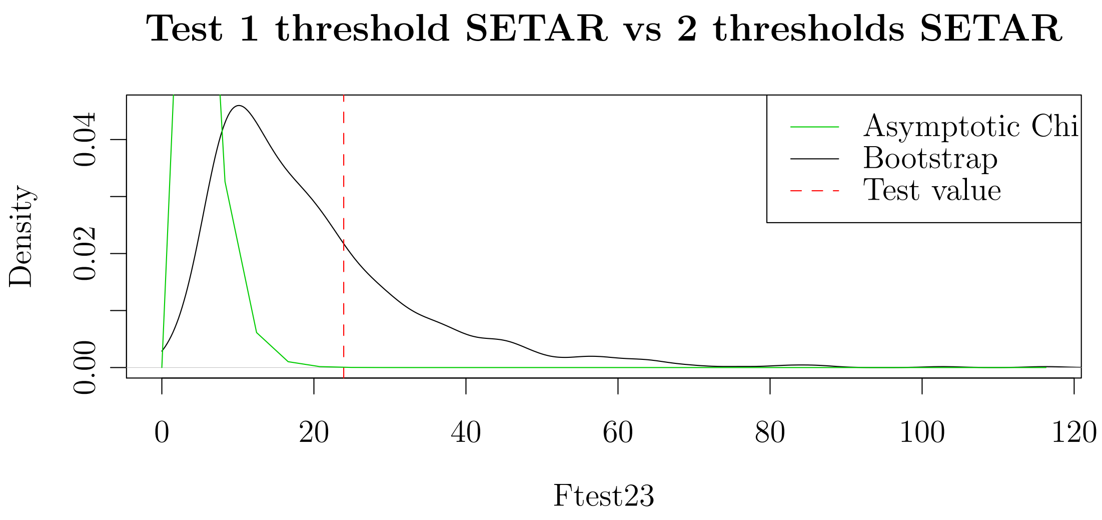
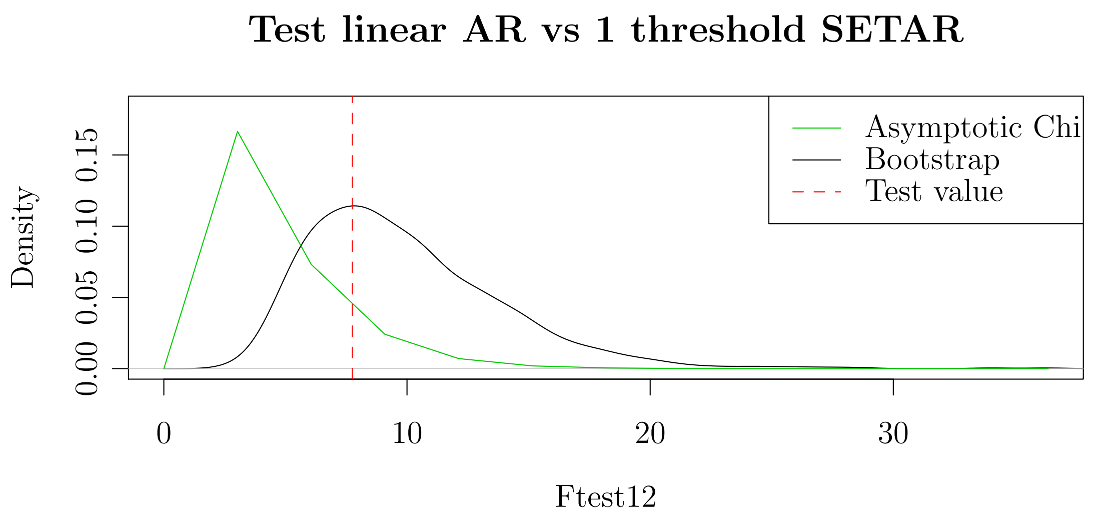
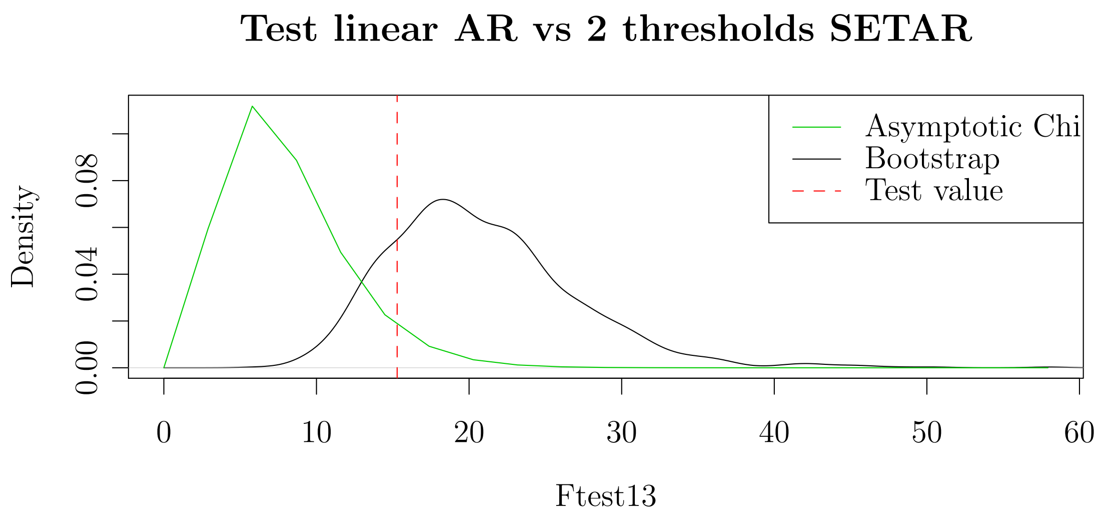


Figure 4.12. Plots of the ***SETAR*** tests of ***sTestNIB12MTN***.





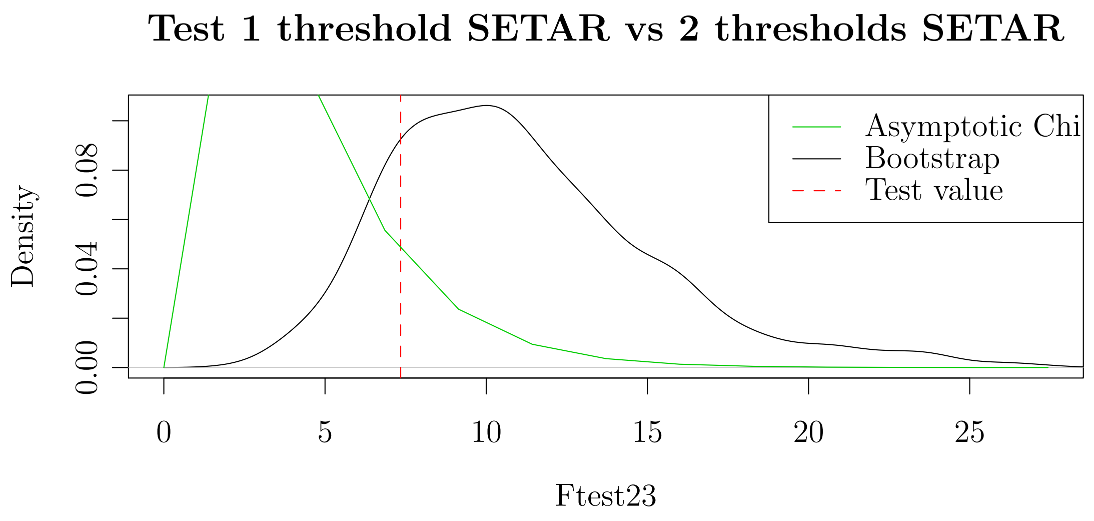


Table 4.21. The results of the ***setarTest*** of ***sNIB12MTN*** with 1000 bootstrap replications.

	F-test	90%	95%	97.5%	99%	P-value
1vs2	52.7	20.7	30.2	53.8	96.0	0.027
1vs3	80.7	64.5	84.9	108.8	173.0	0.060
2vs3	23.9	36.7	45.6	56.3	64.8	0.265

Table 4.22. The results of the ***setarTest*** of ***sTestNIB12MTN*** with 1000 bootstrap replications.

	F-test	90%	95%	97.5%	99%	P-value
1vs2	7.8	15.1	17.3	19.3	22.9	0.666
1vs3	15.3	28.8	31.2	34.6	41.6	0.815
2vs3	7.3	16.4	18.9	21.5	23.8	0.824

we search for an ARMA(p,q) model with insignificant autocorrelation in the residuals by performing a loop of Ljung-Box tests. The P-values of these Ljung-Box tests are shown in Table 4.23. We see that the autocorrelation is undoubtedly insignificant when the AR order is $p = 2$, and the MA order is $q = 8$. This is also confirmed by Figure 4.13 on the facing page where the ACF and PACF functions are plotted for the residuals of the AR(1) and ARMA(2,8) models. So we select the ARMA(2,8) as the preferred model for the data set ***sTestNIB12MTN***.

Figure 4.14 on page 58 shows the plots of the estimated ***ACF*** and ***PACF*** functions of the residuals when fitting the data ***sNIB12MTN*** to SETAR(2) and SETAR(3) models. We see that there still are some autocorrelation in the SETAR(2) model, but in the SETAR(3) model the autocorrelation is insignificant. So, we select the SETAR(3) model as the preferred model for the data set ***sNIB12MTN***. Table 4.24 on the facing page shows the estimated coefficients of this SETAR(3) model. Note that the threshold value of this SETAR(3) model for the data set ***sNIB12MTN*** is equal to the threshold value of the TVECM in Table 4.16 on page 46 for the data set ***NIB12MTN***.

4.7 Out-of-sample forecasting of the term spread

Next, we perform an out-of-sample forecasting as in Di Narzo (2008). We use the observations of ***sNIB12MTN*** and ***sTestNIB12MTN*** from May 1985 to February 2010 when estimating four selected models. We predict values of ***sNIB12MTN*** and ***sTestNIB12MTN***

Table 4.23. The P-values of the Ljung-Box tests of the ARMA(p, q) models for ***sTestNIB12MTN***. The degrees of freedom is 11 in each of these tests.

	$q = 0$	$q = 1$	$q = 2$	$q = 3$	$q = 4$	$q = 5$	$q = 6$	$q = 7$	$q = 8$
$p = 1$	0.005	0.025	0.035	0.027	0.024	0.005	0.005	0.193	0.188
$p = 2$	0.016	0.029	0.015	0.024	0.006	0.008	0.018	0.191	0.440
$p = 3$	0.063	0.016	0.120	0.033	0.027	0.048	0.025	0.163	0.342

Figure 4.13. Plots of the estimated *ACF* and *PACF* functions of the residuals when fitting the data *sTestNIB12MTN* to AR(1) and ARMA(2,8) models.

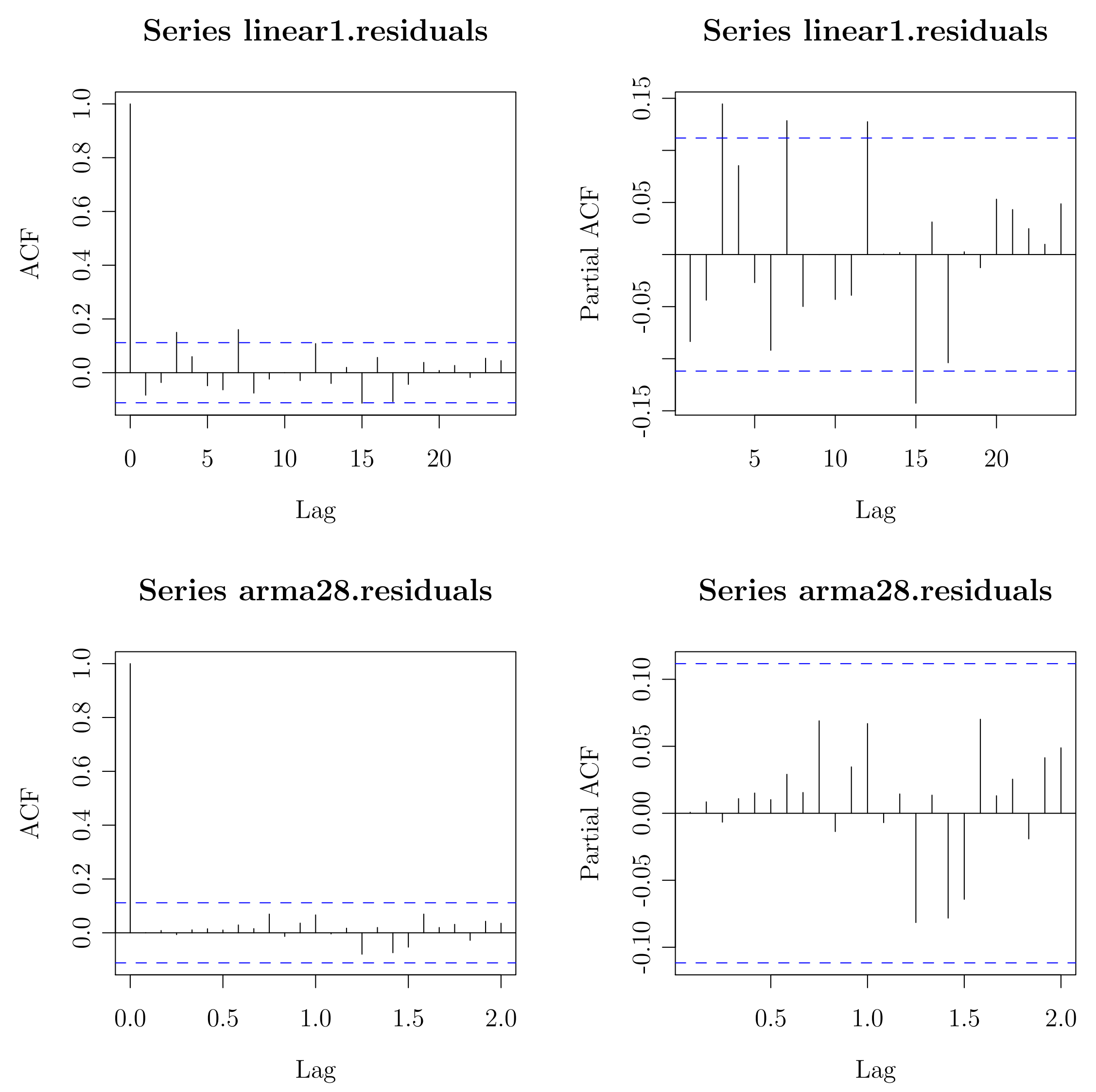
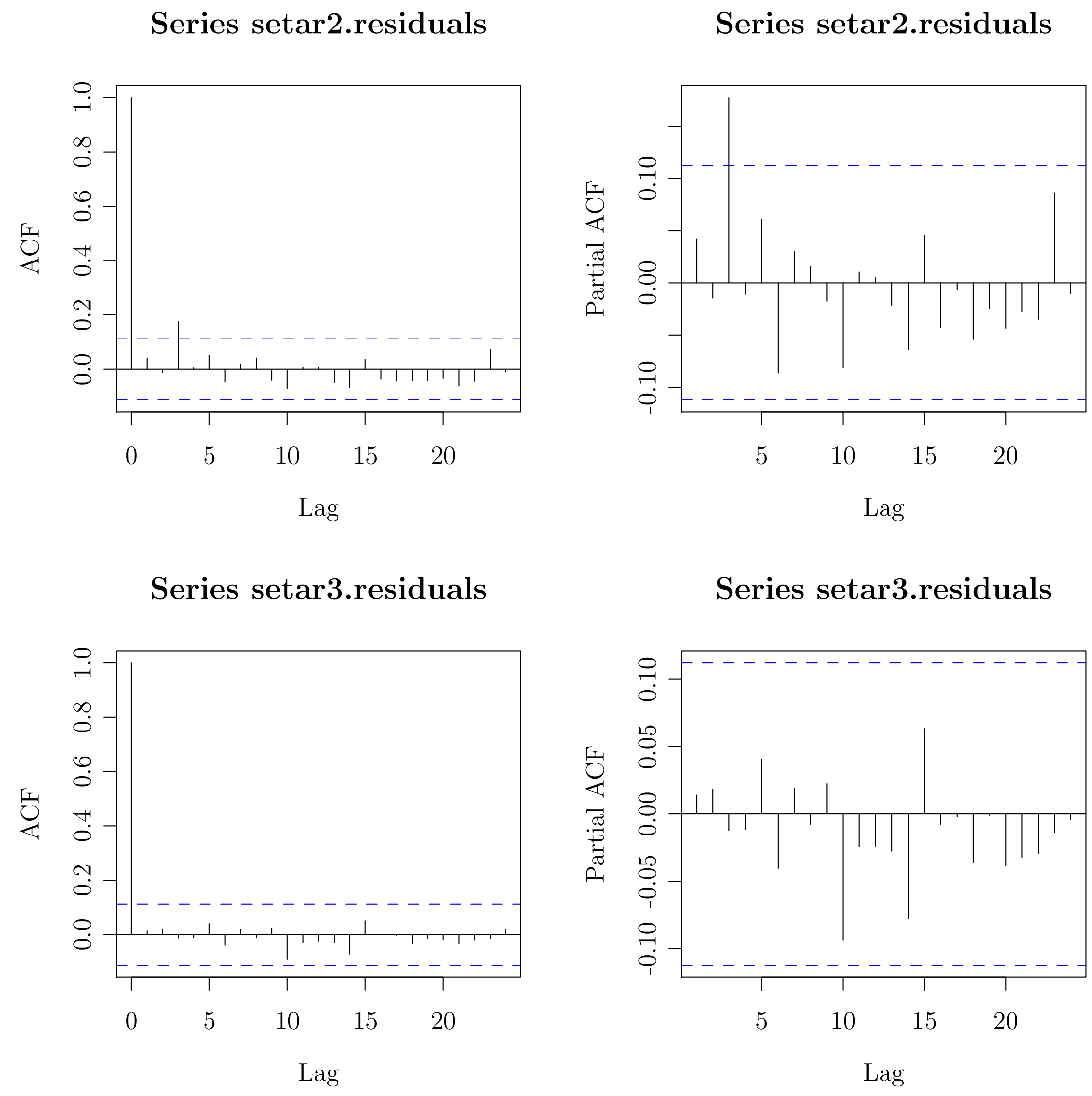


Table 4.24. The coefficients of the SETAR(3) model for the data set *sNIB12MTN*. The threshold value of this model is $\gamma = -0.83$.

	Estimate	Std. Error	t value	Pr(>|t|)
const L	-1.464	0.228	-6.431	0.000
phiL.1	0.126	0.055	2.274	0.024
phiL.2	-0.056	0.054	-1.036	0.301
phiL.3	-0.032	0.085	-0.373	0.710
const H	-0.006	0.092	-0.060	0.952
phiH.1	0.526	0.281	1.875	0.062
phiH.2	-0.104	0.249	-0.420	0.675
phiH.3	0.561	0.084	6.692	0.000

Figure 4.14. Plots of the estimated *ACF* and *PACF* functions of the residuals when fitting the data *sNIB12MTN* to SETAR(2) and SETAR(3) models.



for the last 10 months of 2010 by using these selected models, and compare these predictions with the observed values.

Figure 4.15 shows the comparison of the observed and predicted values of the cointegration relation ***sNIB12MTN*** for the last ten months of 2010. We see that the SETAR(3) model gives undoubtedly better prediction than the SETAR(1) model and the AR(1) and AR(3) models.

Figure 4.16 on the next page shows the comparison of the observed and predicted values of the cointegration relation ***sTestNIB12MTN*** (where the outliers are removed) for the last ten months of 2010 of the data set. In this case we see that the linear models AR(1) and ARMA(2,8) give approximately as good prediction as the SETAR(1) and SETAR(3) models, which is in accordance with the result of the ***setarTest*** above.

Figure 4.15. Predictions of ***sNIB12MTN*** for the last ten months of 2010.

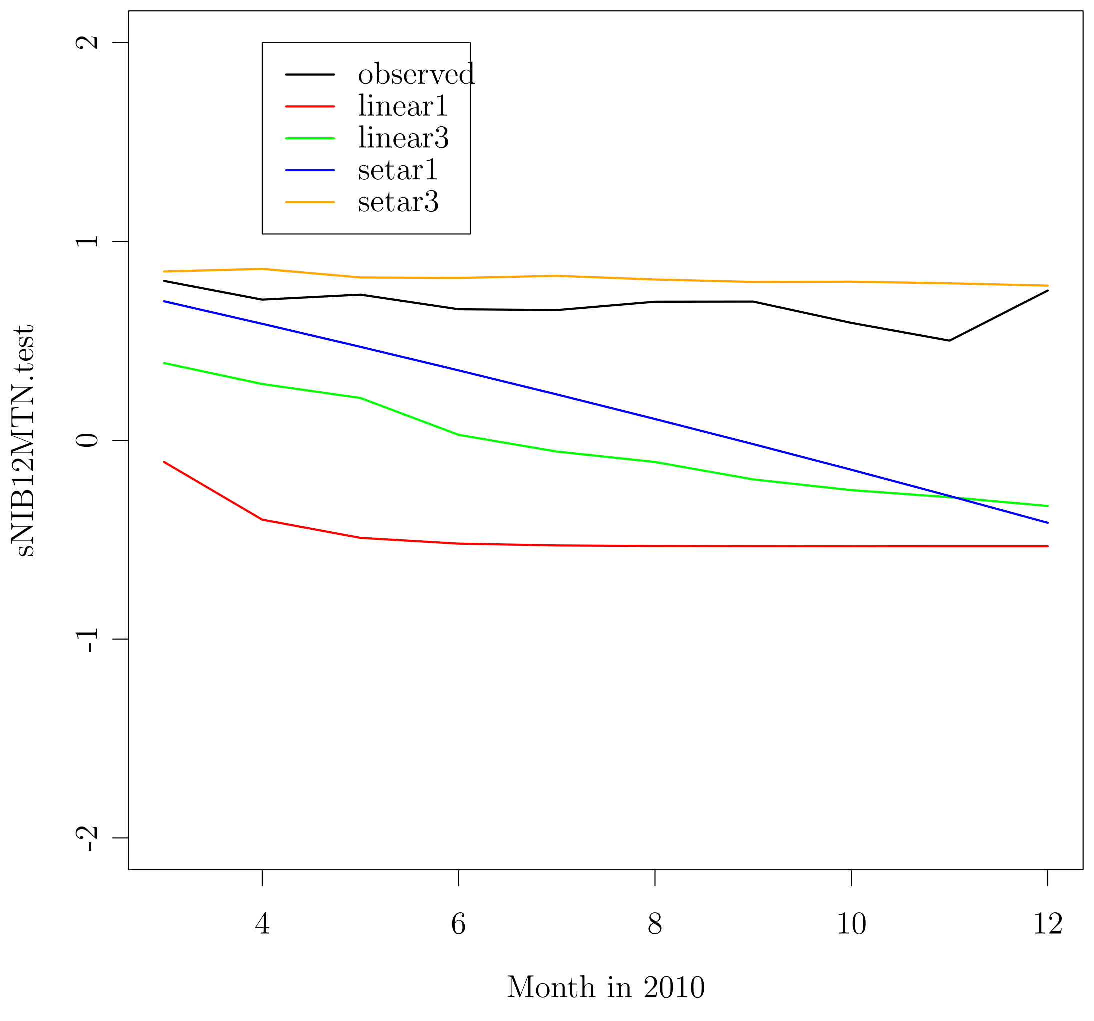
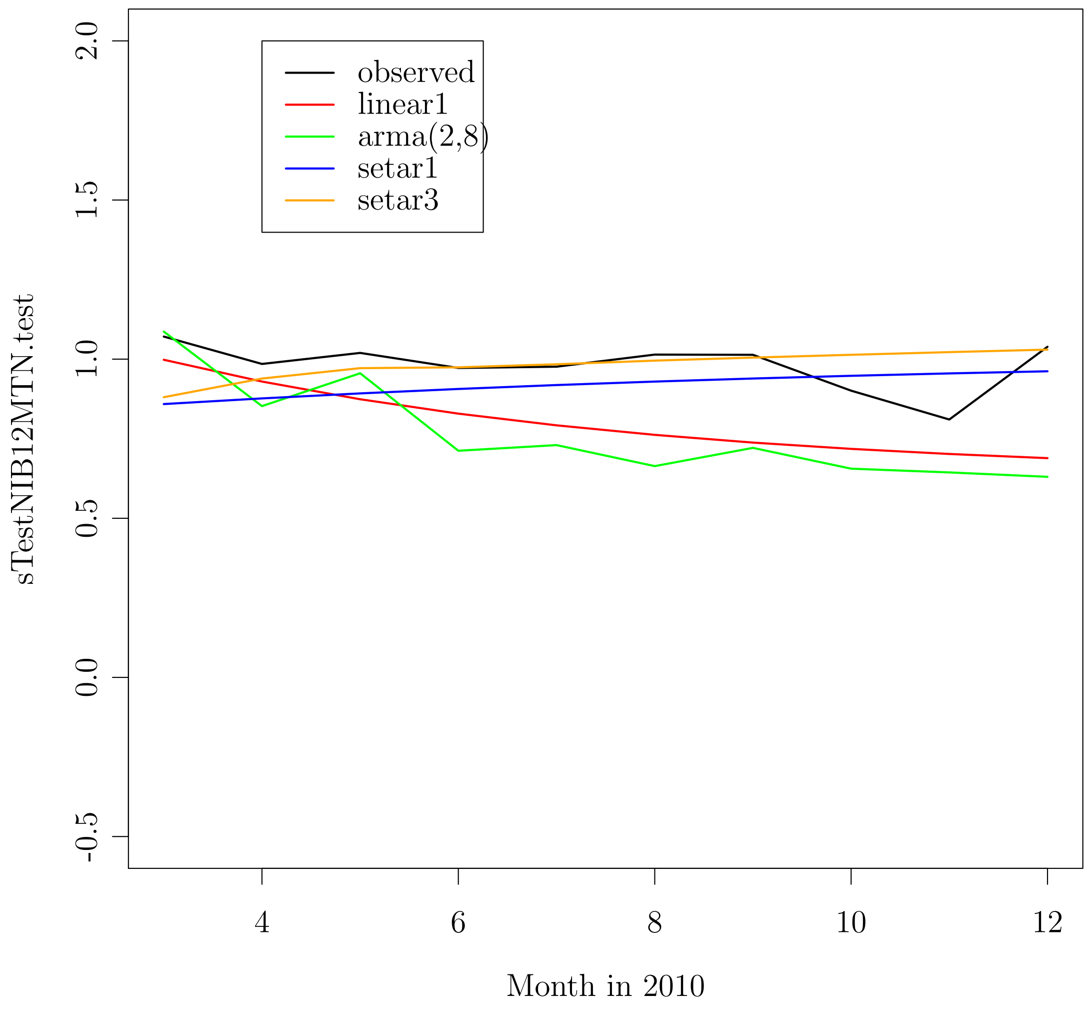


Figure 4.16. Predictions of *`sTestNIB12MTN`* for the last ten months of 2010.



4.8 Summary

4.8.1 Analysis of the NIBOR rates of the maturities tomorrow next and 12 months

In this chapter we have modeled the bivariate time series consisting of `NIB12M` and `NIBTN` by using threshold cointegration. The Dickey-Fuller and the KPSS test show that these two interest rates are $I(1)$ but without trend and drift, i.e., pure randow walks. As a starting point of the multivariate analysis, we model the bivariate time series by using a VAR(p)-model. By choosing $p = 3$, there are no significant autocorrelation in the residuals. The LM test for ARCH effects in the residuals, shows that there are no significant ARCH effects in the residuals of each of the two time series, but the multivariate test gives significant ARCH effects in the residuals. Also, the null hypothesis of normally distributed residuals are strongly rejected; there are both skewness and kurtosis. The Johansen test for cointegration rank strongly rejects the hypothesis of no cointegration relations, i.e., the cointegration rank is $r = 1$. The cointegration relation is approximately equal to the term spread, i.e., the difference between the two interest rates. The Hansen and Seo test strongly rejects the null hypothesis of linear cointegration.

The residuals of the TVECM have better properties than the residuals of the VAR(p) model we started with. The residuals in the TVECM are on the average smaller in absolute value than the residuals in the VAR(p) model as the SSR is lowered. The LM statistic for multivariate ARCH effects is considerable smaller for the residuals of the TVECM than for the residuals of the VAR(p) model. So, there are not any significant ARCH effects in the residuals of the TVECM, neither univariate nor multivariate. However, in the normality test the residuals of the TVECM do not behave better than the residuals of the VAR model. The null hypothesis of normally distributed residuals is strongly rejected, also for the residuals of the TVECM.

The term spread is modeled by a SETAR(3) model. When testing one regime against two regimes, the null hypothesis of one regime, i.e., a linear model, is strongly rejected, but when testing 2 regimes against 3 regimes, the null hypothesis of 2 regimes cannot be rejected at any reasonable significance level. Therefore, we choose a two-regime SETAR(3) model for the term spread. In the out-of-sample forecasting of the term spread, the SETAR(3) model with two regimes gives much better prediction of the term spread than the linear models.

We also illustrate how large influence the 6 outliers in `NIBTN` and the 2 outliers in the `NIB12M` have on the analysis. If these outliers are removed by using interpolation between adjacent values, the null hypothesis of linear cointegration cannot be rejected in the Hansen and Seo test.

4.8.2 Testing of Norwegian interest rates

At www.norges-bank.no there are monthly NIBOR rates of nine different maturities (TN, 1W, 2W, 1M, 2M, 3M, 6M, 9M and 12M) and monthly interest rates on government bonds

Table 4.25. The results of the setarTest with 1000 bootstrap replications on the term spread of each of the different pairs of Norwegian interest rates.

Interest rates		**P-value**			**thDelay**	**Thresholds**		**% in each regime**		
Rate 1	**Rate 2**	**1vs2**	**1vs3**	**2vs3**		**th1**	**th2**	**L**	**M**	**H**
NIB1W	NIBTN	0.130	0.688	0.989	0	-0.170	-0.090	10.8	26.4	62.8
NIB2W	NIBTN	0.004	0.009	0.093	0	-0.170	0.120	11.7	76.6	11.7
NIB1M	NIBTN	0.048	0.080	0.127	0	-0.270	-0.160	13.2	12.3	74.5
NIB2M	NIBTN	0.007	0.072	0.575	0	-0.400	0.270	12.0	76.6	11.3
NIB3M	NIBTN	0.031	0.154	0.532	0	-0.400	0.050	13.2	40.2	46.5
NIB6M	NIBTN	0.010	0.090	1.000	0	-0.360	-0.130	17.2	11.5	71.3
NIB9M	NIBTN	0.435	0.826	0.925	0	-0.780	0.190	13.3	40.3	46.5
NIB12M	NIBTN	0.015	0.088	1.000	0	-0.320	0.170	19.7	26.0	54.3
SOBL3	NIBTN	0.513	0.306	0.185	0	0.040	0.470	54.4	13.2	32.4
SOBL5	NIBTN	0.009	0.043	0.949	0	-1.360	-0.920	11.1	10.2	78.7
SOBL10	NIBTN	0.005	0.006	0.072	0	-1.330	-0.940	12.0	10.1	77.9
NIB2W	NIB1W	0.006	0.071	0.748	0	-0.040	0.020	11.3	42.0	46.7
NIB1M	NIB1W	0.129	0.476	0.692	0	-0.170	0.100	10.8	71.3	17.9
NIB2M	NIB1W	0.818	0.862	0.750	0	0.090	0.130	63.1	10.9	25.9
NIB3M	NIB1W	0.017	0.101	0.768	0	-0.350	-0.130	10.5	10.1	79.4
NIB6M	NIB1W	0.013	0.065	0.580	0	-0.440	-0.090	13.2	13.2	73.6
NIB9M	NIB1W	0.869	0.544	0.213	0	-0.170	0.270	25.2	29.6	45.1
NIB12M	NIB1W	0.007	0.000	0.025	0	-0.320	-0.050	17.9	10.1	72.0
SOBL3	NIB1W	0.081	0.187	0.581	0	-0.850	0.470	19.9	45.9	34.2
SOBL5	NIB1W	0.006	0.016	0.262	0	-0.290	0.190	48.0	10.1	41.9
SOBL10	NIB1W	0.002	0.004	0.654	0	-1.380	-0.760	10.5	18.9	70.6
NIB1M	NIB2W	0.173	0.073	0.089	0	-0.110	-0.040	19.0	23.4	57.7
NIB2M	NIB2W	0.683	0.906	0.921	0	-0.180	-0.010	15.3	28.5	56.2
NIB3M	NIB2W	0.165	0.564	0.917	0	0.020	0.070	44.5	12.0	43.4
NIB6M	NIB2W	0.158	0.477	0.882	0	-0.550	0.480	10.6	77.0	12.4
NIB9M	NIB2W	0.955	0.917	0.745	0	0.090	0.260	41.6	14.2	44.2
NIB12M	NIB2W	0.369	0.063	0.046	0	0.210	0.470	49.6	20.1	30.3
SOBL3	NIB2W	0.125	0.294	0.657	0	-1.180	0.380	12.0	51.1	36.9
SOBL5	NIB2W	0.296	0.351	0.441	0	-0.050	0.750	50.0	16.1	33.9
SOBL10	NIB2W	0.118	0.072	0.181	0	-1.050	0.180	16.8	36.5	46.7
NIB2M	NIB1M	0.275	0.645	0.888	0	0.040	0.100	52.9	27.0	20.1
NIB3M	NIB1M	0.001	0.000	0.001	0	0.070	0.100	50.0	11.1	38.9
NIB6M	NIB1M	0.009	0.033	0.407	0	-0.390	-0.090	10.5	15.5	74.0
NIB9M	NIB1M	0.724	0.791	0.763	0	-0.120	0.140	24.8	18.6	56.6
NIB12M	NIB1M	0.003	0.027	0.490	0	-0.280	-0.050	17.8	10.2	72.0
SOBL3	NIB1M	0.068	0.230	0.683	0	-1.110	-0.750	12.1	11.7	76.2
SOBL5	NIB1M	0.001	0.035	0.983	0	-1.340	-0.780	10.8	16.4	72.8
SOBL10	NIB1M	0.001	0.010	0.976	0	-0.910	1.750	25.0	52.9	22.1
NIB3M	NIB2M	0.001	0.008	0.336	0	-0.060	0.110	15.3	72.6	12.0
NIB6M	NIB2M	0.006	0.006	0.126	0	0.200	0.340	69.3	19.7	10.9
NIB9M	NIB2M	0.542	0.684	0.764	0	-0.210	0.260	21.2	36.3	42.5
NIB12M	NIB2M	0.232	0.358	0.643	0	0.150	0.480	46.4	28.8	24.8
SOBL3	NIB2M	0.029	0.154	0.703	0	-0.990	0.490	14.2	51.5	34.3
SOBL5	NIB2M	0.112	0.102	0.240	0	-1.060	1.230	16.8	61.7	21.5
SOBL10	NIB2M	0.043	0.020	0.130	0	-1.310	0.080	11.7	39.1	49.3
NIB6M	NIB3M	0.006	0.025	0.274	0	-0.190	0.150	11.8	60.5	27.7
NIB9M	NIB3M	0.486	0.640	0.759	0	-0.160	0.380	21.2	66.4	12.4
NIB12M	NIB3M	0.000	0.009	0.514	0	-0.380	-0.170	10.9	10.9	78.3
SOBL3	NIB3M	0.030	0.016	0.059	0	-0.970	-0.630	14.6	13.5	71.9
SOBL5	NIB3M	0.063	0.161	0.624	0	-1.090	1.080	16.7	61.0	22.3
SOBL10	NIB3M	0.004	0.004	0.186	0	-1.260	0.440	14.3	46.1	39.6
NIB9M	NIB6M	0.571	0.209	0.102	0	-0.040	0.020	27.4	12.4	60.2
NIB12M	NIB6M	0.134	0.266	0.617	0	-0.140	0.080	15.9	32.8	51.4
SOBL3	NIB6M	0.113	0.090	0.245	0	-0.780	-0.050	18.1	39.9	42.0
SOBL5	NIB6M	0.013	0.033	0.489	0	-1.310	0.260	11.1	52.7	36.1
SOBL10	NIB6M	0.004	0.028	0.800	0	-1.390	0.540	11.1	52.7	36.1
NIB12M	NIB9M	0.775	0.498	0.292	0	-0.060	-0.020	14.6	10.6	74.8
SOBL3	NIB9M	0.122	0.080	0.165	0	-0.930	-0.540	11.1	16.4	72.6
SOBL5	NIB9M	0.140	0.116	0.183	0	-1.120	0.280	11.5	42.9	45.6
SOBL10	NIB9M	0.198	0.129	0.211	0	-1.150	-0.030	15.0	23.0	61.9
SOBL3	NIB12M	0.231	0.210	0.319	0	-0.230	0.100	52.3	15.3	32.4
SOBL5	NIB12M	0.004	0.012	0.283	0	-1.160	0.180	10.5	53.3	36.2
SOBL10	NIB12M	0.004	0.014	0.402	0	-1.030	-0.090	21.1	30.9	48.0
SOBL5	SOBL3	0.045	0.036	0.155	0	0.040	0.190	47.0	13.5	39.5
SOBL10	SOBL3	0.008	0.027	0.507	0	-0.420	0.680	13.5	52.3	34.2
SOBL10	SOBL5	0.064	0.014	0.048	0	-0.210	0.500	11.8	62.0	26.2

Table 4.26. The results of the setarTest with **1000** bootstrap replications and optimal ***thDelay*** on the term spread of each of the different pairs of Norwegian interest rates.

Interest rates		**P-value**			**thDelay**	**Thresholds**		**% in each regime**		
Rate 1	**Rate 2**	**1vs2**	**1vs3**	**2vs3**		**th1**	**th2**	**L**	**M**	**H**
NIB1W	NIBTN	0.044	0.017	0.015	2	-0.130	-0.110	18.6	12.2	69.3
NIB2W	NIBTN	0.000	0.003		2	-0.140	0.120	16.8	70.8	12.4
NIB1M	NIBTN	0.016	0.082	0.234	1	-0.210	0.290	18.9	68.8	12.3
NIB2M	NIBTN	0.008	0.025	0.461	3	0.140	0.220	71.5	11.3	17.2
NIB3M	NIBTN	0.032	0.147	0.523	0	-0.400	0.050	13.2	40.2	46.5
NIB6M	NIBTN	0.016	0.093	0.999	0	-0.360	-0.130	17.2	11.5	71.3
NIB9M	NIBTN	0.193	0.002	0.000	2	0.100	0.220	46.0	11.5	42.5
NIB12M	NIBTN	0.017	0.044	0.376	3	0.180	0.510	46.7	26.0	27.3
SOBL3	NIBTN	0.053	0.062	0.320	2	-0.590	-0.200	29.2	15.7	55.2
SOBL5	NIBTN	0.006	0.047	0.957	0	-1.360	-0.920	11.1	10.2	78.7
SOBL10	NIBTN	0.006	0.004	0.080	0	-1.330	-0.940	12.0	10.1	77.9
NIB2W	NIB1W	0.007	0.083	0.747	0	-0.040	0.020	11.3	42.0	46.7
NIB1M	NIB1W	0.064	0.355	0.918	1	-0.160	-0.070	11.5	12.8	75.7
NIB2M	NIB1W	0.075	0.023	0.039	3	0.120	0.230	70.1	19.3	10.6
NIB3M	NIB1W	0.014	0.092	0.776	0	-0.350	-0.130	10.5	10.1	79.4
NIB6M	NIB1W	0.011	0.079	0.569	0	-0.440	-0.090	13.2	13.2	73.6
NIB9M	NIB1W	0.587	0.159	0.111	3	0.080	0.510	40.7	37.2	22.1
NIB12M	NIB1W	0.007	0.002	0.018	0	-0.320	-0.050	17.9	10.1	72.0
SOBL3	NIB1W	0.068	0.172	0.569	0	-0.850	0.470	19.9	45.9	34.2
SOBL5	NIB1W	0.003	0.010	0.267	0	-0.290	0.190	48.0	10.1	41.9
SOBL10	NIB1W	0.000	0.005	0.662	0	-1.380	-0.760	10.5	18.9	70.6
NIB1M	NIB2W	0.002	0.004	0.198	2	-0.010	0.050	54.4	28.8	16.8
NIB2M	NIB2W	0.009	0.013	0.096	3	0.080	0.180	66.1	21.9	12.0
NIB3M	NIB2W	0.021	0.006	0.035	3	0.060	0.270	52.6	33.2	14.2
NIB6M	NIB2W	0.106	0.070	0.163	1	-0.150	0.530	24.8	64.2	10.9
NIB9M	NIB2W	0.082	0.010	0.015	3	0.040	0.460	39.8	35.4	24.8
NIB12M	NIB2W	0.287	0.256	0.264	1	-0.200	0.750	23.7	66.8	9.5
SOBL3	NIB2W	0.038	0.128	0.723	2	0.520	0.780	67.2	12.8	20.1
SOBL5	NIB2W	0.023	0.019	0.124	3	-0.700	0.520	29.9	32.1	38.0
SOBL10	NIB2W	0.072	0.192	0.692	3	0.430	1.280	57.7	10.2	32.1
NIB2M	NIB1M	0.001	0.000	0.024	3	-0.100	0.140	13.1	75.5	11.3
NIB3M	NIB1M	0.000	0.000	0.008	2	-0.170	0.300	16.1	74.5	9.5
NIB6M	NIB1M	0.008	0.033	0.420	0	-0.390	-0.090	10.5	15.5	74.0
NIB9M	NIB1M	0.016	0.108	0.870	3	-0.110	0.630	27.0	62.8	10.2
NIB12M	NIB1M	0.000	0.012	0.498	0	-0.280	-0.050	17.8	10.2	72.0
SOBL3	NIB1M	0.063	0.225	0.668	0	-1.110	-0.750	12.1	11.7	76.2
SOBL5	NIB1M	0.002	0.041	0.980	0	-1.340	-0.780	10.8	16.4	72.8
SOBL10	NIB1M	0.002	0.014	0.968	0	-0.910	1.750	25.0	52.9	22.1
NIB3M	NIB2M	0.002	0.006	0.364	0	-0.060	0.110	15.3	72.6	12.0
NIB6M	NIB2M	0.015	0.005	0.113	0	0.200	0.340	69.3	19.7	10.9
NIB9M	NIB2M	0.011	0.076		3	0.350	0.500	70.4	19.9	9.7
NIB12M	NIB2M	0.084	0.047		3	0.100	0.450	42.7	29.9	27.4
SOBL3	NIB2M	0.033	0.165	0.743	0	-0.990	0.490	14.2	51.5	34.3
SOBL5	NIB2M	0.104	0.118	0.258	0	-1.060	1.230	16.8	61.7	21.5
SOBL10	NIB2M	0.032	0.000	0.005	2	-0.810	-0.140	27.4	20.1	52.6
NIB6M	NIB3M	0.004	0.017	0.304	0	-0.190	0.150	11.8	60.5	27.7
NIB9M	NIB3M	0.012	0.028	0.262	1	-0.260	0.290	15.0	56.2	28.8
NIB12M	NIB3M	0.001	0.006	0.503	0	-0.380	-0.170	10.9	10.9	78.3
SOBL3	NIB3M	0.028	0.009	0.040	0	-0.970	-0.630	14.6	13.5	71.9
SOBL5	NIB3M	0.005	0.000	0.016	1	-0.080	1.180	53.8	25.9	20.3
SOBL10	NIB3M	0.002	0.000	0.014	1	-0.250	1.630	45.1	33.1	21.8
NIB9M	NIB6M	0.032	0.127	0.809	1	-0.040	0.140	27.9	52.7	19.5
NIB12M	NIB6M	0.092	0.019	0.036	2	0.170	0.320	64.5	26.4	9.1
SOBL3	NIB6M	0.065	0.156		1	-0.140	0.570	55.5	22.4	22.1
SOBL5	NIB6M	0.011	0.038	0.490	0	-1.310	0.260	11.1	52.7	36.1
SOBL10	NIB6M	0.001	0.027	0.771	0	-1.390	0.540	11.1	52.7	36.1
NIB12M	NIB9M	0.302	0.003	0.000	1	0.030	0.100	38.9	20.8	40.3
SOBL3	NIB9M	0.051	0.115	0.437	1	-0.930	-0.510	11.5	16.8	71.7
SOBL5	NIB9M	0.125	0.125	0.222	0	-1.120	0.280	11.5	42.9	45.6
SOBL10	NIB9M	0.140	0.092	0.119	1	-1.297	-0.560	10.6	19.0	70.4
SOBL3	NIB12M	0.249	0.123	0.140	1	-0.900	-0.530	11.0	22.1	66.9
SOBL5	NIB12M	0.004	0.009	0.294	0	-1.160	0.180	10.5	53.3	36.2
SOBL10	NIB12M	0.006	0.023	0.407	0	-1.030	-0.090	21.1	30.9	48.0
SOBL5	SOBL3	0.003	0.029	0.823	3	-0.240	0.100	15.7	39.5	44.8
SOBL10	SOBL3	0.007	0.025	0.484	0	-0.420	0.680	13.5	52.3	34.2
SOBL10	SOBL5	0.067	0.018	0.026	1	-0.160	0.500	16.1	58.0	25.9

Table 4.27. The results of the *setarTest* of *sNIB12MTN* without the outliers in autumn 1992 with 1000 bootstrap replications.

	F-test	90%	95%	97.5%	99%	P-value
1vs2	58.6	16.3	21.4	28.4	48.0	0.007
1vs3	63.4	44.7	54.8	67.4	96.1	0.028
2vs3	4.1	19.9	25.1	31.7	45.4	0.979

of three maturities (3Y, 5Y, 10Y). There are also monthly interest rates on treasury bills of four maturities (3M, 6M, 9M and 12M), but these time series are considered to be too short when performing a threshold cointegration analysis as they contain less than 100 observations. Consequently, there are 66 different pairs of Norwegian interest rates (NIBOR rates and rates on government bonds) which may be tested for threshold cointegration. As the function *TVECM.XHStest* is very time-consuming, our goal has been just finding an example of a pair of Norwegian interest rates with significant threshold cointegration, and performing a thorough analysis of this pair. By estimating TVECMs for each of the 66 pairs, we checked whether the size of the narrowest regime decreased from approximately 0.1 to approximately 0.05, when the trimming parameter was lowered from 0.1 to 0.05. All such pairs were excluded as we think that the trimming parameter should not determine the optimal size of the regimes. The remaining pairs were then tested for threshold cointegration by trial and error by using the function *TVECM.XHStest*. The pair of the NIBOR TN rate and the NIBOR 12M rate was the first pair we encountered with significant threshold cointegration.

We have also performed a more systematic search among the 66 different pairs. We have seen above that a few outliers with very divergent values may change the conclusions of the statistical tests considerably. Therefore, the decision whether an outlier should be removed or not, should be considered seriously. In Juselius (2006) it is recommended that outliers are removed from financial time series by including dummy variables in the VAR model when the outliers are explained by known economical shocks. In autumn 1992 there was extensive speculation against the krone during the period of turbulence in European foreign exchange markets such that Norges Bank abandoned the fixed exchange regime of the Norwegian krone in December 1992 (Gjedrem 1999). So, the outliers in the interest rates in September, November and December 1992 should be removed. As dummy variables are not implemented in the *TVECM* function, we remove the outliers by interpolating between August and October 1992, and between October 1992 and January 1993 in all the 12 time series of Norwegian interest rates we consider. The results of the function *setarTest* run on the term spread of *NIB12M* and *NIBTN* with these outliers in autumn 1992 removed, are shown in Table 4.27. We see the results are similar to the results in Table 4.21 on page 56 where no outliers are removed. So, even if we have removed the very large outliers in autumn 1992, there is still strong evidence that the two-regime SETAR(3) is superior both to the AR(3) model and the three-regime SETAR(3) model, when modeling the term spread of the interest rates *NIB12M* and *NIBTN*. On the other hand, when removing all the 6 outliers in *NIBTN* and the 2 outliers in *NIB12M*, there is no longer any evidence for threshold effects in the term spread as Table 4.22 on page 56 shows.

First, we run the Johansen trace test by using the function *ca.jo*. For 36 of the 36 pairs of NIBOR rates, the null hypothesis of $r = 0$ cointegration relations was strongly rejected (at the 1 % level). So, all pairs of NIBOR rates have significant cointegration. On the contrary, for 3 of the 3 pairs of interest rates on government bonds, the null hypothesis of $r = 0$ cointegration relations cannot ble rejected at any reasonable significance level. Hence, there are no significant cointegration among the monthly interest rates on government bonds. Regarding the 27 pairs consisting of one NIBOR rate and the interest rate on one government bond, the null hypothesis of $r = 0$ cointegration relations is rejected at the 1, 5 and 10 % level in 1, 10, and 5 cases, respectively, while there are no significant cointegration among 11 of these 27 pairs of interest rates.

We also run the ADF test on the term spread, i.e., the difference between the interest rates in each pair. In 64 of the 66 pairs, the null hypothesis that the term spread follows a unit root process without drift and trend, is rejected at the 1 % level.

Next, we run the ***setarTest*** on the term spread of each of the 66 pairs of Norwegian interest rates, using ***nboot=1000*** bootstrap replications to get good estimates of the P-values, and ***m=4*** lagged terms of s_t (i.e., the terms s_{t-1}, s_{t-2}, s_{t-3} and s_{t-4}) in each regime. In TVECMs the threshold variable which governs the regime selection at time t, is fixed to s_{t-1}, while in SETAR models we may choose the threshold variable among the lagged terms present in the model by using the parameter ***thDelay*** (threshold delay). If ***thDelay=d***, the threshold variable is selected as s_{t-1-d}, so the possible values of ***thDelay*** is $0, 1, \ldots, (m-1)$ where 0 is the default value. We find the optimal value of ***thDelay*** by estimating a SETAR model with three regimes using the function ***setar*** with ***thDelay=0:3***, which means that the ***setar*** function searches for the optimal value of the threshold delay among the possible values $0, 1, 2, 3$. Table 4.25 on page 62 and Table 4.26 on page 63 show the results of the ***setarTest*** when the parameter threshold delay has its default value 0 and its optimal value, respectively. We see that the P-value when testing one regime against two regimes, is less than 0.05 for 33 and 45 of the 66 interest rate pairs in Table 4.25 and Table 4.26, respectively. So, there is sgnificant threshold effects in the term spread of many interest rate pairs. Hence, we would expect significant threshold cointegration in many interest rate pairs. In Table 4.28 we illustrate the effect of choosing the optimal threshold delay rather than its default value. We see that the number of cases with significant threshold effects increases considerably by using the optimal threshold delay. This means that threshold delay should be implemented both in the ***TVECM*** function and the ***TVECM.XHStest*** function.

Table 4.28. The number of interest rate pairs in Table 4.25 on page 62 and Table 4.26 on page 63 with P-value < 0.05 in the ***setarTest***.

Number of regimes in		*thDelay*	
H_0	H_1	**0**	**optimal**
1	2	33	45
1	3	25	41
2	3	4	15

According to Table 4.28 a three-regime SETAR model should be selected for the term spread for 15 interest rate pairs, i.e., the two-regime model is rejected at the 5 % level in these cases. However, most of these models have a very narrow middle regime as shown in Table 4.26 on page 63, contrary to the ideas from economic theory of a wide middle regime where the time series is stable. But, the pair consisting of `NIB3M` and `NIB1M` is very interesting. When `thDelay=2`, the division into regimes seems reasonable, and the two-regime model is rejected at the 1 % level. Hence, this pair should be tested for threshold cointegration, but we are not able to do it before the functions `TVECM` and `TVECM.XHStest` are extended with the threshold delay functionality. However, we need an example of a three-regime TVECM to illustrate the algorithm developed in Chapter 3. So, in Chapter 5 we simulate a bivariate time series generated by such a model.

Chapter 5

Analysis of a simulated TVECM with three regimes

In this chapter we simulate a bivariate time series from a TVECM with two thresholds by using the function `TVECM.sim` in the package **tsDyn**. Thereafter, we analyse this time series by using the same tools as in Chapter 4. We see that the parameters and the thresholds are estimated quite accurately, and that both a two-regime model and a linear model are strongly rejected when tested against a three-regime model.

5.1 Simulation

A bivariate TVECM with a constant term, an error correction term and two lagged differences for each of the variables, contains 12 parameters in each regime, i.e., 36 parameters when the TVECM has two thresholds. In order of being able to estimate so many parameters, we choose the number of observations in the simulated time series fairly large, $N = 2\,000$. Due to the fact that the parameters have to satisfy a stability condition, we cannot choose these 36 parameters arbitrarily. (The stability condition for a TVECM is similar to the stability condition (2.9) for a VAR(p) model.) For simplicity, we use the parameters of the first TVECM we estimated by using the time series NIB12M and NIBTN. That is, we let the parameters in the lower and upper regime of the simulated TVECM be equal to the parameters in the lower regime of the estimated TVECM, and we let the parameters in the middle regime of the simulated model be equal to the parameters in the upper regime of the estimated model. When estimating this TVECM in Chapter 4, we discovered that the `TVECM` function did not find the TVECM with the smallest SSR without specifying a search interval for the threshold. As the TVECM shown in Table 4.16 on page 46 was estimated with such a search interval specified by `th1=list(int=c(-1.5,1.5))`, these parameters are a bit different from those used in the simulation, which are shown in Table 5.1 on the following page. As a consequence, also the cointegration value changed when introducing the search interval for the threshold. Therefore, we also keep the cointegration value $\beta = 1.046$ from this first estimation in our simulation. The threshold values γ_1 and γ_2 are tuned such that approximately 12% of the observations are in the lower and the upper regime. As a result, $\gamma_1 = -3.7$ and

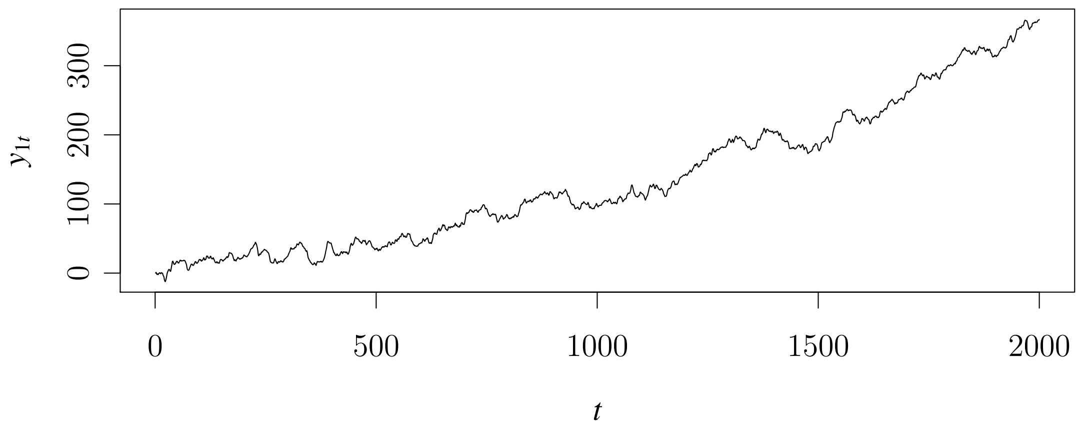
Table 5.1. The coefficients of the simulated TVECM. The thresholds and the cointegration value used in the simulation is $\gamma_1 = -3.7$, $\gamma_2 = 1.2$ and $\beta = 1.046$.

Regime	Term	Equation NIB12M	Equation NIBTN
Lower	ECT	0.0999	0.9871
	Const	0.0948	1.3375
	NIB12M t -1	0.4434	1.4526
	NIBTN t -1	0.0123	0.0056
	NIB12M t -2	-0.0426	-0.2651
	NIBTN t -2	0.0333	0.0539
Middle	ECT	-0.0424	0.0069
	Const	0.0142	0.0237
	NIB12M t -1	0.5554	0.8862
	NIBTN t -1	0.1809	-0.2722
	NIB12M t -2	-0.1230	0.2262
	NIBTN t -2	-0.0789	-0.6334
Upper	ECT	0.0999	0.9871
	Const	0.0948	1.3375
	NIB12M t -1	0.4434	1.4526
	NIBTN t -1	0.0123	0.0056
	NIB12M t -2	-0.0426	-0.2651
	NIBTN t -2	0.0333	0.0539

$\gamma_2 = 1.2$. The error terms are drawn from a bivariate normal distribution with covariance matrix equal to the identity matrix, and the starting values are chosen as $y_0 = \begin{bmatrix} 0 & 0.5 \end{bmatrix}'$ and $y_1 = \begin{bmatrix} 1 & 1.5 \end{bmatrix}'$. The simulated time series $y_t = \begin{bmatrix} y_{1t} & y_{2t} \end{bmatrix}'$ is plotted in Figure 5.1 on the next page. Both y_{1t} and y_{2t} seem to be nonstationary without a constant mean. However, the cointegration relation $w_t = y_{1t} - \beta y_{2t}$, which is plotted in Figure 5.2 on page 70 seems to be stationary. We see that the majority of the observations are in the middle regime between the lower threshold (red line) and the upper threshold (green line). Further, this time series returns rapidly from the outer regimes: if an observation is in the lower or upper regime, then the next observation is not in this regime. Sometimes, it jumps directly from the lower to the upper regime or vice versa, but normally it returns back to be middle regime and stays there for a while. Thus, this time series has a clear mean reverting property.

The differenced series Δy_{1t}, Δy_{2t} and Δw_t which are plotted aginst w_{t-1} in Figure 5.3 on page 71, seem to be stationary. These plots are much more symmetric than the corresponding plots in Figure 4.6 on page 45 of ΔR_t, Δr_t and Δs_t. While the plots in Figure 4.6 on page 45 contain points where s_{t-1} is large negative and lack points where $s_{t-1} > 2$, the plots in Figure 4.6 on page 45 seem to be approximately symmetric around $w_{t-1} = -1.3$. Hence, it is reasonable that a good model for the time series y_t has three regimes instead of two regimes, i.e., the observations with large postive values of w_t are moved to a new upper regime.

Figure 5.1. Plot of the simulated time series $y_t = [y_{1t} \;\; y_{2t}]'$.



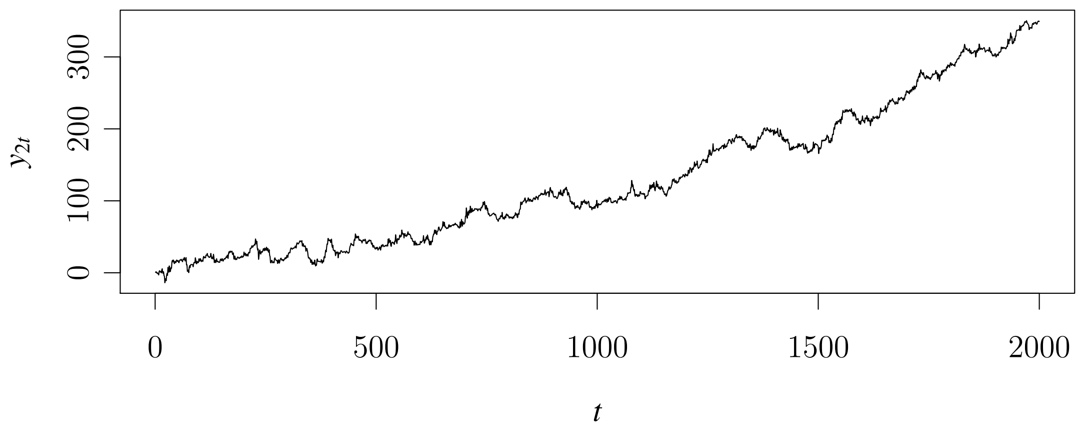
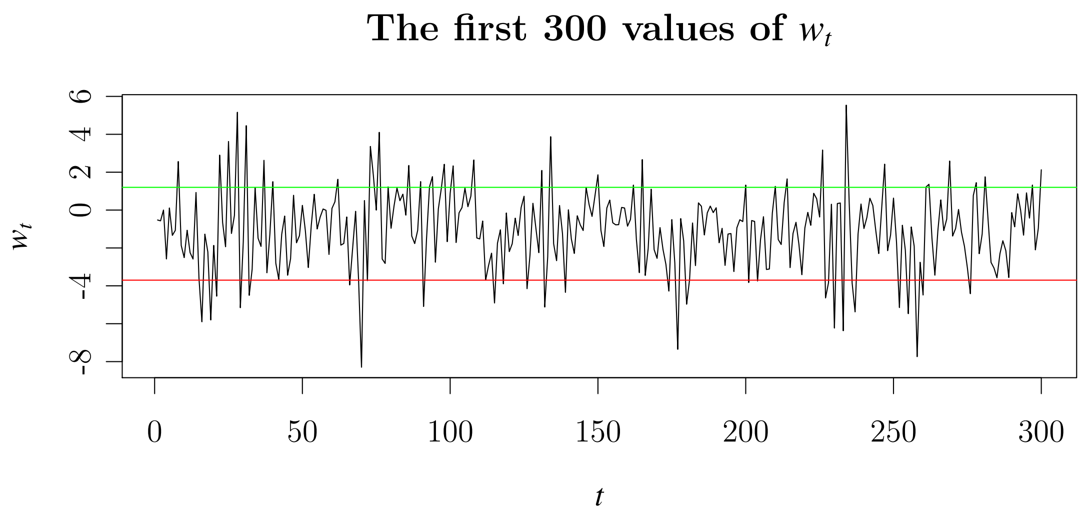


Figure 5.2. Plot of the cointegration relation $w_t = y_{1t} - \beta y_{2t}$ of the simulated series y_t. The horisontal red line in each plot denotes the threshold $\gamma_1 = -3.7$, and the horisontal green line in each plot denotes the threshold $\gamma_2 = 1.2$.



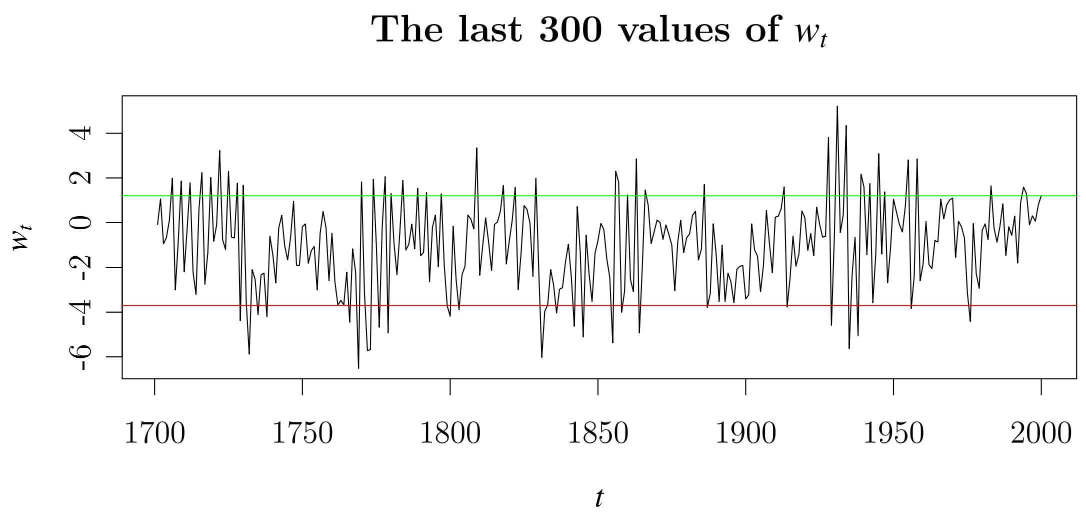
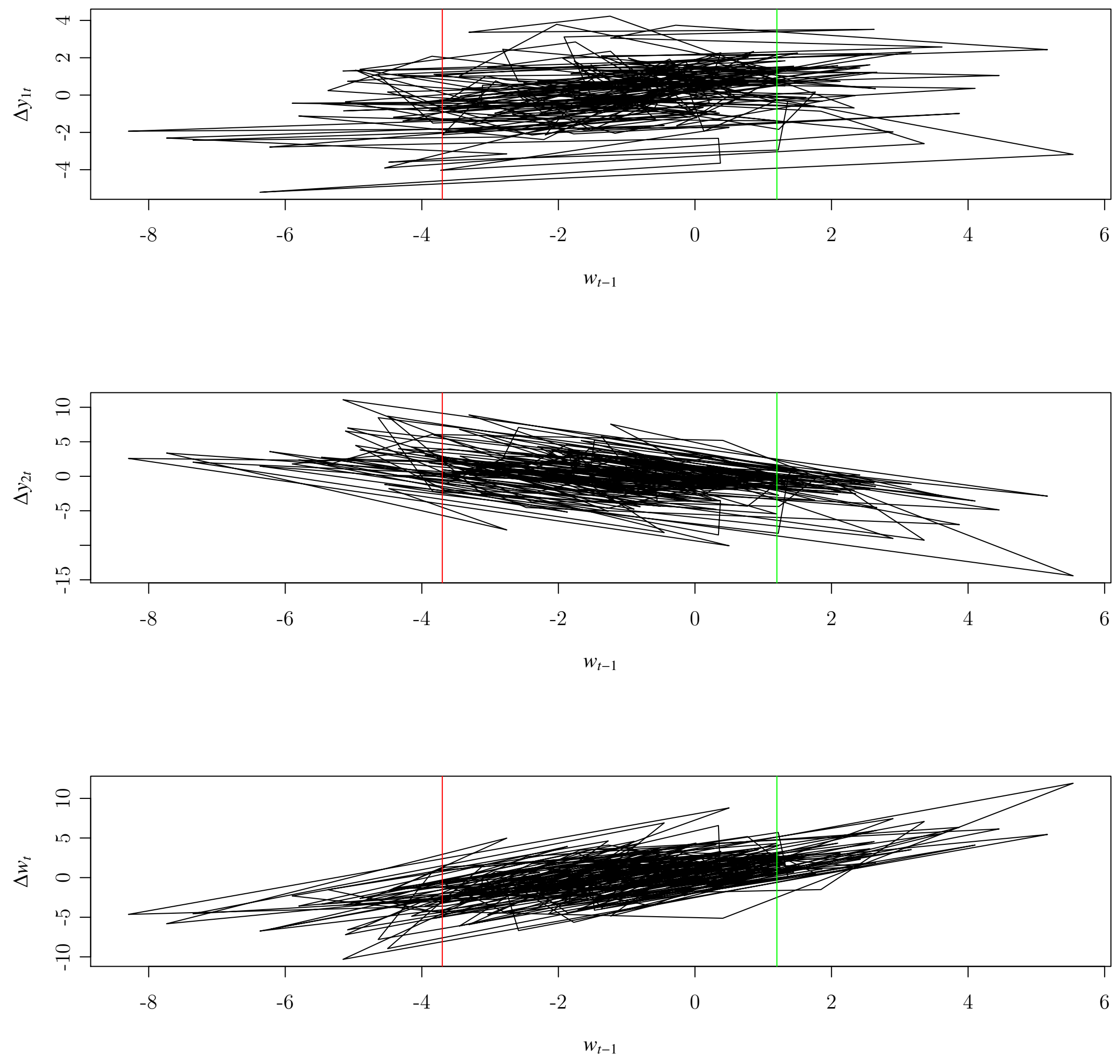


Figure 5.3. Plots of Δy_{1t}, Δy_{2t} and Δw_t against w_{t-1} for $t = 1, \ldots, 300$. The vertical red line in each plot denotes the threshold $\gamma_1 = -3.7$, and the vertical green line in each plot denotes the threshold $\gamma_2 = 1.2$.



5.2 Estimation of a TVECM

We estimate a TVECM with two thresholds from the simulated time series by using the function *TVECM* in the package **tsDyn**. We let the search interval for γ_1 and γ_2 in the grid search be $[-5, 3]$. In Figure 5.4 we see that the SSR as a function of γ has minima when $\gamma_1 = -3.7$ and $\gamma_2 = 1.2$ so this plot detects the original threshold values used when simulating the time series y_t. The result of the grid search for the threshold values is $\gamma_1 = -3.704$, $\gamma_2 = 1.196$], i.e., close to, but not exactly equal to the original threshold values -3.7 and 1.2. However, if we estimate the TVECM once again by using the function *TVECM* with fixed threshold values $\gamma_1 = -3.7$ and $\gamma_2 = 1.2$, we get exactly the same SSR as before (4006.463 in the first model and 4006.463 in the second model). Consequently, the minor differences in the threshold values, do not influence on the partion of the observations into the three regimes, i.e., the two estimated models are exactly equal.

Figure 5.4. Plot of the grid search for the threshold γ in *TVECM(tvecm1.data,nthresh=2,lag=2,...)*.

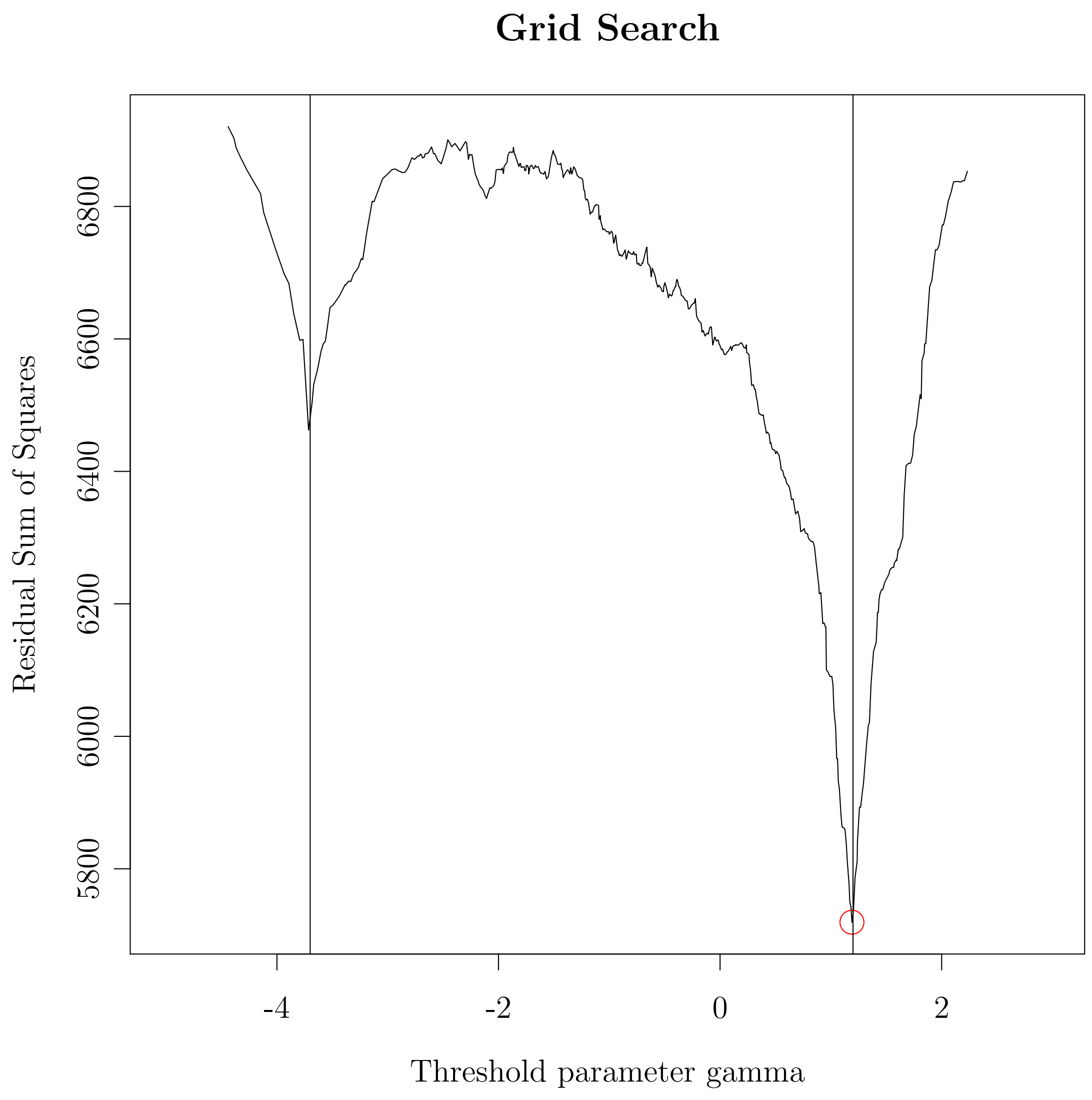


Table 5.2 shows the coefficients of the estimated TVECM. We see that there are only minor differences between these coefficients and those used when simulating the time series y_t.

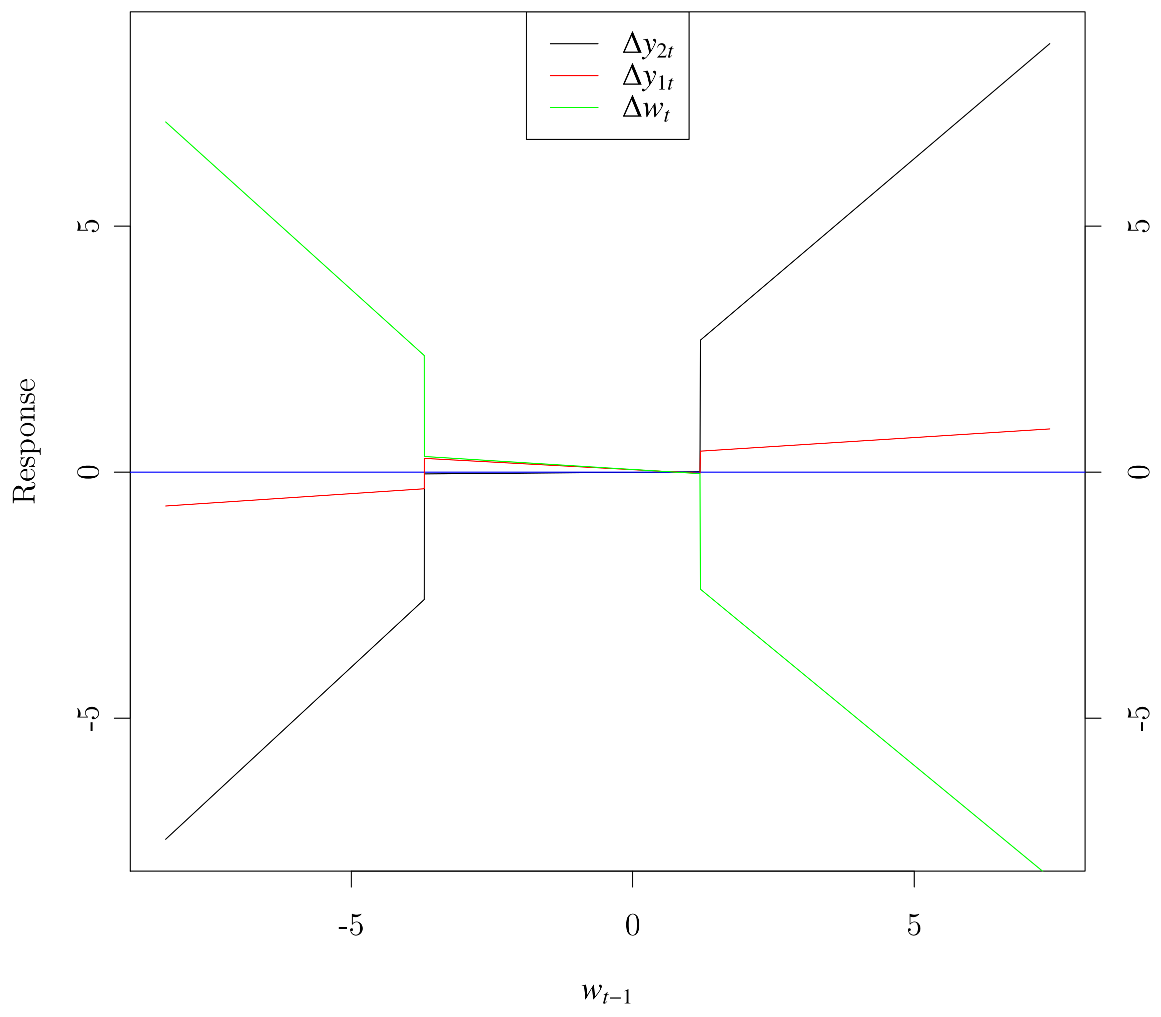
Given the short-run dynamics, i.e., the lagged terms of Δy_{1t} and Δy_{2t}, the time series Δy_{1t} and Δy_{2t} and Δw_t are linear functions of w_{t-1} in each regime. Figure 5.5 on the next page shows the graph of these response functions. We see that when w_{t-1} is large negative, Δy_{2t} is large negative and Δy_{1t} is small negative, such that $w_t = w_{t-1} + \Delta w_t = w_{t-1} + \Delta y_{1t} - \beta \Delta y_{2t}$ is much larger than w_{t-1}. Thus, if we are in the lower regime, which is an unstable area, the error correction term causes a large step towards the middle regime. However, in the middle regime, the coefficients of the constant term and the error correction term are not significant, while most of the coefficients of the lagged terms of $\Delta y_{1t}t$ and Δy_{2t} are significant, see Table 5.2. So, in the middle regime, the bivariate time series Δy_t is approximately a stable VAR(2) process. Finally, when w_{t-1} is large positive, Δy_{2t} is large positive and Δy_{1t} is small positive, such that $w_t = w_{t-1} + \Delta w_t = w_{t-1} + \Delta y_{1t} - \beta \Delta y_{2t}$ is much smaller than w_{t-1}. Thus, if we are in the upper regime, which is an unstable area, the error correction term causes a large step towards the middle regime.

Next, we run the Hansen and Seo test (the function ***TVECM.XHSTest***) for the simulated time series y_t. Table 5.3 on page 75 shows that the results of the Hansen and Seo test for the simulated time series y_t. We see that the null hypothesis of linear cointegration is strongly rejected irrespective of one or two thresholds in the alternative hypothesis, and irrespective of the bootstrap type. The estimated P-values from the bootstrapping

Table 5.2. The coefficients of the estimated TVECM from the simulated data with $\beta = 1.046$, `lag=2` and threshold values $\gamma_1 = -3.7$ and $\gamma_2 = 1.2$. The standard errors of the coefficients are in the parenthesis. The symbols '***', '**', '*' and '.' denote significance at the 0.1 %, 1 %, 5 % and 10 % level, respectively.

Regime	Term	Equation Var1	Equation Var2
Lower	ECT	0.0763(0.4195)	1.0608(2.5e-28)***
	Const	-0.0558(0.8809)	1.3397(0.0003)***
	Var1 t -1	0.4669(4.4e-07)***	1.2941(1.2e-42)***
	Var2 t -1	0.0312(0.5650)	0.0631(0.2454)
	Var1 t -2	-0.0601(0.3793)	-0.2774(5.2e-05)***
	Var2 t -2	0.0576(0.1225)	0.0922(0.0137)*
Middle	ECT	-0.0621(0.0345)*	0.0088(0.7652)
	Const	0.0495(0.2348)	-0.0041(0.9210)
	Var1 t -1	0.5689(2.2e-56)***	0.9213(5.5e-132)***
	Var2 t -1	0.1780(5.9e-19)***	-0.2926(8.4e-47)***
	Var1 t -2	-0.1155(3.9e-05)***	0.2585(8.1e-20)***
	Var2 t -2	-0.0796(2.1e-08)***	-0.6559(1.2e-317)***
Upper	ECT	0.0727(0.4122)	0.9710(4.7e-27)***
	Const	0.3385(0.0418)*	1.5142(2.2e-19)***
	Var1 t -1	0.3620(3.5e-05)***	1.5182(5.3e-63)***
	Var2 t -1	0.0570(0.2663)	0.0164(0.7493)
	Var1 t -2	-0.0905(0.1029)	-0.3188(1.1e-08)***
	Var2 t -2	0.0866(0.0141)*	0.0670(0.0577).

Figure 5.5. The responses of Δy_{1t}, Δy_{2t} and Δw_t to w_{t-1} in the estimated TVECM for the simulated time series y_t.



are exactly equal to 0 in all cases because no one of the bootstrap replications gives a test statistic as large as the supLM value for the simulated time series y_t. Consequently, the Hansen and Seo test detects threshold cointegration for this simulated time series, which is as it should be. In the last two lines of the table, we have run ***TVECM.XHStest*** with the parameter ***ngridTh*** equal to the length $N = 2000$ of the simulated time series such that all the w_t values are included in the set of possible threshold values in the grid search. We see that the set of all possible threshold values contains 1798 elements, and that the supLM statistic is 264.2 and 533.8 when the number of thresholds is 1 and 2 respectively, which is a bit larger than the results (262.2 and 529, respectively) when ***ngridTh*** has the value 200, i.e., when only $\frac{20000}{1798} = 11.1$ % of the possible threshold values are included in the grid search. Nor the percentages of the observations in each regime have changed noticeably. So, for this simulated time series, there are only minor differences between the results when running the grid search with all possible threshold values included and the results when running the grid search with only a fraction of the possible threshold values included. However, keep in mind that when estimating TVECMs from real data, the function LM(γ_1, γ_2) may be highly irregular, such that it is advisable to run ***TVECM.XHStest*** with the parameter ***ngridTh*** equal to the length of the time series, and, if necessary due to time consumption, a small value on the parameter ***nboot***.

5.3 Testing a two-regime threshold model against a three-regime threshold model

In Section 5.2 on page 72 linear cointegration was strongly rejected when tested against threshold cointegration, but we do not yet know whether the TVECM should have two or three regimes. As in Section 4.6 on page 53, we solve this problem by running the ***setarTest*** on the cointegration relation $w_t = y_{1t} - \beta y_{2t}$ (where $\beta = 1.046$ is fixed). The results in Table 5.4 on the following page show that a linear model (one regime) is strongly rejected when testing against two and three regimes, which is consistent with the results of the Hansen and Seo test in Table 5.3. In addition, a two-regime SETAR model is strongly rejected when testing against a three-regime SETAR model. (All the estimated P-values are exactly 0 because no one of the bootstrap replications gives as large value of the test statistic as the value of the test statistic for the time series w_t.) Consequently, the SETAR model for the time series w_t should have three regimes, and the TVECM for the bivariate time series y_t should also have three regimes. Also, the plots of the ***setarTest***

Table 5.3. The results of the Hansen and Seo test for the simulated time series y_t.

nthresh	ngridTh	boot type	nboot	supLM	P-value	Seconds	L %	M %	U %
1	200	FixedReg	1000	262.2	0.000	545.2	86.8		13.2
2	200	FixedReg	1000	529.0	0.000	80690.1	10.0	76.9	13.2
1	200	ResBoot	1000	262.2	0.000	1050.5	86.8		13.2
2	200	ResBoot	200	529.0	0.000	16623.2	10.0	76.9	13.2
1	1798	FixedReg	2	264.2	0.000	18.8	87.0		13.0
2	1798	FixedReg	2	533.8	0.000	24580.5	10.1	76.9	13.0

of w_t in Figure 5.6 on the facing page confirm these results. We see that in each of the three plots the value of the test statistic is far to the right in the tail of the estimated distribution. At last, we note that the threshold values $\gamma_1 = -3.704$ and $\gamma_2 = 1.189$ which we got from the ***setarTest***, are very close to the oringal threshold values $\gamma_1 = -3.7$ and $\gamma_2 = 1.2$.

5.4 Summary

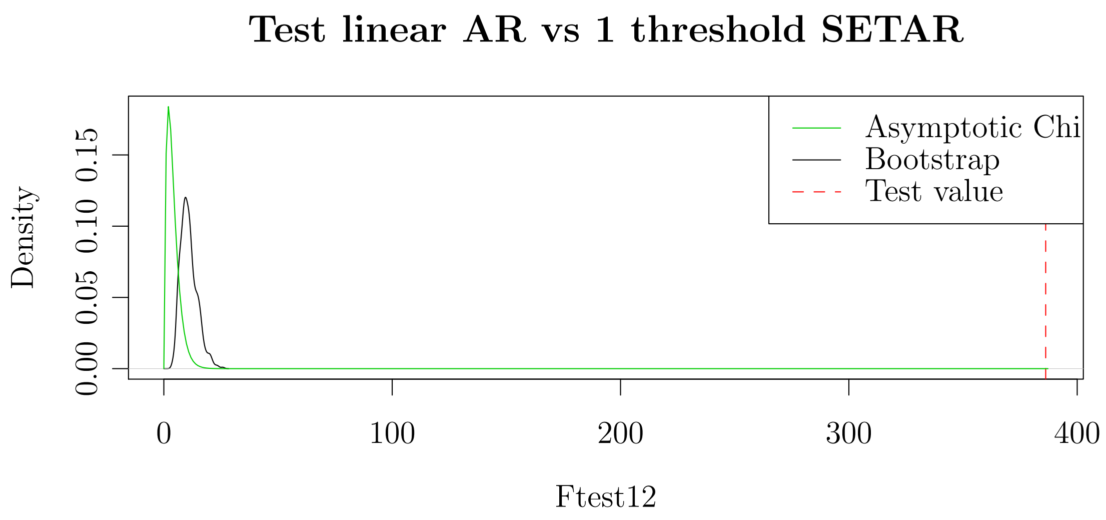
In this chapter we have simulated a TVECM with three regimes, and thereafter estimated a three-regime TVECM from the simulated time series. When specifying large enough search intervals for the threshold values, the ***TVECM*** function finds the thresholds which were used in the simulation quite accurately, and the plot of the grid search for the thresholds shows clear minima for these threshold values. As expected, the coefficients of the estimated TVECM are close to the coefficients of the three-regime TVECM which was used in the simulation. When running the Hansen and Seo test on this simulated time series, the null hypothesis of linear cointegration is strongly rejected. The function ***TVECM.XHStest*** finds that both the minimum value of SSR and the maximum value of the LM statistic occur for the thresholds −3.71077 and 1.18421, which are quite close to the thresholds −3.7 and 1.2 which were used in the simulation.

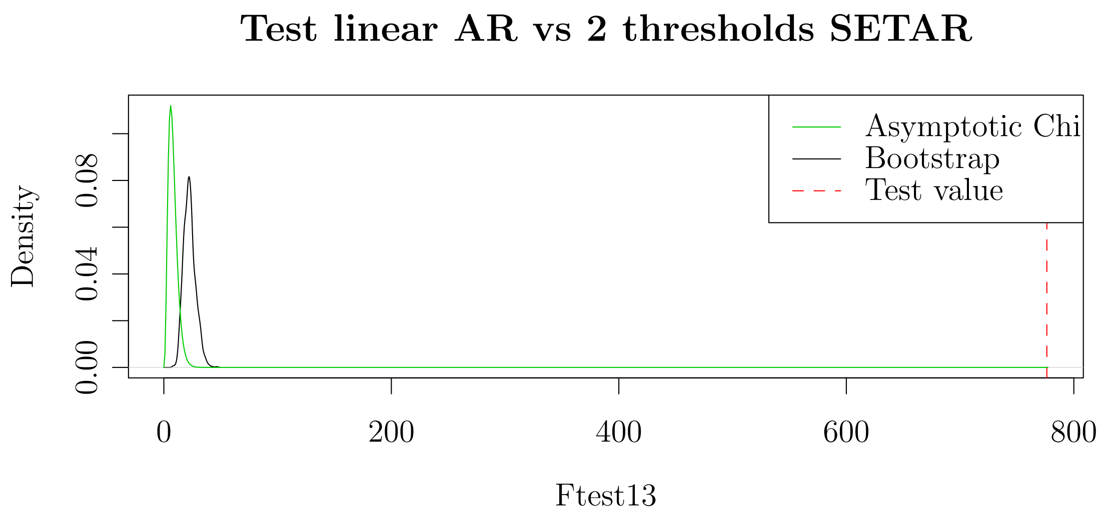
The cointegration relation, which is approximately equal to the difference between the two time series, is modeled by a SETAR process. When testing one regime against two regimes, the null hypothesis of one regime is strongly rejected. When testing two regimes against three regimes, the null hypothesis of two regimes is strongly rejected. So, a three-regime threshold model should be used for this data set, which is in accordance with the fact that the data was generated from a three-regime TVECM.

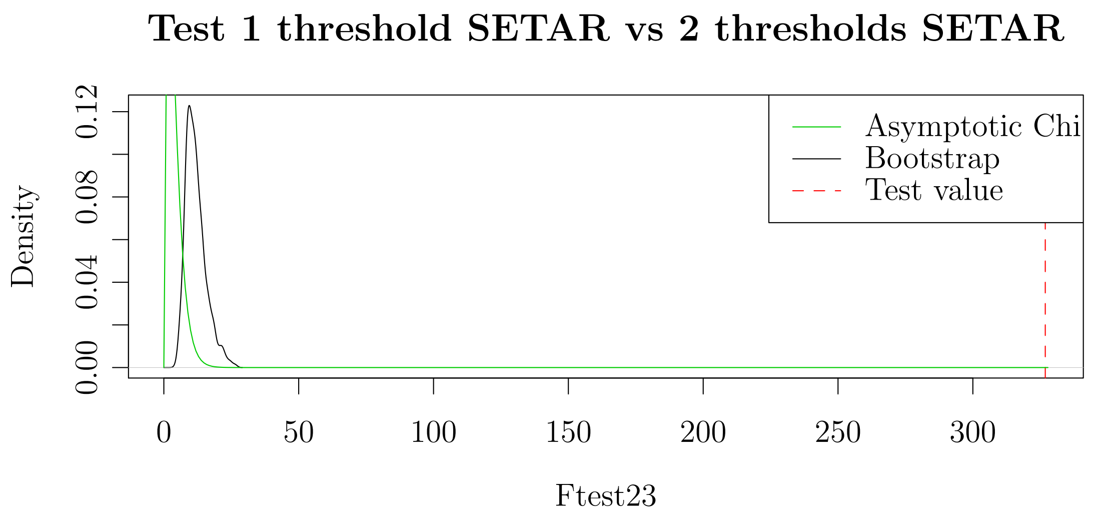
Table 5.4. The results of the ***setarTest*** of ***ssim*** with 1000 bootstrap replications.

	F-test	90%	95%	97.5%	99%	P-value
1vs2	386.2	15.9	17.5	19.9	21.2	0.000
1vs3	776.3	29.7	32.0	33.6	36.5	0.000
2vs3	326.9	16.9	18.9	21.5	23.3	0.000

Figure 5.6. Plots of the *SETAR* tests of w_t.







Chapter 6

Conclusion and discussion

In this thesis we have extended the Hansen and Seo test implemented in the R package **tsDyn** to the case of three regimes in the alternative hypothesis. We have generalized the two existing versions of the LM statistic used in the Hansen and Seo test to the case of three regimes, and have shown that they are equal under certain conditions. The grid search algorithm, which is necessary when maximizing the LM statistic, is also extended to the case of three regimes, and it is rewritten such that if the cointegration value β is given, it really maximizes the LM statistic under the constraints specified by the user. As the time consumption of this algorithm increases as a quadratic function of the number of possible threshold values, our version of the Hansen and Seo test is very time-consuming when the number of bootstrapping replications is e.g., 1000, which is often recommended when estimating P-values by using bootstrapping. On the other hand, it is not at all satisfactory with an algorithm which does not maximize the LM statistic correctly. Fortunately, in the case of modeling long bivariate time series with only a few digits in each observation, the time consumption is considerably reduced if we use the cointegration value $\beta = 1$ instead of the estimated value of β. The reason is that the number of different values of the threshold variable, which in this case is the lagged difference between the two time series, is small compared to the total number of observations in the time series.

Our analysis of interest rates provides strong evidence that the monthly NIBOR rates of the maturities tomorrow next and 12 months are cointegrated $I(1)$ processes, and that a two-regime TVECM is superior to a linear VECM when modeling this bivariate time series. If two lagged differences are included in the TVECM, the autocorrelation in the residuals is removed, and there are no ARCH effects in the residuals of the TVECM, neither univariate nor multivariate. However, there are both skewness and kurtosis in the residuals, probably due to the existence of some outliers. In addition, we find strong evidence that a two-regime SETAR(3) model is superior to both an AR(3) model and a three-regime SETAR(3) model, when modeling the cointegration relation. In the out-of-sample forecasting of the cointegration relation, we find that the two-regime SETAR(3) model gives much better prediction than an AR(1), an AR(3) and a two-regime SETAR(1) model.

It is a remarkable fact that the evidence for a threshold model being superior to a linear model completely disappears when removing 6 outliers from the NIBOR tomorrow next series and 2 outliers from the NIBOR 12 months series by interpolation between adjacent values. Certainly, it is not recommended to remove outliers uncritically by using interpolation, but this analysis shows how influential a few outliers may be. Hence, it is very important how we treat the outliers. Naturally, if there are errors in the series, e.g., typing mistakes, it is surely okay to correct the time series. On the other hand, if an outlier in a financial time series is caused by e.g., a financial shock, Juselius (2006) recommends including a dummy variable to explain this shock. It is well known that the outliers in September, November and December 1992 are explained by the turbulence in the financial markets when the fixed exchange rate regime of the Norwegian krone was abandoned. Hence, we should include dummy variables for these 3 months in our analysis. However, dummy variables are not implemented in the R package **tsDyn**, so the best we can do is to remove these three outliers by using interpolation. If we remove only these three outliers, then the two-regime SETAR(3) model is still superior to an AR(3) model with approximately the same P-value as when no outliers are removed.

We may consider a TVECM as a tool to take care of outliers. By using large coefficients of the error correction term in the outer regime(s), we achieve that the time series returns rapidly back to ordinary values. The residuals of the TVECM have better properties than the residuals of the VAR(p) model we started with. The residuals in the TVECM are on the average smaller in absolute value than the residuals in the VAR(p) model as the SSR is lowered. The LM statistic for multivariate ARCH effects is considerably smaller for the residuals of the TVECM than for the residuals of the VAR(p) model such that there are not any significant ARCH effects in the residuals of the TVECM. However, in the normality test the residuals of the TVECM do not behave better than the residuals of the VAR(p) model. The null hypothesis of normally distributed residuals is strongly rejected, also for the residuals of the TVECM. The reason is probably the large outliers. Summing up, the residuals of the TVECM have better statistical properties than the residuals of the VAR(p) model, probably due to the ability of the TVECM to model nonlinearities, but it is not sufficient. In addition, dummy variables should be used whenever it is appropriate.

In Chapter 5 we have simulated a bivariate time series with a three-regime TVECM as data generation process, and estimated a three-regime TVECM from this simulated time series. As expected, the coefficients of the estimated TVECM are close to the coefficients of the three-regime TVECM which was used in the simulation. Though, the function ***TVECM*** found the original thresholds only when large search intervals were specified for the thresholds in the ***TVECM*** call. When running the function ***TVECM.XHStest*** on this estimated TVECM, the null hypothesis of linear cointegration is strongly rejected. Further, both the **SupLM** statistic and the minimal SSR occurs for thresholds which are very close to the original thresholds used in the simulation. On the contrary, when runnning ***TVECM.XHStest*** on real data, e.g., interest rates, the thresholds minimizing SSR are often very different from the thresholds maximizing the LM statistic, which is a bit strange.

The cointegration relation of the simulated time series was modeled by using a SETAR(3) model. When testing one regime against two regimes, the null hypothesis of one regime is strongly rejected. When testing two regimes against three regimes, the null hypothesis of two regimes is strongly rejected. Therefore, a three-regime threshold model should be used for this data set, which is in accordance with the fact that the data was generated from a three-regime TVECM.

We have also investigated the NIBOR rates of 9 different maturites and the rates on Norwegian government bonds of 3 maturities systematically. In all these 12 time series we have removed the outliers in autumn 1992 by interpolation between adjacent values. There is highly significant cointegration in each of the 36 pairs of NIBOR rates, while there is no cointegration in the three pairs of government bonds. When modeling the term spread of the 66 pairs of interest rates, a two-regime SETAR(4) model is superior to an AR(4) model at the 5 % level in approximately 50 % of the cases. So, we may expect significant threshold cointegration in a lot of these bivariate time series. In addition, we have found that a three-regime SETAR(4) model is superior to a two-regime SETAR(4) model for the term spread in some cases. Consequently, we may expect to find that a three-regime TVECM is superior to a two-regime TVECM for some of the interest rate pairs. Indeed, this is an important topic for future research, but cannot be performed until the following limitations and bugs in **tsDyn** are resolved:

- According to economic theory, the models should have a wide middle regime where the interest rates are quite stable, and much narrower outer regimes where the interest rates are unstable. Therefore, a lot of the cases with significant threshold effects in Table 4.25 on page 62 and Table 4.26 on page 63 have a too narrow middle regime to be interesting in practice. Maybe, the size of the regimes in the functions ***TVECM***, ***TVECM.XHStest***, ***setar*** and ***setarTest*** should be bounded both below and above with specific bounds for each regime. In this way, we achieve that only models with reasonable sizes of the regimes are taken into consideration when estimating TVECMs and SETAR models, and when testing these models for significant threshold effects. Hence, the elapse time of the very time-consuming functions ***setarTest*** and ***TVECM.XHStest*** would be considerably reduced.

- Table 4.28 shows that the number of cases with significant threshold effects increases considerably when the optimal threshold delay is chosen instead of the default value 0. As the threshold delay is not implemented in the functions ***TVECM*** and ***TVECM.XHStest***, we are neither able to estimate TVECMs with threshold delay, nor able to test such TVECMs for threshold effects. In addition, in some cases the function ***setarTest*** fails when testing a two-regime model against a three-regime model. (See the blank fields in the fifth column of Table 4.26 on page 63.)

- The grid search for β and the threshold(s) in the function ***TVECM*** is far from perfect as the search intervals for β and the threshold(s) have to be chosen by care; if not, the global minimum of SSR is not found. As all the tests for threshold effects are dependent on this β value, the results of these tests may be unreliable.

- Since dummy variables are recommended to take care of outliers, the R package **tsDyn** should be extended such that dummy variables may be included when estimating and testing TVECMs, SETAR models and VAR models.

Bibliography

Aho, K. (2011). *asbio: A collection of statistical tools for biologists.* R package version 0.3-40. URL: http://cran.r-project.org/web/packages/asbio/asbio.pdf.

Arize, A.; Malindretos, J.; Obi, Z. (2002). "Long- and short-term interest rates in 19 countries: Tests of cointegration and parameter instability". *Atlantic Economic Journal* 30 (2), pp. 105–120. ISSN: 0197-4254. URL: http://dx.doi.org/10.1007/BF02299156.

Balke, N.S.; Fomby, T.B. (1997). "Threshold Cointegration". English. *International Economic Review* 38.3, pp. 627–645. ISSN: 00206598. URL: http://www.jstor.org/stable/2527284.

Buigut, S.; Rao, V.K. (2010). "Exceptation Hypothesis and the Term Structure of Hong Kong Interbank Rates". *International Research Journal of Finance and Economics* (39), pp. 72–85. ISSN: 1450-2887. URL: http://www.eurojournals.com/irjfe_39_07.pdf.

Campbell, J.Y.; Shiller, R.J. (1991). "Yield Spreads and Interest Rate Movements: A Bird's Eye View". English. *The Review of Economic Studies* 58.3, pp. 495–514. ISSN: 00346527. URL: http://www.jstor.org/stable/2298008.

Clements, M.P.; Galvão, A.B. (2004). "A comparison of tests of nonlinear cointegration with application to the predictability of US interest rates using the term structure". *International Journal of Forecasting* 20.2, pp. 219 –236. ISSN: 0169-2070. URL: http://www.sciencedirect.com/science/article/pii/S0169207003000992.

Di Narzo, A.F. (2008). *Nonlinear autoregressive time series models in R using tsDyn version 0.7.* URL: http://cran.r-project.org/web/packages/tsDyn/vignettes/tsDyn.pdf.

Di Narzo, A.F.; Aznarte, J.L.; Stigler, M. (2011). *tsDyn: Nonlinear time series models with regime switching.* R package version 0.7-60. URL: http://cran.r-project.org/web/packages/tsDyn/index.html.

Doornik, J.A.; Hansen, H. (1994). *An omnibus test for univariate and multivariate normalit.* Economics Papers W4&91. Economics Group, Nuffield College, University of Oxford. URL: http://ideas.repec.org/p/nuf/econwp/9604.html.

Engle, R.F. (Jan. 1984). "Wald, likelihood ratio, and Lagrange multiplier tests in econometrics". In: *Handbook of Econometrics.* Ed. by Z. Griliches and M. D. Intriligator. Vol. 2. Handbook of Econometrics. Elsevier. Chap. 13, pp. 775–826. URL: http://ideas.repec.org/h/eee/ecochp/2-13.html.

Engle, R.F.; Granger, C.W.J. (1987). "Co-integration and Error Correction: Representation, Estimation, and Testing". *Econometrica* 55.2, pp. 251–76. URL: http://EconPapers.repec.org/RePEc:ecm:emetrp:v:55:y:1987:i:2:p:251-76.

Engsted, T. (1996). "The predictive power of the money market term structure". *International Journal of Forecasting* 12.2, pp. 289 –295. ISSN: 0169-2070. URL: http://www.sciencedirect.com/science/article/pii/0169207095006249.

Gjedrem, S. (1999). *Monetary policy challenges.* URL: http://m.norges-bank.no/en/about/published/articles-and-chronicles/gjed999htm/.

Granger, C.W.J. (May 1981). "Some properties of time series data and their use in econometric model specification". *Journal of Econometrics* 16.1, pp. 121–130. URL: http://ideas.repec.org/a/eee/econom/v16y1981i1p121-130.html.

Hall, A.D.; Anderson, H.M.; Granger, C.W.J. (1992). "A Cointegration Analysis of Treasury Bill Yields". English. *The Review of Economics and Statistics* 74.1, pp. 116–126. ISSN: 00346535. URL: http://www.jstor.org/stable/2109549.

Hansen, B.E. (1999). "Testing for Linearity". *Journal of Economic Surveys* 13(5), pp. 551–576. ISSN: 0950-0804. URL: http://www.ssc.wisc.edu/~bhansen/papers/jes_99.pdf.

Hansen, B.E.; Seo, B. (Oct. 2002). "Testing for two-regime threshold cointegration in vector error-correction models". *Journal of Econometrics* 110.2, pp. 293–318. URL: http://ideas.repec.org/a/eee/econom/v110y2002i2p293-318.html.

Johansen, S. (1988). "Statistical analysis of cointegration vectors". *Journal of Economic Dynamics and Control* 12.2 - 3, pp. 231 –254. ISSN: 0165-1889. URL: http://www.sciencedirect.com/science/article/pii/0165188988900413.

Johansen, S.; Juselius, K. (May 1990). "Maximum Likelihood Estimation and Inference on Cointegration–With Applications to the Demand for Money". *Oxford Bulletin of Economics and Statistics* 52.2, pp. 169–210. URL: http://ideas.repec.org/a/bla/obuest/v52y1990i2p169-210.html.

Juselius, K. (2006). *The cointegrated VAR model: Methodology and Applications.* New York: Oxford University Press. ISBN: 978-0-19-928567-9.

Kwiatkowski, D. (1992). "Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?" *Journal of Econometrics* 54.1-3, pp. 159 –178. ISSN: 0304-4076. URL: http://www.sciencedirect.com/science/article/pii/030440769290104Y.

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis.* Springer Verlag. ISBN: 978-3-540-26239-8.

Modigliani, F.; Shiller, R.J. (1973). "Inflation, Rational Expectations and the Term Structure of Interest Rates". English. *Economica.* New Series 40.157, pp. 12–43. ISSN: 00130427. URL: http://www.jstor.org/stable/2552679.

Musti, S.; D'Ecclesia, R.L. (Mar. 2008). "Term structure of interest rates and the expectation hypothesis: The euro area". *European Journal of Operational Research* 185.3, pp. 1596–1606. URL: http://ideas.repec.org/a/eee/ejores/v185y2008i3p1596-1606.html.

Pfaff, B. (2008a). *Analysis of Integrated and Cointegrated Time Series with R.* Second. New York: Springer. ISBN: 978-0-387-75966-1. URL: http://www.pfaffikus.de.

— (2008b). "VAR, SVAR and SVEC Models: Implementation Within R Package vars". *Journal of Statistical Software* 27.4. URL: http://www.jstatsoft.org/v27/i04/.

Seo, B. (2003). "Nonlinear mean reversion in the term structure of interest rates". *Journal of Economic Dynamics & Control* 27, pp. 2243–2265.

Shumway, R.H.; Stoffer, D.S. (2006). *Time Series Analysis and Its Applications.* Second. New York: Springer. ISBN: 978-0-387-29317-2.

Siklos, P.L.; Wohar, M.E. (1996). “Cointegration and the term structure: A multicountry comparison”. *International Review of Economics & Finance* 5.1, pp. 21–34. URL: http://ideas.repec.org/a/eee/reveco/v5y1996i1p21-34.html.

Stigler, M. (2011). *Threshold cointegration: overview and implementation in R.* R package tsDyn version 0.7-60. URL: http://cran.r-project.org/web/packages/tsDyn/vignettes/ThCointOverview.pdf.

Swinton, J. (2011). *The xtable gallery.* R package xtable version 1.6-0. URL: http://cran.uib.no/web/packages/xtable/vignettes/xtableGallery.pdf.

Tsay, R.S. (2010). *Analysis of Financial Time Series.* Third. New Jersey: Wiley. ISBN: 978-0-470-41435-4.

Zeileis, A.; Grothendieck, G. (2005). “zoo: S3 Infrastructure for Regular and Irregular Time Series”. *Journal of Statistical Software* 14.6, pp. 1–27. URL: http://www.jstatsoft.org/v14/i06/.

Appendices

Appendix A

The source code of TVECM.HSTest

This appendix contains the source code of the function ***TVECM.HSTest*** in the version 0.7-40 of the package **tsDyn** which was the starting point for our work on the Hansen and Seo test, while the next appendix contains the source code of our new version of this function. To save space, only the subfunctions where we have done changes, are included in these appendices.

```
TVECM.HStest <- function(data, lag=1, ngridTh=300, trim=0.05, nboot=100,
    fixed.beta=NULL,  intercept=TRUE, boot.type=c("FixedReg", "ResBoot"),
    hpc=c("none", "foreach"), common=c("All", "ECT"), type=c("2Reg", "SymReg")) {
## Check args:
boot.type<-match.arg(boot.type)
hpc<-match.arg(hpc)
common<-match.arg(common)
type<-match.arg(type)
dir=FALSE #internal value, was used to try different implementation of lmtest
### Organize Data
data<-as.matrix(data)
if(ncol(data)>2) {warning("Please no more than two equations")}
if(is.null(colnames(data))){colnames(data)<-paste("Var", c(1:2), sep="")}
T<-nrow(data)
p<-lag
y<-diff(data)[(p+1):(T-1),]
DeltaX<-embed(diff(data),p+1)[,-(1:2)]
if(intercept)  DeltaX<-cbind(1, DeltaX)
x<-DeltaX
t<-nrow(y)
### Compute beta with VECM() and extract ect
if(is.null(fixed.beta)){
  ve<-VECM(data, lag=lag, include="const", estim="ML")
}else{
  ve<-VECM(data, lag=lag, include="const",  beta=fixed.beta, estim="2OLS")
}
ect<-ve$model[,grep("ECT", colnames(ve$model))]
w0<-matrix(ect[!is.na(ect)], ncol=1)
###Set up of the grid
```

```
q<- if(type=="2Reg") sort(w0) else sort(abs(w0))
if(ngridTh>(1-2*trim)*t) {
  newNgridTh<-round((1-2*trim)*t-1)
  warning("ngridTh (", ngridTh,
      ") bigger than number of potential threshold values, set to ",newNgridTh,"\n")
  ngridTh<-newNgridTh
}
gamma2<-q[round(seq(from=trim*t, to=(1-trim)*t,length.out=ngridTh))]
gamma2<-unique(gamma2)
ngridTh<-length(gamma2)
###########
###Lm Test
###########
lmtest02<-function(y,x,w0,gammas,dir=dir){
#y: y var, x: intercept and lags matrix, w0: ECT term, gammas: potential thresholds
  X<-cbind(w0,x)                 #X: ECT and intercept and lags
  if(dir){
    q<-qr(X)
    res_restr<-qr.resid(q,y)
  } else{
    z0zz<-X%*%solve(t(X)%*%X)
    res_restr<-lm.fit(X,y)$residuals        #residuals from the linear VECM given b0
  }
  res_restr1<-res_restr[,1]
  res_restr2<-res_restr[,2]
  store<-rep(NA, ngridTh)
  Ttrim<-trim*t
  ngridTh<-min(t*(1-2*trim), length(gammas))
  for(j in 1:ngridTh){
    d1<-if(type=="2Reg") ifelse(w0<=gammas[j],1,0) else
        ifelse(w0<= - gammas[j] | w0> gammas[j],1,0) #d1: dummy variable
    n1<-sum(d1)
    if (min(c(n1,(t-n1)))>Ttrim){
      z1<-if(common=="All") c(d1)*X else c(d1)*w0
      res_unrestr <-if(dir) qr.resid(q, z1) else z1-z0zz%*%(t(X)%*%z1)
          #z11: residuals from unrestricted model (with threhsold)
      zea<-res_restr1*res_unrestr
      zeb<-res_restr2*res_unrestr
      ze<-cbind(zea,zeb)
          #[(z11.*(e1*ones(1,length(z11(1,:))))),(z11.*(e2*ones(1,length(z11(1,:)))))];
      v<-crossprod(ze)
      z11y<-crossprod(res_unrestr,y)
      s<-matrix(c(z11y), ncol=1)  #vectorization of the parameter matrix z11y
      VV<-crossprod(v)
      VVinv<-try(solve(VV), silent=TRUE)
      if(inherits(VVinv, "try-error")) VVinv<-ginv(VV)
      store[j]<-t(s)%*%VVinv%*%t(v)%*%s
    } #end of the if
  } #end of the whole loop
  return(store)
} #end of the function lmtest01
```

```
#### LM test for fixed regressor bootstrap: X'X^-1 evaluated only once
lmtest02_boot<-function(y,x,w0,gammas,dir=dir){
  X<-cbind(w0,x)                 #X: ECT and intercept and lags
  if(dir){
    res_restr<-qr.resid(q,y)
  } else{
    res_restr<-lm.fit(X,y)$residuals        #residuals from the linear VECM given b0
  }
  res_restr1<-res_restr[,1]
  res_restr2<-res_restr[,2]
  store<-rep(0, ngridTh)
  ngridTh<-min(t*(1-2*trim), length(gammas))
  for(j in 1:ngridTh){
    d1<-if(type=="2Reg") ifelse(w0<=gammas[j],1,0) else
        ifelse(w0<= - gammas[j] | w0> gammas[j],1,0) #d1: dummy variable
    n1<-sum(d1)
    if (min(c(n1,(t-n1)))>Ttrim){
      z1<-if(common=="All") c(d1)*X else c(d1)*w0
      res_unrestr <-if(dir) qr.resid(q, z1) else z1-z0zz%*%(t(X)%*%z1)
          #z11: residuals from unrestricted model (with threhsold)
      zea<-res_restr1*res_unrestr
      zeb<-res_restr2*res_unrestr
      ze<-cbind(zea,zeb)
        # [(z11.*(e1*ones(1,length(z11(1,:))))),(z11.*(e2*ones(1,length(z11(1,:)))))];
      v<-crossprod(ze)
      z11y<-crossprod(res_unrestr,y)
      s<-matrix(c(z11y), ncol=1) #vectorization of the parameter matrix z11y
      VV<-crossprod(v)
      VVinv<-try(solve(VV), silent=TRUE)
      if(inherits(VVinv, "try-error")) VVinv<-ginv(VV)
      store[j]<-t(s)%*%VVinv%*%t(v)%*%s
    } #end of the if
  } #end of the whole loop
  lm01<-max(store, na.rm=TRUE)
  lm01
} #end of the function lmtest01
###
lm01<-lmtest02(y,x,w0,gamma2, dir=dir)
teststat<-max(lm01, na.rm=TRUE)
##################################
### Bootstraps
##################################
if(nboot==0){
  CriticalValBoot<-NULL
  PvalBoot<-NULL
  boots.reps<-NULL
  if(hpc=="foreach") warning("hpc='foreach' used only when nboot>0\n")
}else if (nboot>0){
##################################
### Fixed Regressor Bootstrap %
##################################
```

```
  if(boot.type=="FixedReg"){
    X<-cbind(w0,x)  #X: ECT and intercept and lags
    Ttrim<-trim*t
    if(dir){
      q<-qr(X)
    } else{
      z0zz<-X%*%solve(t(X)%*%X)
    }

    lmtest_withBoot<-function(e){
      yr<-rnorm(n=t,0,1)*e
      return(lmtest02_boot(yr,x,w0,gamma2,dir=dir))
    }
    boots.reps<- if(hpc=="none") replicate(nboot, lmtest_withBoot(e=residuals(ve)))
        else foreach(i=1:nboot, .export="lmtest_withBoot", .combine="c")
            %dopar% lmtest_withBoot(e=residuals(ve))
###################################
### Residual Bootstrap
###################################
  } else{
      lmtest_with_resBoot<-function(ve){
      #bootstrap it
        data.boot<-TVECM.sim(TVECMobject=ve, type="boot")
      # estimate VECM
        if(is.null(fixed.beta)){
          ve.boot<-VECM(data.boot, lag=lag, include="const", estim="ML")
        }else{
          ve.boot<-VECM(data.boot, lag=lag, include="const",  beta=fixed.beta,
              estim="2OLS")
        }
      # extract w0, y and x
        ect.boot<-ve.boot$model[,"ECT"]
        which.nas<-1:(p+1)
        w0.boot<-matrix(ect.boot[-which.nas], ncol=1)
        x.boot<-ve.boot$model[-which.nas,-c(1:3)]
        y.boot<-ve.boot$model[,c(1:2)]
        y.boot<-diff(y.boot)[(p+1):(T-1),]
      # set-up grid
        w0.ord.boot<-sort(w0)
        gamma2.boot<-w0.ord.boot[round(seq(from=trim*T,
            to=(1-trim)*T,length.out=ngridTh))]
        gamma2.boot<-unique(gamma2.boot)
        ngridTh.boot<-length(gamma2.boot)
        test.boot<-lmtest02(y.boot,x.boot,w0.boot,gamma2.boot,dir=dir)
        return(max(test.boot, na.rm=TRUE))
      }
      boots.reps<- if(hpc=="none") replicate(nboot, lmtest_with_resBoot(ve)) else
          foreach(i=1:nboot,.export="lmtest_with_resBoot", .combine="c")
              %dopar% lmtest_with_resBoot
  }#end if boot= ResBoot
## result: compute p values and critical values:
   PvalBoot<-mean(ifelse(boots.reps>teststat,1,0))
   CriticalValBoot<-quantile(boots.reps, probs= c(0.9, 0.95,0.99))
}#end if boot>0
####### Return args
args<-list()
args$nboot<-nboot
args$boot.type<-boot.type
ret<-list()
ret$args<-args
ret$stat<-teststat
ret$values<-lm01
ret$ths<-gamma2
ret$maxTh<-gamma2[which(lm01==ret$stat)]
ret$PvalBoot<-PvalBoot
ret$CriticalValBoot<-CriticalValBoot
ret$allBoots<-boots.reps
class(ret)<-"TVECMHanSeo02Test"
return(ret)
}#End of the whole function

### Print method
print.TVECMHanSeo02Test<-function(x,...){
  cat("## Test of linear versus threshold cointegration",
      " of Hansen and Seo (2002) ##\n\n", sep="")
  cat("Test Statistic:\t", x$stat)
  cat("\t(Maximized for threshold value:", x$maxTh, ")\n")
  if(x$args$nboot>0){
      boot.name<-switch(x$args$boot.type, "FixedReg"="Fixed regressor bootstrap",
          "ResBoot"="Residual Bootstrap")
      cat("P-Value:\t", x$PvalBoot, "\t\t(",boot.name, ")\n")
    }
}
```

Appendix B

The source code of TVECM.XHSTest

This appendix contains the source code of our new version of the Hansen and Seo test, which includes the extension to three regimes and an improved grid search algorithm.

```
TVECM.XHStest <- function(data, lag=1, ngridTh=300, trim=0.05, nboot=100,
     fixed.beta=NULL,  intercept=TRUE, boot.type=c("FixedReg", "ResBoot"),
     hpc=c("none", "foreach"), common=c("All", "ECT"),
     type=c("2Reg", "SymReg", "3Reg"), tolerance=1e-12,trace=TRUE,dir=FALSE) {
## Check args:
boot.type<-match.arg(boot.type)
hpc<-match.arg(hpc)
common<-match.arg(common)
type<-match.arg(type)
#dir=FALSE #internal value, was used to try different implementation of lmtest
### Organize Data
data<-as.matrix(data)
if(ncol(data)>2) {warning("Please no more than two equations")}
if(is.null(colnames(data))){colnames(data)<-paste("Var", c(1:2), sep="")}
T<-nrow(data)
p<-lag
y<-diff(data)[(p+1):(T-1),]
DeltaX<-embed(diff(data),p+1)[,-(1:2)]
if(intercept)  DeltaX<-cbind(1, DeltaX)
x<-DeltaX
t<-nrow(y)
### Compute beta with VECM() and extract ect
if(is.null(fixed.beta)){
  ve<-VECM(data, lag=lag, include="const", estim="ML")
}else{
  ve<-VECM(data, lag=lag, include="const",  beta=fixed.beta, estim="2OLS")
}
ect<-ve$model[,grep("ECT", colnames(ve$model))]
w0<-matrix(ect[!is.na(ect)], ncol=1)
w0.ord<- if(type=="2Reg" | type=="3Reg") sort(w0) else sort(abs(w0))
Ttrim<-ceiling(trim*t)
###Set up of the grid
```

```
GridSetup<-function(allgammas){
    Min1<-allgammas[Ttrim] #At least element [1:Min1] to lower regime such that
                           #at least 100trim % goes to lower regime
    Max1<-allgammas[t-Ttrim+1] #At least element [(t-Trim+1):t] to upper
        # regime such that at least 100trim % goes to upper regime
    if (Min1>Max1) stop("The parameter trim is too large\n")
    gamma2<-allgammas #allgammas: all possible threshold values
    i<-2
    while (i <= length(gamma2)) {
        if (abs(gamma2[i-1]-gamma2[i])< tolerance) {
            gamma2<-gamma2[-i]
        } else {
            i<-i+1
        }
    } #gamma2: all possible threshold values, but now all repetitions are removed,
      #i.e. the difference between two consequtive elements are always larger
      #than tolerance
    iMin<-max(which(gamma2<=Min1,arr.ind=TRUE)) #The index of the element of
      #gamma2 which is the smallest possible value of the lower threshold.
    iMax<-max(which(gamma2<=Max1,arr.ind=TRUE))-1 #The index of the element of
      # gamma2 which is the largest possible value of the upper threshold.
    if (iMin>iMax) {
        stop("iMin>iMax probably because the parameter tolerance is too large\n")
    }
    gamma2<-gamma2[iMin:iMax] #The threshold value(s) have to be in this set
    newNgridTh<-length(gamma2)
    if(ngridTh>newNgridTh) {
        warning("ngridTh (", ngridTh,
          ") >= the number of potential threshold values, set to ",
                newNgridTh, "\n")
        ngridTh<-newNgridTh
    }
    else {
        warning("ngridTh (", ngridTh,
          ") < the number of potential threshold values, ",newNgridTh,
               "so only a subset of the potential threshold values is selected.\n")
        if (trace) {
            cat("The set of possible threshold values:\n")
            print(gamma2)
        }
        gamma2<-gamma2[round(seq(from=1, to=newNgridTh,length.out=ngridTh))]
        if (trace) {
            cat("The set of the selected threshold values:\n")
            print(gamma2)
        }
    }
    gamma2<-gamma2+tolerance #necessary to ensure that the condition w_t<=gamma2[j]
#    is TRUE also in the case that w_t is slightly larger than gamma2[j] due to
#    inaccuracies in floating point numbers.
    return(gamma2)
}
```

```
###########
###Lm Test when type="2Reg" or "SymReg". This function returns the whole set of
###LM-values such that the threshold values corresponding to the supLM value may be
###found later. When bootstrapping we use the function lmtest2RegSymReg_boot which
###returns only the supLM value. In addition, SSR is computed for each threshold
###value in lmtest2RegSymReg, but not in lmtest2RegSymReg_boot.  These are the only
###differences between these two functions.
###########
lmtest2RegSymReg<-function(y,x,w0,gammas,dir=dir){
#y: y var, x: intercept and lags matrix, w0: ECT term, gammas: potential thresholds
    ngridTh<-length(gammas)
    X<-cbind(w0,x)                  #X: ECT and intercept and lags
    if(dir){
        q<-qr(X)
        res_restr<-qr.resid(q,y)
    } else{
        z0zz<-X%*%solve(t(X)%*%X)
        res_restr<-lm.fit(X,y)$residuals #residuals from the linear VECM given b0
    }
    res_restr1<-res_restr[,1]
    res_restr2<-res_restr[,2]
    store<-rep(0, ngridTh)
    SSR<-rep(10000, ngridTh)
    for(i in 1:ngridTh){
        d1<-if(type=="2Reg") ifelse(w0<=gammas[i],1,0) else
            ifelse(w0<= - gammas[i] | w0> gammas[i],1,0) #d1: dummy variable
        z1<-if(common=="All") c(d1)*X else c(d1)*w0
        res_unrestr <-if(dir) qr.resid(q, z1) else z1-z0zz%*%(t(X)%*%z1)
             #z11: residuals from unrestricted model (with threhsold)
        zea<-res_restr1*res_unrestr
        zeb<-res_restr2*res_unrestr
        ze<-cbind(zea,zeb)
          # [(z11.*(e1*ones(1,length(z11(1,:))))),(z11.*(e2*ones(1,length(z11(1,:)))))];
        v<-crossprod(ze)
        z11y<-crossprod(res_unrestr,y)
        s<-matrix(c(z11y), ncol=1) #vectorization of the parameter matrix z11y
        Vinv<-try(solve(v), silent=TRUE)
        if(inherits(Vinv, "try-error")) Vinv<-ginv(v)
        store[i]<-t(s)%*%Vinv%*%s
        d2<-1-d1
        z2<-c(d2)*X
        Z<-cbind(z1,z2)
        SSR[i]<-crossprod(c(qr.resid(qr(Z),y)))
    } #end of the whole loop
    ret<-list()
    ret$store<-store
    ret$SSR<-SSR
    return(ret)
} #end of the function lmtest2RegSymReg
###Lm Test for bootstrapping when type="2Reg" or "SymReg". This function returns only
###the supLM value. When not bootstrapping we use the function lmtest2RegSymReg which
```

```
###returns the whole set of the LM values, and the SSR values. This is the only
###difference between these two functions.
###########
lmtest2RegSymReg_boot<-function(y,x,w0,gammas,dir=dir){
#y: y var, x: intercept and lags matrix, w0: ECT term, gammas: potential thresholds
    ngridTh<-length(gammas)
    X<-cbind(w0,x)                  #X: ECT and intercept and lags
    if(dir){
        q<-qr(X)
        res_restr<-qr.resid(q,y)
    } else{
        z0zz<-X%*%solve(t(X)%*%X)
        res_restr<-lm.fit(X,y)$residuals #residuals from the linear VECM given b0
    }
    res_restr1<-res_restr[,1]
    res_restr2<-res_restr[,2]
    store<-rep(0, ngridTh)
    for(i in 1:ngridTh){
        d1<-if(type=="2Reg") ifelse(w0<=gammas[i],1,0) else
            ifelse(w0<= - gammas[i] | w0> gammas[i],1,0) #d1: dummy variable
        z1<-if(common=="All") c(d1)*X else c(d1)*w0
        res_unrestr <-if(dir) qr.resid(q, z1) else z1-z0zz%*%(t(X)%*%z1)
            #z11: residuals from unrestricted model (with threshold)
        zea<-res_restr1*res_unrestr
        zeb<-res_restr2*res_unrestr
        ze<-cbind(zea,zeb)
          #[(z11.*(e1*ones(1,length(z11(1,:))))),(z11.*(e2*ones(1,length(z11(1,:)))))];
        v<-crossprod(ze)
        z11y<-crossprod(res_unrestr,y)
        s<-matrix(c(z11y), ncol=1) #vectorization of the parameter matrix z11y
        Vinv<-try(solve(v), silent=TRUE)
        if(inherits(Vinv, "try-error")) Vinv<-ginv(v)
        store[i]<-t(s)%*%Vinv%*%s
    } #end of the whole loop
    lm01<-max(store, na.rm=TRUE)
    return(lm01)
} #end of the function lmtest2RegSymReg_boot
###########
###Lm Test when type="3Reg". This function returns the whole set of
###LM-values such that the threshold values corresponding to the supLM value may be
###found later. When bootstrapping we use the function lmtest3Reg_boot which returns
###only the supLM value. In addition, SSR is computed for each pair of threshold
###values in lmtest3Reg, but not in lmtest3Reg_boot.  These are the only
###differences between these two functions.
###########
lmtest3Reg<-function(y,x,w0,gammas,dir=dir){
#y: y var, x: intercept and lags matrix, w0: ECT term, gammas: potential thresholds
    ngridTh<-length(gammas)
    w0.ord<-sort(w0)+tolerance #The comparisons between w0.ord and gammas decide
    #  the limits of the loops below. As we have already added tolerance to gammas, we
    # also have to add tolerance to w0.ord, to get correct comparisons.
    X<-cbind(w0,x)                  #X: ECT and intercept and lags
    if(dir){
        q<-qr(X)
        res_restr<-qr.resid(q,y)
    } else{
        z0zz<-X%*%solve(t(X)%*%X)
        res_restr<-lm.fit(X,y)$residuals        #residuals from the linear VECM given b0
    }
    res_restr1<-res_restr[,1]
    res_restr2<-res_restr[,2]
    store<-matrix(0, ncol=ngridTh,nrow=ngridTh)
    SSR<-matrix(100000, ncol=ngridTh,nrow=ngridTh)
    Max2<-w0.ord[which(abs(w0.ord-w0.ord[t-Ttrim])<tolerance,arr.ind=TRUE)[1]-Ttrim]
    # which(abs(w0.ord-w0.ord[t-Ttrim])<tolerance,arr.ind=TRUE)[1] computes the
    # index in w0.ord of the smallest element in w0.ord equal to  w0.ord[t-Ttrim] except
    # for floating point error,  so Max2 is the element in w0.ord with index Trim
    # smaller than the computed index by which(....)[1].
    iMax<-max(which(gammas<=Max2,arr.ind=TRUE)) #The index of the largest element
      # in gammas which is <=Max2
    for(i in 1:iMax){
        d1<- ifelse(w0<=gammas[i],1,0)  #d1: dummy variable
        z1<-if(common=="All") c(d1)*X else c(d1)*w0
        help<-which(abs(w0.ord-gammas[i])<tolerance,arr.ind=TRUE) #The index of the
          # elements in w0.ord which are =gammas[i] except for floating point error
        Min2<-w0.ord[help[length(help)]+Ttrim]-tolerance #Min2 is the element in
          # w0.ord which index is Trim larger than the index in w0.ord of the
          # largest element in w0.ord which is equal to gammas[i]. The tolerance
          # is subtracted due to the test against gammas in the next two statements.
#         help<-length(which(abs(gammas-Min2)<tolerance,arr.ind=TRUE))
        if (gammas[ngridTh] > Min2) { #Min2 in gammas, i.e. the middle
              # regime contains at least
              # Ttrim elements; if help=0, gammas[i] is so large that the middle
              # regime contains less than Trim elements, which means that no LM
              # and SSR values should be computed for this value of i
            jMin<-min(which(gammas >= Min2,arr.ind=TRUE))
                #The index of the least element >= Min2 in gammas
            for (j in jMin:ngridTh) {
                d3<- ifelse(w0>gammas[j],1,0)  #d1: dummy variable
                z3<-if(common=="All") c(d3)*X else c(d3)*w0
                z<-cbind(z1,z3)
                res_unrestr <-if(dir) qr.resid(q, z) else z-z0zz%*%(t(X)%*%z)
                    #z11: residuals from unrestricted model (with threhsold)
                zea<-res_restr1*res_unrestr
                zeb<-res_restr2*res_unrestr
                ze<-cbind(zea,zeb)
       # [(z11.*(e1*ones(1,length(z11(1,:))))),(z11.*(e2*ones(1,length(z11(1,:)))))];
                v<-crossprod(ze)
                z11y<-crossprod(res_unrestr,y)
                s<-matrix(c(z11y), ncol=1)
                    #vectorization of the parameter matrix z11y
                Vinv<-try(solve(v), silent=TRUE)
                if(inherits(Vinv, "try-error")) Vinv<-ginv(v)
                store[i,j]<-t(s)%*%Vinv%*%s
                d2<-1-d1-d3
                z2<-c(d2)*X
                Z<-cbind(z1,z2,z3)
                SSR[i,j]<-crossprod(c(qr.resid(qr(Z),y)))
            } # end for (j in jMin:ngridTh) {
        } #end if (help>0) {
    } #end of the whole loop
    ret<-list()
    ret$store<-store
    ret$SSR<-SSR
    return(ret)
} #end of the function lmtest3Reg
###Lm Test for bootstrapping when type="3Reg". This function returns only the supLM
###value. When not bootstrapping, we use the function lmtest3Reg which
###returns the whole set of LM values, and the SSR values. This is the only
###difference between these two functions.
###########
lmtest3Reg_boot<-function(y,x,w0,gammas,dir=dir){
#y: y var, x: intercept and lags matrix, w0: ECT term, gammas: potential thresholds
    ngridTh<-length(gammas)
    w0.ord<-sort(w0)+tolerance #The comparisons between w0.ord and gammas decide
    #  the limits of the loops below. As we have already added tolerance to gammas,
    # we also have to add tolerance to w0.ord, to get correct comparisons.
    X<-cbind(w0,x)                #X: ECT and intercept and lags
    if(dir){
        q<-qr(X)
        res_restr<-qr.resid(q,y)
    } else{
        z0zz<-X%*%solve(t(X)%*%X)
        res_restr<-lm.fit(X,y)$residuals # residuals from the linear VECM given b0
    }
    res_restr1<-res_restr[,1]
    res_restr2<-res_restr[,2]
    store<-matrix(0, ncol=ngridTh,nrow=ngridTh)
    Max2<-w0.ord[which(abs(w0.ord-w0.ord[t-Ttrim])<tolerance,arr.ind=TRUE)[1]-Ttrim]
    # which(abs(w0.ord-w0.ord[t-Ttrim])<tolerance,arr.ind=TRUE)[1] computes the
    # index in w0.ord of the smallest element in w0.ord equal to  w0.ord[t-Ttrim] except
    # for floating point error,  so Max2 is the element in w0.ord with index Trim
    # smaller than the computed index by which(....)[1].
    iMax<-max(which(gammas<=Max2,arr.ind=TRUE)) #The index of the largest element
      # in gammas which is <=Max2
    for(i in 1:iMax){
        d1<- ifelse(w0<=gammas[i],1,0)  #d1: dummy variable
        z1<-if(common=="All") c(d1)*X else c(d1)*w0
        help<-which(abs(w0.ord-gammas[i])<tolerance,arr.ind=TRUE) #The index of the
          # elements in w0.ord which are =gammas[i] except for floating point error
        Min2<-w0.ord[help[length(help)]+Ttrim]-tolerance #Min2 is the element in
          # w0.ord which index is Trim larger than the index in w0.ord of the
          # largest element in w0.ord which is equal to gammas[i]. The tolerance
          # is subtracted due to the test against gammas in the next two statements.
#         help<-length(which(abs(gammas-Min2)<tolerance,arr.ind=TRUE))
        if (gammas[ngridTh] > Min2) { #Min2 in gammas, i.e. the middle
              # regime contains at least
              # Ttrim elements; if help=0, gammas[i] is so large that the middle
              # regime contains less than Trim elements, which means that no LM
              # and SSR values should be computed for this value of i
            jMin<-min(which(gammas >= Min2,arr.ind=TRUE))
                #The index of the least element >= Min2 in gammas
            for (j in jMin:ngridTh) {
                d3<- ifelse(w0>gammas[j],1,0)  #d1: dummy variable
                z3<-if(common=="All") c(d3)*X else c(d3)*w0
                z<-cbind(z1,z3)
                res_unrestr <-if(dir) qr.resid(q, z) else z-z0zz%*%(t(X)%*%z)
                    #z11: residuals from unrestricted model (with threhsold)
                zea<-res_restr1*res_unrestr
                zeb<-res_restr2*res_unrestr
                ze<-cbind(zea,zeb)
      # [(z11.*(e1*ones(1,length(z11(1,:))))),(z11.*(e2*ones(1,length(z11(1,:)))))];
                v<-crossprod(ze)
                z11y<-crossprod(res_unrestr,y)
                s<-matrix(c(z11y), ncol=1)
                    #vectorization of the parameter matrix z11y
                Vinv<-try(solve(v), silent=TRUE)
                if(inherits(Vinv, "try-error")) Vinv<-ginv(v)
                store[i,j]<-t(s)%*%Vinv%*%s
            } # end for (j in jMin:ngridTh) {
        } #end if (help>0) {
    } #end of the whole loop
  lm01<-max(store, na.rm=TRUE)
  return(lm01)
} #end of the function lmtest3Reg_boot

gamma2<-GridSetup(w0.ord)
lm01<-if (type=="3Reg") lmtest3Reg(y,x,w0,gamma2, dir=dir) else
    lmtest2RegSymReg(y,x,w0,gamma2, dir=dir)
teststat<-max(lm01$store, na.rm=TRUE)
###################################
### Bootstraps
###################################
if(nboot==0){
  CriticalValBoot<-NULL
  PvalBoot<-NULL
  boots.reps<-NULL
  if(hpc=="foreach") warning("hpc='foreach' used only when nboot>0\n")
}else if (nboot>0){
###################################
### Fixed Regressor Bootstrap %
###################################
  if(boot.type=="FixedReg"){
    X<-cbind(w0,x)  #X: ECT and intercept and lags
    Ttrim<-trim*t
    if(dir){
      q<-qr(X)
    } else{
      zOzz<-X%*%solve(t(X)%*%X)
    }

    lmtest_withBoot<-function(e){
      yr<-rnorm(n=t,0,1)*e
      return(if (type=="3Reg") lmtest3Reg_boot(yr,x,w0,gamma2,dir=dir) else
          lmtest2RegSymReg_boot(yr,x,w0,gamma2,dir=dir))
    }
    boots.reps<- if(hpc=="none") replicate(nboot, lmtest_withBoot(e=residuals(ve)))
        else foreach(i=1:nboot, .export="lmtest_withBoot", .combine="c")
        %dopar% lmtest_withBoot(e=residuals(ve))
###################################
### Residual Bootstrap
###################################
  } else{
      lmtest_with_resBoot<-function(ve){
      #bootstrap it
        data.boot<-TVECM.sim(TVECMobject=ve, type="boot")
      # estimate VECM
        if(is.null(fixed.beta)){
          ve.boot<-VECM(data.boot, lag=lag, include="const", estim="ML")
        }else{
          ve.boot<-VECM(data.boot, lag=lag, include="const",
               beta=fixed.beta, estim="2OLS")
        }
      # extract w0, y and x
        ect.boot<-ve.boot$model[,"ECT"]
        which.nas<-1:(p+1)
        w0.boot<-matrix(ect.boot[-which.nas], ncol=1)
        x.boot<-ve.boot$model[-which.nas,-c(1:3)]
        y.boot<-ve.boot$model[,c(1:2)]
        y.boot<-diff(y.boot)[(p+1):(T-1),]
      # set-up grid
        w0.ord.boot<-sort(w0.boot)
        gamma2.boot<-GridSetup(w0.ord.boot)
        return(if (type=="3Reg") lmtest3Reg_boot(y.boot,x.boot,w0.boot,
            gamma2.boot, dir=dir) else
               lmtest2RegSymReg_boot(y.boot,x.boot,w0.boot,gamma2.boot,dir=dir))
    }
      boots.reps<- if(hpc=="none") replicate(nboot, lmtest_with_resBoot(ve)) else
      foreach(i=1:nboot,.export="lmtest_with_resBoot", .combine="c")
      %dopar% lmtest_with_resBoot
  }#end if boot= ResBoot


## result: compute p values and critical values:
   PvalBoot<-mean(ifelse(boots.reps>teststat,1,0))
   CriticalValBoot<-quantile(boots.reps, probs= c(0.9, 0.95,0.99))
}#end if boot>0
####### Return args
args<-list()
args$nboot<-nboot
args$boot.type<-boot.type
ret<-list()
ret$args<-args
ret$stat<-teststat
ret$SSR<-min(lm01$SSR,na.rm=TRUE)
ret$LMvalues<-lm01
ret$ths<-gamma2
if (type=="3Reg") {
    a<-which(abs(lm01$store-ret$stat)<1e-10,arr.ind=TRUE)
    ret$maxThLM<-c(gamma2[a[1,1]],gamma2[a[1,2]])
    d1<- ifelse(w0<=ret$maxThLM[1],1,0)
    d3<- ifelse(w0>ret$maxThLM[2],1,0)
    d2<-1-d1-d3
    ret$PercentsLM<-c(round(mean(d1)*100,digits=1),round(mean(d2)*100,digits=1),
        round(mean(d3)*100,digits=1))
    a<-which(abs(lm01$SSR-ret$SSR)<1e-10,arr.ind=TRUE)
    ret$minThSSR<-c(gamma2[a[1,1]],gamma2[a[1,2]])
    d1<- ifelse(w0<=ret$minThSSR[1],1,0)
    d3<- ifelse(w0>ret$minThSSR[2],1,0)
    d2<-1-d1-d3
    ret$PercentsSSR<-c(round(mean(d1)*100,digits=1),round(mean(d2)*100,digits=1),
        round(mean(d3)*100,digits=1))
} else {
    ret$maxThLM<-gamma2[which(abs(lm01$store-ret$stat)<1e-10)]
    d1<- ifelse(w0<=ret$maxThLM,1,0)
    d2<-1-d1
    ret$PercentsLM<-c(round(mean(d1)*100,digits=1),round(mean(d2)*100,digits=1))
    ret$minThSSR<-gamma2[which(abs(lm01$SSR-ret$SSR)<1e-10)]
    d1<- ifelse(w0<=ret$minThSSR,1,0)
    d2<-1-d1
    ret$PercentsSSR<-c(round(mean(d1)*100,digits=1),round(mean(d2)*100,digits=1))
}
ret$PvalBoot<-PvalBoot
ret$CriticalValBoot<-CriticalValBoot
ret$allBoots<-boots.reps
class(ret)<-"TVECMXHanSeo02Test"
return(ret)
} #End of the whole function

### Print method
print.TVECMXHanSeo02Test<-function(x,...){
  cat("## Test of linear versus threshold cointegration",
      " of Hansen and Seo (2002) ##\n\n", sep="")
  cat("Test Statistic:\t", x$stat)
  cat("\t(Maximized for threshold value:", x$maxThLM, ")\n")
  cat("Percentage of observations in each regime", x$PercentsLM, "%\n\n")
  cat("Minimum SSR:\t", x$SSR)
  cat("\t(Minimized for threshold value:", x$minThSSR, ")\n")
  cat("Percentage of observations in each regime", x$PercentsSSR,"%\n")
  if(x$args$nboot>0){
      boot.name<-switch(x$args$boot.type,
          "FixedReg"="Fixed regressor bootstrap", "ResBoot"="Residual Bootstrap")
      cat("P-Value:\t", x$PvalBoot, "\t\t(",boot.name, ")\n")
    }
}
```

Appendix C

The R code chunks used in Chapter 4 and 5

This apppendix contains all the R code chunks used in the data analysis of Chapter 4 and Chapter 5. In this analysis we have used functions from the R base distribution and the following contributed R packages: **tsDyn** (Stigler 2011), **urca** (Pfaff 2008a), **vars** (Pfaff 2008b), **xtable** (Swinton 2011), **asbio** (Aho 2011) and **zoo** (Zeileis and Grothendieck 2005), which may be downloaded from `http://cran.r-project.org/`.

```
## chunk number 1: setup
setCacheDir("sweave-cache3/values")
options(tikzMetricsDictionary="tikzMetricsDictionary",keep.space=TRUE,
    keep.blank.line=FALSE,width=80,tikzMetricPackages = c("\\usepackage[T1]{fontenc}",
    "\\usetikzlibrary{calc}","\\usepackage{amssymb}","\\usepackage{amsbsy}"))
library(urca)
library(zoo)
library(vars)
library(tsDyn,lib.loc="C:/R/R-2.12.0/testlibrary")
library(xtable)
xcolours<-c("black","red","green","blue","orange","brown","violet",
    "turquoise","pink","magenta","yellow","lightblue")
## chunk number 2: importing.data
renter<-read.table("renter_mnd.sdv",sep=";",dec=",",header=TRUE)
renter<-renter[!is.na(renter$NIBOR.T.N.nom),]
renter<-renter[!is.na(renter$NIBOR.12M.nom),]
NIBTN<-as.zoo(ts(renter$NIBOR.T.N.nom,frequency=12,start=c(1985,5),end=c(2010,12)))
NIB12M<-as.zoo(ts(renter$NIBOR.12M.nom,frequency=12,start=c(1985,5),end=c(2010,12)))
## chunk number 3: fig.NIB12MTN
par(mfrow=c(2,1))
plot(NIB12M,type='p',main='Plot of NIB12M')
plot(NIBTN,type='p',main='Plot of NIBTN')
## chunk number 4: make.TestNIB12MTN
TestNIBTN<-NIBTN
TestNIB12M<-NIB12M
OutliersNIBTN<-NIBTN[c(13,20,21,89,90,92)]
TestNIBTN[13]<-14.205  #interpolation between 14.01 in Apr. 86 and 14.4 in Jun. 86
```

```
TestNIBTN[20]<-14.42  #interpolation between 14.33 in Nov. 86 and 14.6 in Feb. 87
TestNIBTN[21]<-14.51  #interpolation between 14.33 in Nov. 86and 14.6 in Feb. 87
TestNIBTN[89]<-11.00  #interpolation between 11.08 in Aug. 92 and 10.84 in Nov. 92
TestNIBTN[90]<-10.92  #interpolation between 11.08 in Aug. 92 and 10.84 in Nov. 92
TestNIBTN[92]<-11.165  #interpolation between 10.84 in Nov. 92 and 11.49 in Jan. 93
TestNIBTN[c(13,20,21,89,90,92)] #The new values in TestNIBTN:
OutliersNIB12M<-NIB12M[c(20,21)]
TestNIB12M[20]<-14.89  #interpolation between 14.78 in Nov. 86 and 15.12 in Feb. 87
TestNIB12M[21]<-15.0  #interpolation between 14.78 in Nov. 86 and 15.12 in Feb. 87
TestNIB12M[c(20,21)] #The new values in TestNIB12M:
## chunk number 5: fig.TestNIB12MTN
par(mfrow=c(2,1))
plot(TestNIB12M,type='p',main='Plot of TestNIB12M')
plot(TestNIBTN,type='p',main='Plot of TestNIBTN')
## chunk number 6: compute.VAR.tsDyn
NIB12MTN <- cbind(NIB12M,NIBTN)
TestNIB12MTN <- cbind(TestNIB12M,TestNIBTN)
NIB12MTN.lVAR<-lineVar(NIB12MTN,lag=3,include="const",model="VAR",I="level",beta=NULL,
        estim=c("2OLS", "ML"),LRinclude=c("none", "const", "trend","both"))
summary(NIB12MTN.lVAR)
NIB12MTN.VAR.summary<-summary(NIB12MTN.lVAR)
NIB12MTN.VAR.tab<-t(NIB12MTN.VAR.summary$bigcoefficients)
tab.NIB12MTN.VAR<-xtable(NIB12MTN.VAR.tab,digits = 3,align="|l|l|l|")
## chunk number 7: tab.NIB12MTN.var.coeff
print(tab.NIB12MTN.VAR,floating=FALSE,colnames=TRUE,rownames=TRUE,
    sanitize.text.function = function(x){x})
## chunk number 8: diag.test.NIB12MTN.var
NIB12MTN.VAR<-VAR(NIB12MTN,type="const",lag.max=10,ic="SC")
NIB12MTN.VAR.summary<-summary(NIB12MTN.VAR)
NIB12MTN.VAR.arch<-arch.test(NIB12MTN.VAR,multivariate.only=FALSE)
NIB12MTN.VAR.normality<-normality.test(NIB12MTN.VAR,multivariate.only=FALSE)
NIB12MTN.VAR.serial<-serial.test(NIB12MTN.VAR)
NIB12MTN.VAR2<-VAR(NIB12MTN,p=2,type="none")
NIB12MTN.VAR2.serial<-serial.test(NIB12MTN.VAR2)
NIB12MTN.VAR.ArchNormality<-rbind(c(NIB12MTN.VAR.arch$arch.mul$statistic,
    NIB12MTN.VAR.arch$arch.mul$parameter,NIB12MTN.VAR.arch$arch.mul$p.value),
    c(NIB12MTN.VAR.arch$arch.uni$NIB12M$statistic,
        NIB12MTN.VAR.arch$arch.uni$NIB12M$parameter,
        NIB12MTN.VAR.arch$arch.uni$NIB12M$p.value),
    c(NIB12MTN.VAR.arch$arch.uni$NIBTN$statistic,
        NIB12MTN.VAR.arch$arch.uni$NIBTN$parameter,
        NIB12MTN.VAR.arch$arch.uni$NIBTN$p.value),
    c(NIB12MTN.VAR.normality$jb.mul$JB$statistic,
        NIB12MTN.VAR.normality$jb.mul$JB$parameter,
        NIB12MTN.VAR.normality$jb.mul$JB$p.value),
    c(NIB12MTN.VAR.normality$jb.mul$Skewness$statistic,
        NIB12MTN.VAR.normality$jb.mul$Skewness$parameter,
        NIB12MTN.VAR.normality$jb.mul$Skewness$p.value),
    c(NIB12MTN.VAR.normality$jb.mul$Kurtosis$statistic,
        NIB12MTN.VAR.normality$jb.mul$Kurtosis$parameter,
        NIB12MTN.VAR.normality$jb.mul$Kurtosis$p.value),
    c(NIB12MTN.VAR.normality$jb.uni$NIB12M$statistic,
        NIB12MTN.VAR.normality$jb.uni$NIB12M$parameter,
        NIB12MTN.VAR.normality$jb.uni$NIB12M$p.value),
    c(NIB12MTN.VAR.normality$jb.uni$NIBTN$statistic,
        NIB12MTN.VAR.normality$jb.uni$NIBTN$parameter,
        NIB12MTN.VAR.normality$jb.uni$NIBTN$p.value))
rownames(NIB12MTN.VAR.ArchNormality)<-c("Multivariate ARCH-LM test",
    "ARCH-LM test of NIB12M","ARCH-LM test of NIBTN","Multivariate JB test",
    "Multivariate Skewness test","Multivariate Kurtosis test","JB test of NIB12M",
    "JB test of NIBTN")
colnames(NIB12MTN.VAR.ArchNormality)<-c("Statistic","df","p-value")
NIB12MTN.VAR.SerialTests<-rbind(c(NIB12MTN.VAR2.serial$serial$statistic,
    NIB12MTN.VAR2.serial$serial$parameter,NIB12MTN.VAR2.serial$serial$p.value),
    c(NIB12MTN.VAR.serial$serial$statistic,NIB12MTN.VAR.serial$serial$parameter,
    NIB12MTN.VAR.serial$serial$p.value))
rownames(NIB12MTN.VAR.SerialTests)<-c("\\code{K=2}","\\code{K=3}")
colnames(NIB12MTN.VAR.SerialTests)<-c("Statistic","df","p-value")
## chunk number 9: tab.NIB12MTN.VAR.ArchNormality
print(xtable(NIB12MTN.VAR.ArchNormality,digits = c(0,1,0,3),align="|l|r|r|r|",
    display=c("f","f","f","g")),floating=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 10: tab.NIB12MTN.VAR.SerialTests
print(xtable(NIB12MTN.VAR.SerialTests,digits = c(0,1,0,3),align="|l|r|r|r|",
    display=c("f","f","f","g")),floating=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 11: ADF-NIBTN.ct
NIBTN.ct<-ur.df(NIBTN,lags=10,selectlags="AIC",type='trend')
summary(NIBTN.ct)
NIBTN.ct.tab<-cbind(t(as.matrix(NIBTN.ct@teststat)),as.matrix(NIBTN.ct@cval))
rownames(NIBTN.ct.tab)<-c("$\\tau_3$","$\\phi_2$","$\\phi_3$")
colnames(NIBTN.ct.tab)<-c("statistic","1\\%","5\\%","10\\%")
## chunk number 12: tab.NIBTN.ct
print(xtable(NIBTN.ct.tab,digits = c(2,2,2,2,2)),floating=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 13: fig.NIBTN.ct
plot(NIBTN.ct)
## chunk number 14: ADF-NIBTN.co
NIBTN.co<-ur.df(NIBTN,lags=10,selectlags="AIC",type='drift')
summary(NIBTN.co)
NIBTN.co.tab<-cbind(t(as.matrix(NIBTN.co@teststat)),as.matrix(NIBTN.co@cval))
rownames(NIBTN.co.tab)<-c("$\\tau_2$","$\\phi_1$")
colnames(NIBTN.co.tab)<-c("statistic","1\\%","5\\%","10\\%")
## chunk number 15: tab.NIBTN.co
print(xtable(NIBTN.co.tab,digits = c(2, 2, 2, 2, 2)),floating=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 16: ADF-diffNIBTN.ct
diffNIBTN<-diff(NIBTN)
diffNIBTN.ct<-ur.df(diffNIBTN,lags=10,selectlags="AIC",type='trend')
summary(diffNIBTN.ct)
diffNIBTN.ct.tab<-cbind(t(as.matrix(diffNIBTN.ct@teststat)),
    as.matrix(diffNIBTN.ct@cval))
```

```
rownames(diffNIBTN.ct.tab)<-c("$\\tau_3$","$\\phi_2$","$\\phi_3$")
colnames(diffNIBTN.ct.tab)<-c("statistic","1\\%","5\\%","10\\%")
## chunk number 17: tab.diffNIBTN.ct
print(xtable(diffNIBTN.ct.tab,digits = c(2, 2, 2, 2, 2)),floating=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 18: KPSS-NIBTN
NIBTN.kpss<-ur.kpss(NIBTN,type="mu",lags="long")
summary(NIBTN.kpss)
NIBTN.kpss.tau<-ur.kpss(NIBTN,type="tau",lags="long")
summary(NIBTN.kpss.tau)
NIBTN.kpss.tab<-rbind(cbind(t(as.matrix(NIBTN.kpss@teststat)),
    as.matrix(NIBTN.kpss@cval)),cbind(t(as.matrix(NIBTN.kpss.tau@teststat)),
    as.matrix(NIBTN.kpss.tau@cval)))
rownames(NIBTN.kpss.tab)<-c("$\\hat{\\eta}_\\mu$","$\\hat{\\eta}_\\tau$")
colnames(NIBTN.kpss.tab)<-c("statistic","10\\%","5\\%","2.5\\%","1\\%")
## chunk number 19: tab.NIBTN.kpss
print(xtable(NIBTN.kpss.tab,digits = c(2, 2, 2, 2, 2, 2)),floating=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 20: ADF-NIB12M.ct
NIB12M.ct<-ur.df(NIB12M,lags=10,selectlags="AIC",type='trend')
summary(NIB12M.ct)
NIB12M.ct.tab<-cbind(t(as.matrix(NIB12M.ct@teststat)),as.matrix(NIB12M.ct@cval))
rownames(NIB12M.ct.tab)<-c("$\\tau_3$","$\\phi_2$","$\\phi_3$")
colnames(NIB12M.ct.tab)<-c("statistic","1\\%","5\\%","10\\%")
## chunk number 21: tab.NIB12M.ct
print(xtable(NIB12M.ct.tab,digits = c(2, 2, 2, 2, 2)),floating=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 22: fig.NIB12M.ct
plot(NIB12M.ct)
## chunk number 23: ADF-NIB12M.co
NIB12M.co<-ur.df(NIB12M,lags=10,selectlags="AIC",type='drift')
summary(NIB12M.co)
NIB12M.co.tab<-cbind(t(as.matrix(NIB12M.co@teststat)),as.matrix(NIB12M.co@cval))
rownames(NIB12M.co.tab)<-c("$\\tau_2$","$\\phi_1$")
colnames(NIB12M.co.tab)<-c("statistic","1\\%","5\\%","10\\%")
## chunk number 24: tab.NIB12M.co
print(xtable(NIB12M.co.tab,caption = str,digits = c(2,2,2,2,2)),
    floating=FALSE,sanitize.text.function = function(x){x})
## chunk number 25: ADF-diffNIB12M.ct
diffNIB12M<-diff(NIB12M)
diffNIB12M.ct<-ur.df(diffNIB12M,lags=10,selectlags="AIC",type='trend')
summary(diffNIB12M.ct)
diffNIB12M.ct.tab<-cbind(t(as.matrix(diffNIB12M.ct@teststat)),
    as.matrix(diffNIB12M.ct@cval))
rownames(diffNIB12M.ct.tab)<-c("$\\tau_3$","$\\phi_2$","$\\phi_3$")
colnames(diffNIB12M.ct.tab)<-c("statistic","1\\%","5\\%","10\\%")
## chunk number 26: tab.diffNIB12M.ct
print(xtable(diffNIB12M.ct.tab,digits = c(2,2,2,2,2)),floating=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 27: KPSS-NIB12M
NIB12M.kpss<-ur.kpss(NIB12M,type="mu",lags="long")
```

```
summary(NIB12M.kpss)
NIB12M.kpss.tau<-ur.kpss(NIB12M,type="tau",lags="long")
summary(NIB12M.kpss.tau)
NIB12M.kpss.tab<-rbind(cbind(t(as.matrix(NIB12M.kpss@teststat)),
    as.matrix(NIB12M.kpss@cval)),cbind(t(as.matrix(NIB12M.kpss.tau@teststat)),
    as.matrix(NIB12M.kpss.tau@cval)))
rownames(NIB12M.kpss.tab)<-c("$\\hat{\\eta}_\\mu$","$\\hat{\\eta}_\\tau$")
colnames(NIB12M.kpss.tab)<-c("statistic","10\\%","5\\%","2.5\\%","1\\%")
## chunk number 28: tab.NIB12M.kpss
print(xtable(NIB12M.kpss.tab,digits = c(2,2,2,2,2,2)),floating=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 29: cajo-NIB12MTN
NIB12MTN.trace <- ca.jo(NIB12MTN, type = 'trace', K = 3, ecdet="none")
summary(NIB12MTN.trace)
NIB12MTN.eigen <- ca.jo(NIB12MTN, type = 'eigen', K = 3, ecdet="none")
summary(NIB12MTN.eigen)
NIB12MTN.trace.tab<-cbind(as.matrix(NIB12MTN.trace@teststat),
    as.matrix(NIB12MTN.trace@cval))
NIB12MTN.eigen.tab<-cbind(as.matrix(NIB12MTN.eigen@teststat),
    as.matrix(NIB12MTN.eigen@cval))
colnames(NIB12MTN.trace.tab)<-c("statistic","10\\%","5\\%","1\\%")
colnames(NIB12MTN.eigen.tab)<-c("statistic","10\\%","5\\%","1\\%")
## chunk number 30: tab.NIB12MTN.trace
print(xtable(NIB12MTN.trace.tab,digits = c(2,2,2,2,2)),floating=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 31: tab.NIB12MTN.eigen
print(xtable(NIB12MTN.eigen.tab,digits = c(2,2,2,2,2)),floating=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 32: fig.coint.relation
y1<-NIB12MTN%*%as.matrix(NIB12MTN.trace@V[1:2,1],nrow=2,ncol=1)
y2<-NIB12MTN%*%as.matrix(NIB12MTN.trace@V[1:2,2],nrow=2,ncol=1)
r1<-NIB12MTN.trace@RK%*%NIB12MTN.trace@V[,1]
r2<-NIB12MTN.trace@RK%*%NIB12MTN.trace@V[,2]
par(mfrow=c(2,2))
plot(y1,type='l',main='Plot of $\\boldsymbol{\\beta}_{1}y$')
plot(r1,type='l',main='Plot of $\\boldsymbol{\\beta}_{1}R_1$')
plot(y2,type='l',main='Plot of $\\boldsymbol{\\beta}_{2}y$')
plot(r2,type='l',main='Plot of $\\boldsymbol{\\beta}_{2}R_1$')
## chunk number 33: compute.vecm.tsDyn
NIB12MTN.vecm<-VECM(NIB12MTN, lag=2,r=1, include = "const",
    beta=NULL,estim="ML",LRinclude="none")
NIB12MTN.vecm.summary<-summary(NIB12MTN.vecm)
NIB12MTN.vecm.tab<-rbind(t(NIB12MTN.vecm.summary$bigcoefficients),
    t(round(NIB12MTN.vecm$model.specific$coint,digits=4)))
rownames(NIB12MTN.vecm.tab)[7]<-"Cointegration relation"
tab.NIB12MTN.vecm<-xtable(NIB12MTN.vecm.tab,digits = 3,align="|l|l|l|")
## chunk number 34: compute.vecm.urca eval=FALSE
## NIB12MTN.eigen<-ca.jo(NIB12MTN,type="eigen",K=3,spec= "transitory",
##     ecdet="none")
## NIB12MTN.vecm<-cajorls(NIB12MTN.eigen,r=1)
## NIB12MTN.vecm
## NIB12MTN.vecm.tab<-rbind(NIB12MTN.vecm$rlm$coefficients,
##     t(NIB12MTN.vecm$beta[,1]))
## rownames(NIB12MTN.vecm.tab)[7]<-"Cointegration relation"
## tab.NIB12MTN.vecm<-xtable(NIB12MTN.vecm.tab,digits = 3,align="|l|l|l|")
## NIB12MTN.vecm$residuals<-NIB12MTN.vecm$rlm$residuals
## chunk number 35: tab.NIB12MTN.vecm.coeff
print(tab.NIB12MTN.vecm,floating=FALSE,colnames=TRUE,rownames=TRUE,
    sanitize.text.function = function(x){x})
## chunk number 36: TVECM.model
pdf(file="thesis3-TVECM-GridSearch.pdf", onefile=TRUE)
NIB12MTN.tvecm<-TVECM(NIB12MTN,nthresh=1,lag=2,ngridBeta=400,ngridTh=400,
    beta=list(int=c(0.85,1.25)),th1=list(int=c(-1.5,1.5)),plot=TRUE,
    trim=0.1,common="All",trace=FALSE)
TestNIB12MTN.tvecm<-TVECM(TestNIB12MTN,nthresh=1,lag=2,ngridBeta=400,
    ngridTh=400,beta=list(int=c(0.85,1.25)),th1=list(int=c(-1.0,1.25)),
    plot=FALSE,trim=0.1,common="All",trace=FALSE)
dev.off(which = dev.cur())
NIB12MTN.tvecm.summary<-summary(NIB12MTN.tvecm)
A<-t(rbind(NIB12MTN.tvecm.summary$bigcoefficients$Bdown,
    NIB12MTN.tvecm.summary$bigcoefficients$Bup))
B<-rbind(colnames(A),A)
B<-cbind(rownames(B),B)
tvecm.model.coeff<-xtable(B,align=c("|l|l|l|l|l|l|"))
## chunk number 37: fig.diffSeries
diffR<-diff(NIB12M)
diffr<-diff(NIBTN)
Spread<-NIB12M-NIBTN
diffSpread<-diff(Spread)
SpreadMinus1<-Spread[2:length(Spread)]
par(mfrow=c(3,1))
plot(SpreadMinus1,diffR,type="l",ylab="$\\Delta R_t$",
    xlab="$s_{t-1}=R_{t-1}-r_{t-1}$")
abline(v=NIB12MTN.tvecm$model.specific$Thresh,col="red")
plot(SpreadMinus1,diffr,type="l",ylab="$\\Delta r_t$",
    xlab="$s_{t-1}=R_{t-1}-r_{t-1}$")
abline(v=NIB12MTN.tvecm$model.specific$Thresh,col="red")
plot(SpreadMinus1,diffSpread,type="l",ylab="$\\Delta s_t$",
    xlab="$s_{t-1}=R_{t-1}-r_{t-1}$")
abline(v=NIB12MTN.tvecm$model.specific$Thresh,col="red")
## chunk number 38: make.table.TVECM.model
make.table.TVECM<-function() {
    Result<-numeric(0)
    for (i in 1:2) {
        for (j in 1:3) {
            x.tvecm<-TVECM(NIB12MTN,nthresh=i,lag=j,
                ngridBeta=0,ngridTh=400,beta=list(exact=1),
                th1=list(int=c(-1.5,1.5)),plot=FALSE,trim=0.1,
                common="All",trace=FALSE)
            Tsum<-summary(x.tvecm)
            if (length(Tsum$nobs_regime)<2.5) {
                nobsRegime<-c(round(Tsum$nobs_regime[1],digits=3),NA,
                round(Tsum$nobs_regime[2],digits=3))
            } else {
                nobsRegime<-round(Tsum$nobs_regime,digits=3)
            }
            Result<-rbind(Result,c(i,j,1,Tsum$nparB,round(Tsum$aic,digits=1),
                round(Tsum$bic,digits=1),round(Tsum$SSR,digits=1),
                nobsRegime))
            x.tvecm<-TVECM(NIB12MTN,nthresh=i,lag=j,ngridBeta=400,
                ngridTh=400,plot=FALSE,trim=0.1,beta=list(int=c(0.85,1.25)),
                th1=list(int=c(-1.5,1.5)),common="All",trace=FALSE)
            Tsum<-summary(x.tvecm)
            if (length(Tsum$nobs_regime)<2.5) {
                nobsRegime<-c(round(Tsum$nobs_regime[1],digits=3),NA,
                    round(Tsum$nobs_regime[2],digits=3))
            } else {
                nobsRegime<-round(Tsum$nobs_regime,digits=3)
            }
            Result<-rbind(Result,c(i,j,
                round(Tsum$model.specific$beta,digits=3),Tsum$nparB,
                round(Tsum$aic,digits=1),round(Tsum$bic,digits=1),
                round(Tsum$SSR,digits=1),nobsRegime))
        }
    }
    colnames(Result)<-c("nthresh","lag","$\\beta$","Parameters","AIC","BIC",
        "SSR","ndown","nmiddle","nup")
    return(Result)
}
Result.tvecm<-make.table.TVECM()
## chunk number 39: tab.Result.tvecm
print(xtable(Result.tvecm,digits=c(0,0,0,3,0,1,1,1,3,3,3)),
    include.rownames=FALSE,floating=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 40: Residuals.TVECM.model
ArchLM.test<-function(y,q=-1) {
    T<-dim(y)[1]
    K<-dim(y)[2]
    if (q<0) {
        if (K>1) {
            q<-5
        } else if (K==1) {
            q<-16
        }
    }
    Res<-matrix(0,nrow=T,ncol=K*(K+1)/2)
    for (i in 1:T) {
        help<-y[i,] %*% t(y[i,])
        Res[i,]<-help[lower.tri(help,diag=TRUE)]
    }
    Res.Varq<-VAR(Res,p=q,type="const")
    Residq<-cbind(Res.Varq$varresult$y1$residuals,Res.Varq$varresult$y2$residuals,
        Res.Varq$varresult$y3$residuals)
    Sigmaq<-(1/(T-q)) * t(Residq) %*% Residq
    Res.Var0<-lm(Res ~1)
    Resid0<-Res.Var0$residuals
    Sigma0<-(1/T) * t(Resid0) %*% Resid0
    InvSigma0<-solve(Sigma0)
    ArchLM<-T*K*(K+1)/2-T*sum(diag(Sigmaq%*%InvSigma0))
    df<-q*K*K*(K+1)*(K+1)/4
    Pval=pchisq(ArchLM,df,lower.tail=FALSE)
    return(list(ArchLM=ArchLM,df=df,Pval=Pval))
}
NIB12MTN.VAR.resid<-cbind(NIB12MTN.VAR$varresult$NIB12M$residuals,
    NIB12MTN.VAR$varresult$NIBTN$residuals)
ArchLM.test.VAR<-ArchLM.test(NIB12MTN.VAR.resid)
ArchLM.test.tvecm<-ArchLM.test(NIB12MTN.tvecm$residuals)
ARCHtest.tab<-c(ArchLM.test.VAR$ArchLM,ArchLM.test.VAR$df,ArchLM.test.VAR$Pval)
ARCHtest.tab<-rbind(ARCHtest.tab,
    c(ArchLM.test.tvecm$ArchLM,ArchLM.test.tvecm$df,ArchLM.test.tvecm$Pval))
rownames(ARCHtest.tab)<-c("VAR model","TVECM")
colnames(ARCHtest.tab)<-c("Test statistic","df","P-value")
library(asbio)
NIB12MTN.VAR.resid.DH<-DH.test(NIB12MTN.VAR.resid,c("NIB12M","NIBTN"))
NIB12MTN.VAR.DH.tab<-rbind(NIB12MTN.VAR.resid.DH$multi,NIB12MTN.VAR.resid.DH$univ)
dimnames(NIB12MTN.VAR.DH.tab)[[1]][1]<-"Multivariate"
NIB12MTN.tvecm.resid.DH<-DH.test(NIB12MTN.tvecm$residuals,c("NIB12M","NIBTN"))
NIB12MTN.tvecm.DH.tab<-rbind(NIB12MTN.tvecm.resid.DH$multi,NIB12MTN.tvecm.resid.DH$univ)
dimnames(NIB12MTN.tvecm.DH.tab)[[1]][1]<-"Multivariate"
NIB12MTN.DH.tab<-cbind(NIB12MTN.VAR.DH.tab,NIB12MTN.tvecm.DH.tab)
## chunk number 41: tab.tvecm.model
print(tvecm.model.coeff,hline.after=c(1,nrow(tvecm.model.coeff)),
    floating=FALSE,quote=FALSE,include.colnames=FALSE,include.rownames=FALSE,
    add.to.row=list(pos=list(0,1),
    command=c("\\hline & \\multicolumn{2}{|l|}{\\textbf{Lower regime}} &
        \\multicolumn{2}{|l|}{\\textbf{Upper regime}}\\\\[1mm]\\cline{2-5}","")),
    sanitize.text.function = function(x){x})
## chunk number 42: tab.ARCHtest.Residuals
print(xtable(ARCHtest.tab,digits = c(2,2,0,5),align=c("|r|r|r|r|")),floating=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 43: tab.DHtest.Residuals
NIB12MTN.DH<-xtable(NIB12MTN.DH.tab,digits=c(2,2,0,5,2,0,5),align=c("|l|r|r|r|r|r|r|"))
print(NIB12MTN.DH,hline.after=c(0,1,nrow(NIB12MTN.DH)),
    floating=FALSE,quote=FALSE,include.colnames=TRUE,include.rownames=TRUE,
    add.to.row=list(pos=list(-1,1),
    command=c("\\hline & \\multicolumn{3}{|l|}{\\textbf{VAR model}} &
        \\multicolumn{3}{|l|}{\\textbf{TVECM}}\\\\[1mm]\\cline{2-7}","")),
    sanitize.text.function = function(x){x})
#print(xtable(NIB12MTN.DH.tab),floating=FALSE,rownames=TRUE,colnames=TRUE,
#    sanitize.text.function = function(x){x})
## chunk number 44: responses.TVECM1
w<-NIB12MTN%*%NIB12MTN.tvecm$model.specific$coint
m<-2
w<-w[(m+2):length(w)]
```

```
Thresh<-NIB12MTN.tvecm$model.specific$Thresh
Bdown<-NIB12MTN.tvecm$coefficients$Bdown
Bup<-NIB12MTN.tvecm$coefficients$Bup
Res1R<-ifelse(w<=Thresh,Bdown[1,2]+Bdown[1,1]*w,Bup[1,2]+Bup[1,1]*w)
Res1r<-ifelse(w<=Thresh,Bdown[2,2]+Bdown[2,1]*w,Bup[2,2]+Bup[2,1]*w)
s1<-cbind(Res1R,Res1r)%*%NIB12MTN.tvecm$model.specific$coint
Ind<-order(w)
## chunk number 45: fig.responses.TVECM1
par(mfrow=c(1,1))
plot(w[Ind],Res1r[Ind],type="l",lty=1,col=xcolours[1],xlab="$s_{t-1}$",
     ylab="Response",xlim=c(-5,max(w)),ylim=c(-5,5))
axis(4)
abline(0,0,col="blue")
lines(w[Ind],Res1R[Ind],type="l",lty=1,col=xcolours[2])
lines(w[Ind],s1[Ind],type="l",lty=1,col=xcolours[3])
legend("topright",c("$\\Delta r_t$","$\\Delta R_t$","$\\Delta s_t$"),
    lty=1,col=xcolours[1:3])
## chunk number 46: responses.TVECM2
m<-2
NIB12MTN.tvecm2<-TVECM(NIB12MTN,nthresh=2,lag=m,ngridBeta=0,ngridTh=200,
    beta=list(exact=NIB12MTN.tvecm$model.specific$beta),plot=FALSE,
    trim=0.1,common="All",trace=FALSE)
w<-NIB12MTN%*%NIB12MTN.tvecm2$model.specific$coint
w<-w[(m+2):length(w)]
Thresh<-NIB12MTN.tvecm2$model.specific$Thresh
Bdown<-NIB12MTN.tvecm2$coefficients$Bdown
Bmiddle<-NIB12MTN.tvecm2$coefficients$Bmiddle
Bup<-NIB12MTN.tvecm2$coefficients$Bup
Res2R<-ifelse(w<=Thresh[1],Bdown[1,2]+Bdown[1,1]*w,
    ifelse(w<=Thresh[2],Bmiddle[1,2]+Bmiddle[1,1]*w,Bup[1,2]+Bup[1,1]*w))
Res2r<-ifelse(w<=Thresh[1],Bdown[2,2]+Bdown[2,1]*w,
    ifelse(w<=Thresh[2],Bmiddle[2,2]+Bmiddle[2,1]*w,Bup[2,2]+Bup[2,1]*w))
s2<-cbind(Res2R,Res2r)%*%NIB12MTN.tvecm$model.specific$coint
Ind<-order(w)
## chunk number 47: fig.responses.TVECM2
par(mfrow=c(1,1))
plot(w[Ind],Res2r[Ind],type="l",lty=1,col=xcolours[1],xlim=c(-5,max(w)),
     xlab="$s_{t-1}$",ylab="Response",ylim=c(-5,5))
axis(4)
abline(0,0,col="blue")
lines(w[Ind],Res2R[Ind],type="l",lty=1,col=xcolours[2])
lines(w[Ind],s2[Ind],type="l",lty=1,col=xcolours[3])
legend("topright",c("$\\Delta r_t$","$\\Delta R_t$","$\\Delta s_t$"),
    lty=1,col=xcolours[1:3])
## chunk number 48: Han-Seo-Test
TimeUsed.NIB12MTN.Fix.3Reg<-system.time(NIB12MTN.HSTest.FixedReg.3Reg<-
    TVECM.XHStest(NIB12MTN,nboot=1000,lag=2,trim=0.1,tolerance=1e-10,
    fixed.beta=NIB12MTN.tvecm$model.specific$beta,ngridTh=1000,type="3Reg",
    boot.type="FixedReg",hpc="none",trace=FALSE))
TimeUsed.NIB12MTN.Res.3Reg<-system.time(NIB12MTN.HSTest.ResBoot.3Reg<-
    TVECM.XHStest(NIB12MTN,nboot=1000,lag=2,trim=0.1,
    tolerance=1e-10,fixed.beta=NIB12MTN.tvecm$model.specific$beta,
    ngridTh=1000,type="3Reg",boot.type="ResBoot",hpc="none",trace=FALSE))
TimeUsed.NIB12MTN.Fix.2Reg<-system.time(NIB12MTN.HSTest.FixedReg.2Reg<-
    TVECM.XHStest(NIB12MTN,nboot=1000,lag=2,trim=0.1,
    tolerance=1e-10,fixed.beta=NIB12MTN.tvecm$model.specific$beta,
    ngridTh=1000,type="2Reg",boot.type="FixedReg",hpc="none",trace=FALSE))
TimeUsed.NIB12MTN.Res.2Reg<-system.time(NIB12MTN.HSTest.ResBoot.2Reg<-
    TVECM.XHStest(NIB12MTN,nboot=1000,lag=2,trim=0.1,
    tolerance=1e-10,fixed.beta=NIB12MTN.tvecm$model.specific$beta,
    ngridTh=1000,type="2Reg",boot.type="ResBoot",hpc="none",trace=FALSE))
TimeUsed.TestNIB12MTN.Fix.2Reg<-system.time(TestNIB12MTN.HSTest.FixedReg.2Reg<-
    TVECM.XHStest(TestNIB12MTN,nboot=1000,lag=2,
    trim=0.1,tolerance=1e-10,fixed.beta=TestNIB12MTN.tvecm$model.specific$beta,
    ngridTh=1000,type="2Reg",boot.type="FixedReg",hpc="none",trace=FALSE))
TimeUsed.TestNIB12MTN.Res.2Reg<-system.time(TestNIB12MTN.HSTest.ResBoot.2Reg<-
    TVECM.XHStest(TestNIB12MTN,nboot=1000,lag=2,trim=0.1,tolerance=1e-10,
    fixed.beta=TestNIB12MTN.tvecm$model.specific$beta,ngridTh=1000,type="2Reg",
    boot.type="ResBoot",hpc="none",trace=FALSE))
TimeUsed.TestNIB12MTN.Fix.3Reg<-system.time(TestNIB12MTN.HSTest.FixedReg.3Reg<-
    TVECM.XHStest(TestNIB12MTN,nboot=1000,lag=2,trim=0.1,tolerance=1e-10,
    fixed.beta=TestNIB12MTN.tvecm$model.specific$beta,ngridTh=1000,type="3Reg",
    boot.type="FixedReg",hpc="none",trace=FALSE))
TimeUsed.TestNIB12MTN.Res.3Reg<-system.time(TestNIB12MTN.HSTest.ResBoot.3Reg<-
    TVECM.XHStest(TestNIB12MTN,nboot=1000,lag=2,trim=0.1,tolerance=1e-10,
    fixed.beta=TestNIB12MTN.tvecm$model.specific$beta,ngridTh=1000,type="3Reg",
    boot.type="ResBoot",hpc="none",trace=FALSE))
## chunk number 49: makes.Result.HSTest.NIB12MTN
Result.HSTest<-c(1,NIB12MTN.HSTest.FixedReg.2Reg$args$boot.type,
    NIB12MTN.HSTest.FixedReg.2Reg$args$nboot,
    format(round(NIB12MTN.HSTest.FixedReg.2Reg$stat,digits=1),nsmall=1),
    format(NIB12MTN.HSTest.FixedReg.2Reg$PvalBoot,nsmall=3),
    round(TimeUsed.NIB12MTN.Fix.2Reg[["elapsed"]],digits=1))
attr(Result.HSTest,"dimnames")<-NULL
Result.HSTest<-rbind(Result.HSTest,c(2,NIB12MTN.HSTest.FixedReg.3Reg$args$boot.type,
    NIB12MTN.HSTest.FixedReg.3Reg$args$nboot,
    round(NIB12MTN.HSTest.FixedReg.3Reg$stat,digits=1),
    NIB12MTN.HSTest.FixedReg.3Reg$PvalBoot,
    round(TimeUsed.NIB12MTN.Fix.3Reg[["elapsed"]],digits=1)),deparse.level=0)
Result.HSTest<-rbind(Result.HSTest,c(1,NIB12MTN.HSTest.ResBoot.2Reg$args$boot.type,
    NIB12MTN.HSTest.ResBoot.2Reg$args$nboot,
    format(round(NIB12MTN.HSTest.ResBoot.2Reg$stat,digits=1),nsmall=1),
    format(NIB12MTN.HSTest.ResBoot.2Reg$PvalBoot,nsmall=3),
    round(TimeUsed.NIB12MTN.Res.2Reg[["elapsed"]],digits=1)),deparse.level=0)
Result.HSTest<-rbind(Result.HSTest,c(2,NIB12MTN.HSTest.ResBoot.3Reg$args$boot.type,
    NIB12MTN.HSTest.ResBoot.3Reg$args$nboot,
    round(NIB12MTN.HSTest.ResBoot.3Reg$stat,digits=1),
    format(NIB12MTN.HSTest.ResBoot.3Reg$PvalBoot,nsmall=3),
    format(round(TimeUsed.NIB12MTN.Res.3Reg[["elapsed"]],digits=1),nsmall=1)),
    deparse.level=0)
Result.HSTest<-rbind(Result.HSTest,c(1,TestNIB12MTN.HSTest.FixedReg.2Reg$args$boot.type,
    TestNIB12MTN.HSTest.FixedReg.2Reg$args$nboot,
    format(round(TestNIB12MTN.HSTest.FixedReg.2Reg$stat,digits=1),nsmall=1),
    TestNIB12MTN.HSTest.FixedReg.2Reg$PvalBoot,
    round(TimeUsed.TestNIB12MTN.Fix.2Reg[["elapsed"]],digits=1)),deparse.level=0)
Result.HSTest<-rbind(Result.HSTest,c(2,TestNIB12MTN.HSTest.FixedReg.3Reg$args$boot.type,
    TestNIB12MTN.HSTest.FixedReg.3Reg$args$nboot,
    round(TestNIB12MTN.HSTest.FixedReg.3Reg$stat,digits=1),
    format(TestNIB12MTN.HSTest.FixedReg.3Reg$PvalBoot,nsmall=3),
    round(TimeUsed.TestNIB12MTN.Fix.3Reg[["elapsed"]],digits=1)),deparse.level=0)
Result.HSTest<-rbind(Result.HSTest,c(1,TestNIB12MTN.HSTest.ResBoot.2Reg$args$boot.type,
    TestNIB12MTN.HSTest.ResBoot.2Reg$args$nboot,
    format(round(TestNIB12MTN.HSTest.ResBoot.2Reg$stat,digits=1),nsmall=1),
    format(TestNIB12MTN.HSTest.ResBoot.2Reg$PvalBoot,nsmall=3),
    round(TimeUsed.TestNIB12MTN.Res.2Reg[["elapsed"]],digits=1)),deparse.level=0)
Result.HSTest<-rbind(Result.HSTest,c(2,TestNIB12MTN.HSTest.ResBoot.3Reg$args$boot.type,
    TestNIB12MTN.HSTest.ResBoot.3Reg$args$nboot,
    round(TestNIB12MTN.HSTest.ResBoot.3Reg$stat,digits=1),
    TestNIB12MTN.HSTest.ResBoot.3Reg$PvalBoot,
    round(TimeUsed.TestNIB12MTN.Res.3Reg[["elapsed"]],digits=1)),deparse.level=0)
Result.HSTest<-cbind(c("NIB12MTN","","","","TestNIB12MTN","","",""),
    Result.HSTest)
colnames(Result.HSTest)<-c("Data set","nthresh","boot type","nboot",
    "supLM","P-value","Seconds")
tab.Result.HSTest<-xtable(Result.HSTest,align=c("|l|l|l|l|l|r|r|r|"),
    digits=c(0,0,0,0,0,1,3,1))
## chunk number 50: tab.Result.HSTest
print(tab.Result.HSTest,hline.after=c(-1,0,4,nrow(tab.Result.HSTest)),
    floating=FALSE,quote=FALSE,include.colnames=TRUE,include.rownames=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 51: fig.res.models
NIB12MTN.tvecm1.res.zoo<-zooreg(NIB12MTN.tvecm$residuals,frequency=12,
    start=c(1985,8))
NIB12MTN.vecm.res.zoo<-zooreg(NIB12MTN.vecm$residuals,frequency=12,
    start=c(1985,8))
par(mfrow=c(1,2))
plot(NIB12MTN.tvecm1.res.zoo[,1],main="Residuals of NIB12M",xlab="Time",
    ylab="",type="p",cex=0.5,col="black",ylim=range(NIB12MTN.vecm.res.zoo[,1]))
lines(NIB12MTN.vecm.res.zoo[,1],type="p",cex=0.5,col="red")
abline(0,0)
legend("topright",c("TVECM","VECM"),bty="o",col=c("black","red"),lty=rep(1,2))
plot(NIB12MTN.tvecm1.res.zoo[,2],main="Residuals of NIBTN",xlab="Time",ylab="",
    type="p",cex=0.5,col="black",ylim=range(NIB12MTN.vecm.res.zoo[,2]))
lines(NIB12MTN.vecm.res.zoo[,2],type="p",cex=0.5,col="red")
abline(0,0)
legend("topright",c("TVECM","VECM"),bty="o",col=c("black","red"),lty=rep(1,2))
## chunk number 52: make.SSR.HSTest
options(warn=-1)
SSR.HSTest<-c(1,format(round(NIB12MTN.HSTest.FixedReg.2Reg$SSR,digits=1),nsmall=1),
    round(NIB12MTN.HSTest.FixedReg.2Reg$minThSSR,digits=3),
    round(NIB12MTN.HSTest.FixedReg.2Reg$PercentsSSR,digits=1))
attr(SSR.HSTest,"dimnames")<-NULL
SSR.HSTest<-rbind(SSR.HSTest,c(2,round(NIB12MTN.HSTest.FixedReg.3Reg$SSR,digits=1),
    round(NIB12MTN.HSTest.FixedReg.3Reg$minThSSR,digits=3),
    round(NIB12MTN.HSTest.FixedReg.3Reg$PercentsSSR,digits=1)),deparse.level=0)
SSR.HSTest<-rbind(SSR.HSTest,c(1,round(TestNIB12MTN.HSTest.FixedReg.2Reg$SSR,digits=1),
    format(round(TestNIB12MTN.HSTest.FixedReg.2Reg$minThSSR,digits=3),nsmall=3),
    round(TestNIB12MTN.HSTest.FixedReg.2Reg$PercentsSSR,digits=1)),deparse.level=0)
SSR.HSTest<-rbind(SSR.HSTest,c(2,round(TestNIB12MTN.HSTest.FixedReg.3Reg$SSR,digits=1),
    round(TestNIB12MTN.HSTest.FixedReg.3Reg$minThSSR,digits=3),
    round(TestNIB12MTN.HSTest.FixedReg.3Reg$PercentsSSR,digits=1)),deparse.level=0)
for (i in 1:4) {
    if (SSR.HSTest[i,1]==SSR.HSTest[i,6]) {
        SSR.HSTest[i,7]<- SSR.HSTest[i,5]
        SSR.HSTest[i,6]<-NA
        SSR.HSTest[i,5]<- SSR.HSTest[i,4]
        SSR.HSTest[i,4]<-NA
    }
}
options(warn=0)
SSR.HSTest<-cbind(c("NIB12MTN","","TestNIB12MTN",""),SSR.HSTest)
colnames(SSR.HSTest)<-c("Data set","nthresh","SSR","$\\gamma_1$","$\\gamma_2$",
    "ndown","nmiddle","nup")
tab.SSR.HSTest<-xtable(SSR.HSTest,align=c("|l|l|l|r|r|r|r|r|r|"),
    digits=c(0,0,0,1,3,3,1,1,1))
## chunk number 53: tab.SSR.HSTest
print(tab.SSR.HSTest,hline.after=c(-1,0,2,nrow(tab.SSR.HSTest)),
    floating=FALSE,quote=FALSE,include.colnames=TRUE,include.rownames=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 54: make.setarTest
sNIB12MTN<- as.zoo(ts(NIB12MTN%*%NIB12MTN.tvecm$model.specific$coint,
    frequency=12,start=c(1985,5),end=c(2010,12)))
sTestNIB12MTN<- as.zoo(ts(TestNIB12MTN%*%TestNIB12MTN.tvecm$model.specific$coint,
    frequency=12,start=c(1985,5),end=c(2010,12)))
sTestNIB12MTN.setarTest1vs<-setarTest(sTestNIB12MTN,m=3,nboot=1000,
    trim=0.1,test="1vs",hpc="none",check=FALSE)
sNIB12MTN.setarTest1vs<-setarTest(sNIB12MTN,m=3,nboot=1000,trim=0.1,
    test="1vs",hpc="none",check=FALSE)
sTestNIB12MTN.setarTest2vs3<-setarTest(sTestNIB12MTN,m=3,nboot=1000,
    trim=0.1,test="2vs3",hpc="none",check=FALSE)
sNIB12MTN.setarTest2vs3<-setarTest(sNIB12MTN,m=3,nboot=1000,trim=0.1,
    test="2vs3",hpc="none",check=FALSE)
## chunk number 55: make.setarTest.table
sNIB12MTN.setarTest<-c(round(sNIB12MTN.setarTest1vs$Ftests[1,1],digits=1),
    round(sNIB12MTN.setarTest1vs$CriticalValBoot[1,1:4],digits=1),
    round(sNIB12MTN.setarTest1vs$PvalBoot[1],digits=3))
sNIB12MTN.setarTest<-rbind(sNIB12MTN.setarTest,
    c(round(sNIB12MTN.setarTest1vs$Ftests[1,2],digits=1),
    round(sNIB12MTN.setarTest1vs$CriticalValBoot[2,1:4],digits=1),
    round(sNIB12MTN.setarTest1vs$PvalBoot[2],digits=3)))
sNIB12MTN.setarTest<-rbind(sNIB12MTN.setarTest,
    c(round(sNIB12MTN.setarTest2vs3$Ftests[1,3],digits=1),
    round(sNIB12MTN.setarTest2vs3$CriticalValBoot[1,1:4],digits=1),
    round(sNIB12MTN.setarTest2vs3$PvalBoot[1],digits=3)))
```

```
colnames(sNIB12MTN.setarTest)<-c("F-test","90\\%","95\\%",
    "97.5\\%","99\\%","P-value")
rownames(sNIB12MTN.setarTest)<-c("1vs2","1vs3","2vs3")
tab.sNIB12MTN.setarTest<-xtable(sNIB12MTN.setarTest,align=c("|l|r|r|r|r|r|r|"),
    digits=c(0,1,1,1,1,1,3))
sTestNIB12MTN.setarTest<-c(round(sTestNIB12MTN.setarTest1vs$Ftests[1,1],digits=1),
    round(sTestNIB12MTN.setarTest1vs$CriticalValBoot[1,1:4],digits=1),
    round(sTestNIB12MTN.setarTest1vs$PvalBoot[1],digits=3))
sTestNIB12MTN.setarTest<-rbind(sTestNIB12MTN.setarTest,
    c(round(sTestNIB12MTN.setarTest1vs$Ftests[1,2],digits=1),
    round(sTestNIB12MTN.setarTest1vs$CriticalValBoot[2,1:4],digits=1),
    round(sTestNIB12MTN.setarTest1vs$PvalBoot[2],digits=3)))
sTestNIB12MTN.setarTest<-rbind(sTestNIB12MTN.setarTest,
    c(round(sTestNIB12MTN.setarTest2vs3$Ftests[1,3],digits=1),
    round(sTestNIB12MTN.setarTest2vs3$CriticalValBoot[1,1:4],digits=1),
    round(sTestNIB12MTN.setarTest2vs3$PvalBoot[1],digits=3)))
colnames(sTestNIB12MTN.setarTest)<-c("F-test","90\\%","95\\%",
    "97.5\\%","99\\%","P-value")
rownames(sTestNIB12MTN.setarTest)<-c("1vs2","1vs3","2vs3")
tab.sTestNIB12MTN.setarTest<-xtable(sTestNIB12MTN.setarTest,
    align=c("|l|r|r|r|r|r|r|"),digits=c(0,1,1,1,1,1,3))
## chunk number 56: tab.sNIB12MTN.setarTest
print(tab.sNIB12MTN.setarTest,hline.after=c(-1,0,nrow(tab.sNIB12MTN.setarTest)),
    floating=FALSE,quote=FALSE,include.colnames=TRUE,include.rownames=TRUE,
    sanitize.text.function = function(x){x})
## chunk number 57: tab.sTestNIB12MTN.setarTest
print(tab.sTestNIB12MTN.setarTest,hline.after=c(-1,0,nrow(tab.sTestNIB12MTN.setarTest)),
    floating=FALSE,quote=FALSE,include.colnames=TRUE,include.rownames=TRUE,
    sanitize.text.function = function(x){x})
## chunk number 58: fig.sNIB12MTN.setarTest
par(mfrow=c(3,1))
plot(sNIB12MTN.setarTest1vs)
plot(sNIB12MTN.setarTest2vs3)
## chunk number 59: fig.sTestNIB12MTN.setarTest
par(mfrow=c(2,1))
plot(sTestNIB12MTN.setarTest1vs)
plot(sTestNIB12MTN.setarTest2vs3)
## chunk number 60: Ljung.Box.loop
Ljung.Pval<-matrix(0,nrow=3,ncol=9)
for (i in 1:3) {
    for (j in 0:8) {
        Ljung.Pval[i,j+1]<-Box.test(arima(sTestNIB12MTN,order=c(i,0,j))$residuals,
            lag=11+i+j,fitdf=i+j,type="Ljung")$p.value
    }
}
rownames(Ljung.Pval)<-c("$p=1$","$p=2$","$p=3$")
colnames(Ljung.Pval)<-c("$q=0$","$q=1$","$q=2$","$q=3$","$q=4$","$q=5$",
    "$q=6$","$q=7$","$q=8$")
tab.Ljung.Pval<-xtable(Ljung.Pval,align=c("|l|l|l|l|l|l|l|l|l|l|"),
    digits=c(3,3,3,3,3,3,3,3,3,3))
## chunk number 61: tab.sTestNIB12MTN.LjungTest
```

```
print(tab.Ljung.Pval,hline.after=c(-1,0,nrow(Ljung.Pval)),
    floating=FALSE,quote=FALSE,include.colnames=TRUE,include.rownames=TRUE,
    sanitize.text.function = function(x){x})
## chunk number 62: fig.sTestNIB12MTN.res.linear
linear1.residuals<-linear(sTestNIB12MTN,m=1)$residuals
arma28.residuals<-arima(sTestNIB12MTN,order=c(2,0,8))$residuals
par(mfrow=c(2,2))
acf(linear1.residuals)
pacf(linear1.residuals)
acf(arma28.residuals)
pacf(arma28.residuals)
## chunk number 63: fig.sNIB12MTN.res.setar
setar2.residuals<-setar(sNIB12MTN,m=2)$residuals
setar3.residuals<-setar(sNIB12MTN,m=3)$residuals
par(mfrow=c(2,2))
acf(setar2.residuals)
pacf(setar2.residuals)
acf(setar3.residuals)
pacf(setar3.residuals)
## chunk number 64: make.tab.setar3.summary
sNIB12MTN.setar3<-setar(sNIB12MTN,m=3,mH=3,mL=3,nthresh=1,
    include="const",trim=0.1)
setar3.summary<-summary(sNIB12MTN.setar3)
tab.setar3.summary<-xtable(setar3.summary$coef,align=c("|r|r|r|r|r|"),
    digits=c(3,3,3,3,3))
## chunk number 65: tab.setar3.summary
print(tab.setar3.summary,hline.after=c(-1,0,nrow(tab.setar3.summary)),
    floating=FALSE,quote=FALSE,include.colnames=TRUE,include.rownames=TRUE,
    sanitize.text.function = function(x){x})
## chunk number 66: compare.models
mod.test<-list()
sNIB12MTN.train<-window(sNIB12MTN,end=c(2010,2))
sNIB12MTN.test<-window(sNIB12MTN,start=c(2010,3))
mod.test[["linear1"]]<-linear(sNIB12MTN.train,m=1)
mod.test[["linear3"]]<-linear(sNIB12MTN.train,m=3)
mod.test[["setar1"]]<-setar(sNIB12MTN.train,m=1)
mod.test[["setar3"]]<-setar(sNIB12MTN.train,m=3)
frc.test<-lapply(mod.test,predict,n.ahead=10)
Thresh1<-mod.test[["setar1"]]$model.specific$coefficients[["th"]]
mod.Ttest<-list()
sTestNIB12MTN.train<-window(sTestNIB12MTN,end=c(2010,2))
sTestNIB12MTN.test<-window(sTestNIB12MTN,start=c(2010,3))
mod.Ttest[["linear1"]]<-linear(sTestNIB12MTN.train,m=1)
mod.Ttest[["arma(2,8)"]]<-arima(sTestNIB12MTN.train,order=c(2,0,8))
mod.Ttest[["setar1"]]<-setar(sTestNIB12MTN.train,m=1)
mod.Ttest[["setar3"]]<-setar(sTestNIB12MTN.train,m=3)
frc.Ttest<-lapply(mod.Ttest,predict,n.ahead=10)
frc.Ttest[["arma(2,8)"]]<-frc.Ttest[["arma(2,8)"]]$pred
## chunk number 67: fig.sNIB12MTN.predict
par(mfrow=c(1,1))
plot(3:12,sNIB12MTN.test,xlab="Month in 2010",ylim=c(-2,2),type="l",lwd=2,
    lty=1,col=xcolours[1])
for (i in 1:length(frc.test)) lines(3:12,frc.test[[i]],lty=1,col=xcolours[i+1],
    type="l",lwd=2)
legend(4,2.0,lty=1,col=xcolours[1:(length(frc.test)+1)],lwd=2,
    legend=c("observed",names(frc.test)))
## chunk number 68: fig.sTestNIB12MTN.predict
par(mfrow=c(1,1))
plot(3:12,sTestNIB12MTN.test,xlab="Month in 2010",lty=1,,col=xcolours[1],
    ylim=c(-0.5,2),type="l",lwd=2)
for (i in 1:length(frc.Ttest)) lines(3:12,frc.Ttest[[i]],
    lty=1,col=xcolours[i+1],type="l",lwd=2)
legend(4,2.0,lty=1,col=xcolours[1:(length(frc.Ttest)+1)],lwd=2,
    legend=c("observed",names(frc.Ttest)))
## chunk number 69: test.pairs
renter<-read.table("renter_mnd.sdv",colClasses=c("character",rep("numeric",9),
    rep("NULL",11),rep("numeric",3),rep("NULL",15)),sep=";",dec=",",header=TRUE)
 # har hentet sdv-filen fra
 # http://www.norges-bank.no/templates/article____55483.aspx
for (i in 2:10) {
    colnames(renter)[i]<-substr(colnames(renter)[i],7,9)
    if (substr(colnames(renter)[i],3,3)==".") {
        colnames(renter)[i]<-substr(colnames(renter)[i],1,2)
    }
    if (substr(colnames(renter)[i],2,2)==".") {
        colnames(renter)[i]<-paste(substr(colnames(renter)[i],1,1),
            substr(colnames(renter)[i],3,3),sep="")
    }
    colnames(renter)[i]<-paste("NIB",colnames(renter)[i],sep="")
}
for (i in 3:4) {
    substr(colnames(renter)[i],5,5)<-"W"
}
for (i in 11:13) {
    if (length(colnames(renter)[i])==6) {
        colnames(renter)[i]<-paste(substr(colnames(renter)[i],1,4),
            substr(colnames(renter)[i],6,6),sep="")
    } else {
       colnames(renter)[i]<-paste(substr(colnames(renter)[i],1,4),
            substr(colnames(renter)[i],6,7),sep="")
   }
}
Result<-list()
j<-0
for(i1 in 2:12) {
    for (i2 in (i1+1):13) {
        j<-j+1
        Result[[j]]<-list()
        rate1rate2<-na.contiguous(zoo(renter[c(i2,i1)],as.yearmon(renter$Dato,"%b-%y")))
        rate1<-zoo(rate1rate2[,1],index(rate1rate2))
        rate2<-zoo(rate1rate2[,2],index(rate1rate2))
        rate1[index(rate1)=="sep 1992"]<-(coredata(rate1[index(rate1)=="aug 1992"])
            +coredata(rate1[index(rate1)=="okt 1992"]))/2
        DiffOctJan<-coredata(rate1[index(rate1)=="jan 1993"])-
            coredata(rate1[index(rate1)=="okt 1992"])
        rate1[index(rate1)=="nov 1992"]<-coredata(rate1[index(rate1)=="okt 1992"])+
            DiffOctJan/3
        rate1[index(rate1)=="des 1992"]<-coredata(rate1[index(rate1)=="okt 1992"])+
            DiffOctJan*2/3
        rate2[index(rate2)=="sep 1992"]<-(coredata(rate2[index(rate2)=="aug 1992"])
            +coredata(rate2[index(rate2)=="okt 1992"]))/2
        DiffOctJan<-coredata(rate2[index(rate2)=="jan 1993"])-
            coredata(rate2[index(rate2)=="okt 1992"])
        rate2[index(rate2)=="nov 1992"]<-coredata(rate2[index(rate2)=="okt 1992"])+
            DiffOctJan/3
        rate2[index(rate2)=="des 1992"]<-coredata(rate2[index(rate2)=="okt 1992"])+
            DiffOctJan*2/3
        rate1rate2<-cbind(rate1,rate2)
        colnames(rate1rate2)<-c(colnames(renter[i2]),colnames(renter[i1]))
        Spread<-rate1-rate2
        Result[[j]]$name1<-colnames(renter)[i2]
        Result[[j]]$name2<-colnames(renter)[i1]
        Result[[j]]$rate1.DF<-ur.df(rate1,lags=10,selectlags="AIC",type='none')
        Result[[j]]$rate2.DF<-ur.df(rate2,lags=10,selectlags="AIC",type='none')
        Result[[j]]$Spread.DF<-ur.df(Spread,lags=10,selectlags="AIC",type='none')
        Result[[j]]$ca.jo <- summary(ca.jo(rate1rate2,type='trace',K=3,ecdet="none"))
        Result[[j]]$setarD0<-try(setar(Spread,m=4,nthresh=2,d=1,trim=0.1,
            thDelay=0),silent=TRUE)
        Result[[j]]$setarTest1vsD0<-try(setarTest(Spread,m=4,nboot=1000,trim=0.1,
            test="1vs",hpc="none",d=1,thDelay=0,check=FALSE),silent=TRUE)
        Result[[j]]$setarTest2vs3D0<-try(setarTest(Spread,m=4,nboot=1000,
            trim=0.1,test="2vs3",hpc="none",d=1,
            thDelay=0,check=FALSE),silent=TRUE)
        Result[[j]]$setarDx<-try(setar(Spread,m=4,nthresh=2,d=1,trim=0.1,
            thDelay=0:3),silent=TRUE)
        Result[[j]]$setarTest1vsDx<-try(setarTest(Spread,m=4,nboot=1000,trim=0.1,
            test="1vs",hpc="none",d=1,thDelay=Result[[j]]$setarDx$model.specific$thDelay,
            check=FALSE),silent=TRUE)
        Result[[j]]$setarTest2vs3Dx<-try(setarTest(Spread,m=4,nboot=1000,trim=0.1,
            test="2vs3",hpc="none",d=1,thDelay=Result[[j]]$setarDx$model.specific$thDelay,
            check=FALSE),silent=TRUE)
    }
}
jmax<-j
## chunk number 70: make.result.pairs
DF<-numeric(0)
Johansen<-numeric(0)
setarTest1vs2D0<-numeric(0)
setarTest1vs3D0<-numeric(0)
setarTest2vs3D0<-numeric(0)
setar.modelD0<-numeric(0)
setarTest1vs2Dx<-numeric(0)
setarTest1vs3Dx<-numeric(0)
```

```
setarTest2vs3Dx<-numeric(0)
setar.modelDx<-numeric(0)
NoOfCoeff<-length(Result[[1]]$setarD0$model.specific$coefficients)
for (j in 1:jmax) {
    DF<-rbind(DF,c(Result[[j]]$name1,Result[[j]]$name2,
        round(Result[[j]]$rate1.DF@teststat,digits=2),
        round(Result[[j]]$rate2.DF@teststat,digits=2),
        round(Result[[j]]$Spread.DF@teststat,digits=2),Result[[j]]$Spread.DF@cval))
    Johansen<-rbind(Johansen, c(Result[[j]]$name1,Result[[j]]$name2,
        round(Result[[j]]$ca.jo@teststat[2],digits=2),
        Result[[j]]$ca.jo@cval[2,],round(Result[[j]]$ca.jo@V[2,1],digits=3)))
    if (class(Result[[j]]$setarTest1vsD0)=="try-error") {
        setarTest1vs2D0<-rbind(setarTest1vs2D0,
            c(Result[[j]]$name1,Result[[j]]$name2,NA,NA,NA,NA,NA))
        setarTest1vs3D0<-rbind(setarTest1vs3D0,
            c(Result[[j]]$name1,Result[[j]]$name2,NA,NA,NA,NA,NA))
    } else {
        setarTest1vs2D0<-rbind(setarTest1vs2D0,c(Result[[j]]$name1,Result[[j]]$name2,
            round(Result[[j]]$setarTest1vsD0$Ftests[1],digits=1),
            format(Result[[j]]$setarTest1vsD0$PvalBoot[1],nsmall=3),
            Result[[j]]$setarTest1vsD0$args$thDelay,
            Result[[j]]$setarTest1vsD0$args$m,Result[[j]]$setarTest1vsD0$args$nboot))
        setarTest1vs3D0<-rbind(setarTest1vs3D0,c(Result[[j]]$name1,Result[[j]]$name2,
            round(Result[[j]]$setarTest1vsD0$Ftests[2],digits=1),
            format(Result[[j]]$setarTest1vsD0$PvalBoot[2],nsmall=3),
            Result[[j]]$setarTest1vsD0$args$thDelay,
            Result[[j]]$setarTest1vsD0$args$m,Result[[j]]$setarTest1vsD0$args$nboot))
    }
    if (class(Result[[j]]$setarTest2vs3D0)=="try-error") {
        setarTest2vs3D0<-rbind(setarTest2vs3D0,
            c(Result[[j]]$name1,Result[[j]]$name2,NA,NA,NA,NA,NA))
    } else {
        setarTest2vs3D0<-rbind(setarTest2vs3D0,c(Result[[j]]$name1,Result[[j]]$name2,
            round(Result[[j]]$setarTest2vs3D0$Ftests[3],digits=1),
            format(Result[[j]]$setarTest2vs3D0$PvalBoot,nsmall=3),
            Result[[j]]$setarTest2vs3D0$args$thDelay,
            Result[[j]]$setarTest2vs3D0$args$m,Result[[j]]$setarTest2vs3D0$args$nboot))
    }
    if (class(Result[[j]]$setar.modelD0)=="try-error") {
        setar.modelD0<-rbind(setar.modelD0,
            c(Result[[j]]$name1,Result[[j]]$name2,NA,NA,NA,NA,NA,NA,NA))
    }else {
        setar.modelD0<-rbind(setar.modelD0,c(Result[[j]]$name1,Result[[j]]$name2,
            format(round(Result[[j]]$setarD0$model.specific$coefficients[
                (NoOfCoeff-1):NoOfCoeff],digits=3),nsmall=3),
            format(100*round(Result[[j]]$setarD0$model.specific$RegProp,digits=3),nsmall=1),
            Result[[j]]$setarD0$model.specific$thDelay,Result[[j]]$setarD0$str$m))
    }
    if (class(Result[[j]]$setarTest1vsDx)=="try-error") {
        setarTest1vs2Dx<-rbind(setarTest1vs2Dx,
            c(Result[[j]]$name1,Result[[j]]$name2,NA,NA,NA,NA,NA))
        setarTest1vs3Dx<-rbind(setarTest1vs3Dx,
            c(Result[[j]]$name1,Result[[j]]$name2,NA,NA,NA,NA,NA))
    } else {
        setarTest1vs2Dx<-rbind(setarTest1vs2Dx,c(Result[[j]]$name1,Result[[j]]$name2,
            round(Result[[j]]$setarTest1vsDx$Ftests[1],digits=1),
            format(Result[[j]]$setarTest1vsDx$PvalBoot[1],nsmall=3),
            Result[[j]]$setarTest1vsDx$args$thDelay,
            Result[[j]]$setarTest1vsDx$args$m,Result[[j]]$setarTest1vsDx$args$nboot))
        setarTest1vs3Dx<-rbind(setarTest1vs3Dx,c(Result[[j]]$name1,Result[[j]]$name2,
            round(Result[[j]]$setarTest1vsDx$Ftests[2],digits=1),
            format(Result[[j]]$setarTest1vsDx$PvalBoot[2],nsmall=3),
            Result[[j]]$setarTest1vsDx$args$thDelay,
            Result[[j]]$setarTest1vsDx$args$m,Result[[j]]$setarTest1vsDx$args$nboot))
    }
    if (class(Result[[j]]$setarTest2vs3Dx)=="try-error") {
        setarTest2vs3Dx<-rbind(setarTest2vs3Dx,
            c(Result[[j]]$name1,Result[[j]]$name2,NA,NA,NA,NA,NA))
    } else {
        setarTest2vs3Dx<-rbind(setarTest2vs3Dx,c(Result[[j]]$name1,Result[[j]]$name2,
            round(Result[[j]]$setarTest2vs3Dx$Ftests[3],digits=1),
            format(Result[[j]]$setarTest2vs3Dx$PvalBoot,nsmall=3),
            Result[[j]]$setarTest2vs3Dx$args$thDelay,
            Result[[j]]$setarTest2vs3Dx$args$m,Result[[j]]$setarTest2vs3Dx$args$nboot))
    }
    if (class(Result[[j]]$setar.modelDx)=="try-error") {
        setar.modelDx<-rbind(setar.modelDx,
            c(Result[[j]]$name1,Result[[j]]$name2,NA,NA,NA,NA,NA,NA,NA))
    }else {
        setar.modelDx<-rbind(setar.modelDx,c(Result[[j]]$name1,Result[[j]]$name2,
            format(round(Result[[j]]$setarDx$model.specific$coefficients[
                (NoOfCoeff-1):NoOfCoeff],digits=3),nsmall=3),
            format(100*round(Result[[j]]$setarDx$model.specific$RegProp,digits=3),
                nsmall=1),
            Result[[j]]$setarDx$model.specific$thDelay,Result[[j]]$setarDx$str$m))
    }
}
colnames(DF)<-c("rate1","rate2","ADF rate1","ADF rate2","ADF Spread","1pct",
    "5pct","10pct")
colnames(Johansen)<-c("rate1","rate2","trace statistic","10pct","5pct","1pct","-beta")
colnames(setarTest1vs2D0)<-c("rate1","rate2","Ftest","Pvalue","thDelay","m","nboot")
colnames(setarTest1vs3D0)<-c("rate1","rate2","Ftest","Pvalue","thDelay","m","nboot")
colnames(setarTest2vs3D0)<-c("rate1","rate2","Ftest","Pvalue","thDelay","m","nboot")
colnames(setar.modelD0)<-c("rate1","rate2","th1","th2","ndown","nmiddle",
    "nup","thDelay","m")
colnames(setarTest1vs2Dx)<-c("rate1","rate2","Ftest","Pvalue","thDelay","m","nboot")
colnames(setarTest1vs3Dx)<-c("rate1","rate2","Ftest","Pvalue","thDelay","m","nboot")
colnames(setarTest2vs3Dx)<-c("rate1","rate2","Ftest","Pvalue","thDelay","m","nboot")
colnames(setar.modelDx)<-c("rate1","rate2","th1","th2","ndown","nmiddle",
    "nup","thDelay","m")
Result.setarTestD0<-numeric(0)
Result.setarTestDx<-numeric(0)
```

```
for (j in 1:jmax) {
    Result.setarTestD0<-rbind(Result.setarTestD0,c(setarTest1vs2D0[j,c(1,2,4)],
        setarTest1vs3D0[j,4],setarTest2vs3D0[j,4:5],setar.modelD0[j,3:7]))
    Result.setarTestDx<-rbind(Result.setarTestDx,c(setarTest1vs2Dx[j,c(1,2,4)],
        setarTest1vs3Dx[j,4],setarTest2vs3Dx[j,4],setar.modelDx[j,8],
        setar.modelDx[j,3:7]))
}
colnames(Result.setarTestD0)<-c("Rate1","Rate2","1vs2","1vs3","2vs3",
    "thDelay","th1","th2","ndown","nmiddle","nup")
colnames(Result.setarTestDx)<-c("Rate1","Rate2","1vs2","1vs3","2vs3",
    "thDelay","th1","th2","ndown","nmiddle","nup")
tab.Result.setarTestD0<-xtable(Result.setarTestD0,
    align=c("|l|l|l|r|r|r|r|r|r|r|r|r|"),digits=c(0,0,0,3,3,3,0,2,2,1,1,1))
tab.Result.setarTestDx<-xtable(Result.setarTestDx,
    align=c("|l|l|l|r|r|r|r|r|r|r|r|r|"),digits=c(0,0,0,3,3,3,0,2,2,1,1,1))
## chunk number 71: tab.Result.setarTestD0
print(tab.Result.setarTestD0,hline.after=c(0,nrow(tab.Result.setarTestD0)),
    floating=FALSE,quote=FALSE,include.colnames=FALSE,include.rownames=FALSE,
    add.to.row=list(pos=list(-1,1),
    command=c("\\hline\\multicolumn{2}{|l|}{\\textbf{Interest rates}} &
        \\multicolumn{3}{|l|}{\\textbf{P-value}} & \\textbf{thDelay} &
        \\multicolumn{2}{|l|}{\\textbf{Thresholds}} &
        \\multicolumn{3}{|l|}{\\textbf{{\\%} in each regime}}\\\\
        \\cline{1-5}\\cline{7-11}
        \\textbf{Rate 1} & \\textbf{Rate 2} & \\textbf{1vs2} &
        \\textbf{1vs3} & \\textbf{2vs3} & & \\textbf{th1} & \\textbf{th2} &
       \\textbf{L} & \\textbf{M} & \\textbf{H}\\\\","")),
      sanitize.text.function = function(x){x})
## chunk number 72: tab.Result.setarTestDx
print(tab.Result.setarTestDx,hline.after=c(-1,nrow(tab.Result.setarTestDx)),
    floating=FALSE,quote=FALSE,include.colnames=FALSE,include.rownames=FALSE,
    add.to.row=list(pos=list(-1,1),
    command=c("\\hline\\multicolumn{2}{|l|}{\\textbf{Interest rates}} &
        \\multicolumn{3}{|l|}{\\textbf{P-value}} & \\textbf{thDelay} &
        \\multicolumn{2}{|l|}{\\textbf{Thresholds}} &
        \\multicolumn{3}{|l|}{\\textbf{{\\%} in each regime}}\\\\
        \\cline{1-5}\\cline{7-11}
        \\textbf{Rate 1} & \\textbf{Rate 2} & \\textbf{1vs2} &
        \\textbf{1vs3} & \\textbf{2vs3} & & \\textbf{th1} & \\textbf{th2} &
       \\textbf{L} & \\textbf{M} & \\textbf{H}\\\\","")),
    sanitize.text.function = function(x){x})
## chunk number 73: testOutlier1992
NIB12MTNx<-na.contiguous(zoo(renter[c("NIB12M","NIBTN")],
    as.yearmon(renter$Dato,"%b-%y")))
NIB12Mx<-zoo(NIB12MTNx[,1],index(NIB12MTNx))
NIBTNx<-zoo(NIB12MTNx[,2],index(NIB12MTNx))
NIB12Mx[index(NIB12Mx)=="sep 1992"]<-(coredata(NIB12Mx[index(NIB12Mx)=="aug 1992"])+
    coredata(NIB12Mx[index(NIB12Mx)=="okt 1992"]))/2
DiffOctJan<-coredata(NIB12Mx[index(NIB12Mx)=="jan 1993"])-
    coredata(NIB12Mx[index(NIB12Mx)=="okt 1992"])
NIB12Mx[index(NIB12Mx)=="nov 1992"]<-coredata(NIB12Mx[index(NIB12Mx)=="okt 1992"])+
    DiffOctJan/3
NIB12Mx[index(NIB12Mx)=="des 1992"]<-coredata(NIB12Mx[index(NIB12Mx)=="okt 1992"])+
    DiffOctJan*2/3
NIBTNx[index(NIBTNx)=="sep 1992"]<-(coredata(NIBTNx[index(NIBTNx)=="aug 1992"])
    +coredata(NIBTNx[index(NIBTNx)=="okt 1992"]))/2
DiffOctJan<-coredata(NIBTNx[index(NIBTNx)=="jan 1993"])-
    coredata(NIBTNx[index(NIBTNx)=="okt 1992"])
NIBTNx[index(NIBTNx)=="nov 1992"]<-coredata(NIBTNx[index(NIBTNx)=="okt 1992"])+
    DiffOctJan/3
NIBTNx[index(NIBTNx)=="des 1992"]<-coredata(NIBTNx[index(NIBTNx)=="okt 1992"])+
    DiffOctJan*2/3
NIB12MTNx<-cbind(NIB12Mx,NIBTNx)
NIB12MTNx.tvecm<-TVECM(NIB12MTNx,nthresh=1,lag=2,ngridBeta=400,ngridTh=400,
    beta=list(int=c(0.85,1.25)),th1=list(int=c(-1.5,1.5)),plot=TRUE,
    trim=0.1,common="All",trace=FALSE)
TimeUsed.NIB12MTNx.Fix.2Reg<-system.time(NIB12MTNx.HSTest.FixedReg.2Reg<-
    TVECM.XHStest(NIB12MTNx,nboot=1000,lag=2,trim=0.1,
    tolerance=1e-10,fixed.beta=NIB12MTNx.tvecm$model.specific$beta,
    ngridTh=1000,type="2Reg",boot.type="FixedReg",hpc="none",trace=FALSE))
TimeUsed.NIB12MTNx.Res.2Reg<-system.time(NIB12MTNx.HSTest.ResBoot.2Reg<-
    TVECM.XHStest(NIB12MTNx,nboot=1000,lag=2,trim=0.1,
    tolerance=1e-10,fixed.beta=NIB12MTNx.tvecm$model.specific$beta,
    ngridTh=1000,type="2Reg",boot.type="ResBoot",hpc="none",trace=FALSE))
TimeUsed.NIB12MTNx.Fix.3Reg<-system.time(NIB12MTNx.HSTest.FixedReg.3Reg<-
    TVECM.XHStest(NIB12MTNx,nboot=1000,lag=2,trim=0.1,
    tolerance=1e-10,fixed.beta=NIB12MTNx.tvecm$model.specific$beta,
    ngridTh=1000,type="3Reg",boot.type="FixedReg",hpc="none",trace=FALSE))
TimeUsed.NIB12MTNx.Res.3Reg<-system.time(NIB12MTNx.HSTest.ResBoot.3Reg<-
    TVECM.XHStest(NIB12MTNx,nboot=1000,lag=2,trim=0.1,
    tolerance=1e-10,fixed.beta=NIB12MTNx.tvecm$model.specific$beta,
    ngridTh=1000,type="3Reg",boot.type="ResBoot",hpc="none",trace=FALSE))
sNIB12MTNx<- NIB12MTNx%*%NIB12MTNx.tvecm$model.specific$coint
dNIB12MTNx<- NIB12Mx-NIBTNx
sNIB12MTNx.setarTest1vs<-setarTest(sNIB12MTNx,m=3,nboot=1000,trim=0.1,
    test="1vs",hpc="none",check=FALSE)
sNIB12MTNx.setarTest2vs3<-setarTest(sNIB12MTNx,m=3,nboot=1000,trim=0.1,
    test="2vs3",hpc="none",check=FALSE)
dNIB12MTNx.setarTest1vs<-setarTest(dNIB12MTNx,m=3,nboot=1000,trim=0.1,
    test="1vs",hpc="none",check=FALSE)
dNIB12MTNx.setarTest2vs3<-setarTest(dNIB12MTNx,m=3,nboot=1000,trim=0.1,
    test="2vs3",hpc="none",check=FALSE)
## chunk number 74: make.xsetarTest.table
sNIB12MTNx.setarTest<-c(round(sNIB12MTNx.setarTest1vs$Ftests[1,1],digits=1),
    round(sNIB12MTNx.setarTest1vs$CriticalValBoot[1,1:4],digits=1),
    round(sNIB12MTNx.setarTest1vs$PvalBoot[1],digits=3))
sNIB12MTNx.setarTest<-rbind(sNIB12MTNx.setarTest,
    c(round(sNIB12MTNx.setarTest1vs$Ftests[1,2],digits=1),
    round(sNIB12MTNx.setarTest1vs$CriticalValBoot[2,1:4],digits=1),
    round(sNIB12MTNx.setarTest1vs$PvalBoot[2],digits=3)))
sNIB12MTNx.setarTest<-rbind(sNIB12MTNx.setarTest,
    c(round(sNIB12MTNx.setarTest2vs3$Ftests[1,3],digits=1),
    round(sNIB12MTNx.setarTest2vs3$CriticalValBoot[1,1:4],digits=1),
    round(sNIB12MTNx.setarTest2vs3$PvalBoot[1],digits=3)))
colnames(sNIB12MTNx.setarTest)<-c("F-test","90\\%","95\\%",
    "97.5\\%","99\\%","P-value")
rownames(sNIB12MTNx.setarTest)<-c("1vs2","1vs3","2vs3")
tab.sNIB12MTNx.setarTest<-xtable(sNIB12MTNx.setarTest,align=c("|l|r|r|r|r|r|r|"),
    digits=c(0,1,1,1,1,1,3))
## chunk number 75: tab.sNIB12MTNx.setarTest
print(tab.sNIB12MTNx.setarTest,hline.after=c(-1,0,nrow(tab.sNIB12MTNx.setarTest)),
    floating=FALSE,quote=FALSE,include.colnames=TRUE,include.rownames=TRUE,
    sanitize.text.function = function(x){x})
## chunk number 76: make.tab.simul.input.coeff
NIB12MTN.tvecmx<-TVECM(NIB12MTN,nthresh=1,lag=2,ngridBeta=400,ngridTh=200,
    beta=list(int=c(0.925,1.125)),plot=FALSE,trim=0.1,common="All",trace=FALSE)
NIB12MTN.tvecmx.summary<-summary(NIB12MTN.tvecmx)
simul.input.coeff<-rbind(t(NIB12MTN.tvecmx.summary$coefficients$Bdown),
      t(NIB12MTN.tvecmx.summary$coefficients$Bup),
      t(NIB12MTN.tvecmx.summary$coefficients$Bdown))
simul.input.coeff<-format(round(simul.input.coeff,digits=4),nsmall=4)
simul.input.coeff<-cbind(c("Lower","","","","","","Middle","","","","","",
      "Upper","","","","",""),dimnames(simul.input.coeff)[[1]],simul.input.coeff)
colnames(simul.input.coeff)[1:2]<-c("Regime","Term")
simul.input.coeff<-rbind(colnames(simul.input.coeff),simul.input.coeff)
simul.input.model.coeff<-xtable(simul.input.coeff,align=c("|l|l|l|r|r|"))
## chunk number 77: tab.simul.input.model
print(simul.input.model.coeff,hline.after=c(0,1,7,13,nrow(simul.input.model.coeff)),
    floating=FALSE,quote=FALSE,include.colnames=FALSE,include.rownames=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 78: simulate.tvecm
N<-2000
set.seed(5)
innov<-rmnorm(N,varcov=diag(2))
tvecm1.data<-TVECM.sim(B=matrix(c(0.0999,0.9871,0.0948,1.3375,0.4434,1.4526,
    0.0123,0.0056,-0.0426,-0.2651,0.0333,0.0539,-0.0424,0.0069,0.0142,0.0237,
    0.5554,0.8862,0.1809,-0.2722,-0.123,0.2262,-0.0789,-0.6334,0.0999,0.9871,
    0.0948,1.3375,0.4434,1.4526,0.0123,0.0056,-0.0426,-0.2651,0.0333,0.0539),
    ncol=18),nthresh=2,Thresh=c(-3.7,1.2),starting=matrix(c(0,1,0.5,1.5),ncol=2),
    beta=1.046,n=N,lag=2,include="const",innov=innov,show.parMat=FALSE)
## chunk number 79: compute.Rsim.rsim.ssim
coint<-c(1,-1.046)
Rsim<-tvecm1.data[,1]
rsim<-tvecm1.data[,2]
diffRsim<-diff(Rsim)
diffrsim<-diff(rsim)
ssim<-tvecm1.data%*%coint
diffssim<-diff(ssim)
ssimMinus1<-ssim[2:length(ssim)]
## chunk number 80: fig.Rsim.rsim
par(mfrow=c(2,1))
plot(Rsim,xlab="$t$",ylab="$y_{1t}$",type='l')
plot(rsim,xlab="$t$",ylab="$y_{2t}$",type='l')
```

```
## chunk number 81: fig.ssim
par(mfrow=c(2,1))
plot(1:300,ssim[1:300],ylab="$w_t$",xlab="$t$",
    main="The first 300 values of $w_t$",type='l')
abline(-3.7,0,col="red")
abline(1.2,0,col="green")
plot((N-299):N,ssim[(N-299):N],ylab="$w_t$",xlab="$t$",
    main="The last 300 values of $w_t$",type='l')
abline(-3.7,0,col="red")
abline(1.2,0,col="green")
## chunk number 82: fig.diffsimSeries
par(mfrow=c(3,1))
plot(ssimMinus1[1:300],diffRsim[1:300],type="l",ylab="$\\Delta y_{1t}$",
     xlab="$w_{t-1}$")
abline(v=-3.7,col="red")
abline(v=1.2,col="green")
plot(ssimMinus1[1:300],diffrsim[1:300],type="l",ylab="$\\Delta y_{2t}$",
     xlab="$w_{t-1}$")
abline(v=-3.7,col="red")
abline(v=1.2,col="green")
plot(ssimMinus1[1:300],diffssim[1:300],type="l",ylab="$\\Delta w_t$",
     xlab="$w_{t-1}$")
abline(v=-3.7,col="red")
abline(v=1.2,col="green")
## chunk number 83: fig.simul.tvecm1x.model
sink(file="errors.txt",type="output",split=FALSE)
simul.tvecm1x.model<-TVECM(tvecm1.data,nthresh=2,lag=2,ngridBeta=0,ngridTh=500,
    plot=TRUE,include="const",th1=list(int=c(-5,3)),th2=list(int=c(-5,3)),
    beta=list(exact=1.046),trim=0.05,common="All",trace=TRUE)
sink(file=NULL,type="output")
abline(v=-3.7)
abline(v=1.2)
text(-3.6,31000,pos=4,"$\\gamma=-3.7$")
text(1.3,33000,pos=4,"$\\gamma=1.2$")
## chunk number 84: estimate.tvecm
simul.tvecm1.model<-TVECM(tvecm1.data,nthresh=2,lag=2,ngridBeta=0,ngridTh=0,
    plot=FALSE,include="const",th1=list(exact=-3.7),th2=list(exact=1.2),
    beta=list(exact=1.046),trim=0.05,common="All",trace=FALSE)
## chunk number 85: make.tab.simul.tvecm1.coeff
simul.tvecm1.coeff<-rbind(t(summary(simul.tvecm1.model)$bigcoefficients$Bdown),
      t(summary(simul.tvecm1.model)$bigcoefficients$Bmiddle),
      t(summary(simul.tvecm1.model)$bigcoefficients$Bup))
simul.tvecm1.coeff<-cbind(c("Lower","","","","","","Middle","","","","","",
      "Upper","","","","",""),dimnames(simul.tvecm1.coeff)[[1]],simul.tvecm1.coeff)
colnames(simul.tvecm1.coeff)[1:2]<-c("Regime","Term")
simul.tvecm1.coeff<-rbind(colnames(simul.tvecm1.coeff),simul.tvecm1.coeff)
simul.tvecm1.model.coeff<-xtable(simul.tvecm1.coeff,align=c("|l|l|l|l|l|"))
## chunk number 86: tab.simul.tvecm1.model
print(simul.tvecm1.model.coeff,hline.after=c(0,1,7,13,nrow(simul.tvecm1.model.coeff)),
    floating=FALSE,quote=FALSE,include.colnames=FALSE,include.rownames=FALSE,
    sanitize.text.function = function(x){x})
```

```
## chunk number 87: simulated.responses
wsim<-ssim[4:length(ssim)]
Threshsim<-simul.tvecm1.model$model.specific$Thresh
Bdownsim<-simul.tvecm1.model$coefficients$Bdown
Bmiddlesim<-simul.tvecm1.model$coefficients$Bmiddle
Bupsim<-simul.tvecm1.model$coefficients$Bup
Res2Rsim<-ifelse(wsim<=Threshsim[1],Bdownsim[1,2]+Bdownsim[1,1]*wsim,
          ifelse(wsim<=Threshsim[2],Bmiddlesim[1,2]+Bmiddlesim[1,1]*wsim,
             Bupsim[1,2]+Bupsim[1,1]*wsim))
Res2rsim<-ifelse(wsim<=Threshsim[1],Bdownsim[2,2]+Bdownsim[2,1]*wsim,
          ifelse(wsim<=Threshsim[2],Bmiddlesim[2,2]+Bmiddlesim[2,1]*wsim,
             Bupsim[2,2]+Bupsim[2,1]*wsim))
ssim2<-cbind(Res2Rsim,Res2rsim)%*%simul.tvecm1.model$model.specific$coint
Indsim<-order(wsim)
## chunk number 88: fig.responses.sim
par(mfrow=c(1,1))
plot(wsim[Indsim],Res2rsim[Indsim],type="l",lty=1,col=xcolours[1],xlim=range(wsim),
     xlab="$w_{t-1}$",ylab="Response",ylim=range(Res2rsim))
axis(4)
abline(0,0,col="blue")
lines(wsim[Indsim],Res2Rsim[Indsim],type="l",lty=1,col=xcolours[2])
lines(wsim[Indsim],ssim2[Indsim],type="l",lty=1,col=xcolours[3])
legend("top",c("$\\Delta y_{2t}$","$\\Delta y_{1t}$","$\\Delta w_t$"),lty=1,
       col=xcolours[1:3])
## chunk number 89: simulate.HSTest.Fixed.3Reg
TimeUsed.Fixed.3Reg<-system.time(simul.tvecm1.test.Fixed.3Reg<-TVECM.XHStest(tvecm1.data,
    nboot=1000,lag=2,trim=0.05,tolerance=1e-10,fixed.beta=1.046,ngridTh=200,type="3Reg",
    boot.type="FixedReg",hpc="none",trace=FALSE))
## chunk number 90: simulate.HSTest.Res.3Reg
TimeUsed.Res.3Reg<-system.time(simul.tvecm1.test.Res.3Reg<-TVECM.XHStest(tvecm1.data,
    nboot=200,lag=2,trim=0.05,tolerance=1e-10,fixed.beta=1.046,ngridTh=200,type="3Reg",
    boot.type="ResBoot",hpc="none",trace=FALSE))
## chunk number 91: simulate.HSTest.2Reg
TimeUsed.Fixed.2Reg<-system.time(simul.tvecm1.test.Fixed.2Reg<-TVECM.XHStest(tvecm1.data,
    nboot=1000,lag=2,trim=0.05,tolerance=1e-10,fixed.beta=1.046,ngridTh=200,type="2Reg",
    boot.type="FixedReg",hpc="none",trace=FALSE))
TimeUsed.Res.2Reg<-system.time(simul.tvecm1.test.Res.2Reg<-TVECM.XHStest(tvecm1.data,
    nboot=1000,lag=2,trim=0.05,tolerance=1e-10,fixed.beta=1.046,ngridTh=200,type="2Reg",
    boot.type="ResBoot",hpc="none",trace=FALSE))
## chunk number 92: Test.grid.search
TimeUsed.Grid.Search.Fixed.3Reg<-system.time(Grid.Search.Fixed.3Reg<-
    TVECM.XHStest(tvecm1.data,nboot=2,lag=2,trim=0.05,tolerance=1e-10,
    fixed.beta=1.046,ngridTh=2000,type="3Reg",
    boot.type="FixedReg",hpc="none",trace=FALSE))
TimeUsed.Grid.Search.Fixed.2Reg<-system.time(Grid.Search.Fixed.2Reg<-
    TVECM.XHStest(tvecm1.data,nboot=2,lag=2,trim=0.05,tolerance=1e-10,
    fixed.beta=1.046,ngridTh=2000,type="2Reg",
    boot.type="FixedReg",hpc="none",trace=FALSE))
## chunk number 93: make.simul.Result.HSTest
options(warn=0)
simul.Result.HSTest<-c(1,length(simul.tvecm1.test.Fixed.2Reg$ths),
    simul.tvecm1.test.Fixed.2Reg$args$boot.type,
    simul.tvecm1.test.Fixed.2Reg$args$nboot,
    round(simul.tvecm1.test.Fixed.2Reg$stat,digits=1),
    format(simul.tvecm1.test.Fixed.2Reg$PvalBoot,nsmall=3),
    format(round(TimeUsed.Fixed.2Reg[["elapsed"]],digits=1),nsmall=1),
    round(simul.tvecm1.test.Fixed.2Reg$PercentsSSR,digits=1))
attr(simul.Result.HSTest,"dimnames")<-NULL
simul.Result.HSTest<-rbind(simul.Result.HSTest,
    c(2,length(simul.tvecm1.test.Fixed.3Reg$ths),
    simul.tvecm1.test.Fixed.3Reg$args$boot.type,
    simul.tvecm1.test.Fixed.3Reg$args$nboot,
    format(round(simul.tvecm1.test.Fixed.3Reg$stat,digits=1),nsmall=1),
    format(simul.tvecm1.test.Fixed.3Reg$PvalBoot,nsmall=3),
    round(TimeUsed.Fixed.3Reg[["elapsed"]],digits=1),
    format(round(simul.tvecm1.test.Fixed.3Reg$PercentsSSR,digits=1),nsmall=1)),
    deparse.level=0)
simul.Result.HSTest<-rbind(simul.Result.HSTest,
    c(1,length(simul.tvecm1.test.Res.2Reg$ths),
    simul.tvecm1.test.Res.2Reg$args$boot.type,
    simul.tvecm1.test.Res.2Reg$args$nboot,
    round(simul.tvecm1.test.Res.2Reg$stat,digits=1),
    format(simul.tvecm1.test.Res.2Reg$PvalBoot,nsmall=3),
    round(TimeUsed.Res.2Reg[["elapsed"]],digits=1),
    round(simul.tvecm1.test.Res.2Reg$PercentsSSR,digits=1)),deparse.level=0)
simul.Result.HSTest<-rbind(simul.Result.HSTest,
    c(2,length(simul.tvecm1.test.Res.3Reg$ths),
    simul.tvecm1.test.Res.3Reg$args$boot.type,
    simul.tvecm1.test.Res.3Reg$args$nboot,
    format(round(simul.tvecm1.test.Res.3Reg$stat,digits=1),nsmall=1),
    format(simul.tvecm1.test.Res.3Reg$PvalBoot,nsmall=3),
    round(TimeUsed.Res.3Reg[["elapsed"]],digits=1),
    format(round(simul.tvecm1.test.Res.3Reg$PercentsSSR,digits=1),nsmall=1)),
    deparse.level=0)
simul.Result.HSTest<-rbind(simul.Result.HSTest,
    c(1,length(Grid.Search.Fixed.2Reg$ths),
    Grid.Search.Fixed.2Reg$args$boot.type,
    Grid.Search.Fixed.2Reg$args$nboot,
    round(Grid.Search.Fixed.2Reg$stat,digits=1),
    format(Grid.Search.Fixed.2Reg$PvalBoot,nsmall=3),
    round(TimeUsed.Grid.Search.Fixed.2Reg[["elapsed"]],digits=1),
    format(round(Grid.Search.Fixed.2Reg$PercentsSSR,digits=1),nsmall=1)),
    deparse.level=0)
simul.Result.HSTest<-rbind(simul.Result.HSTest,
    c(2,length(Grid.Search.Fixed.3Reg$ths),
    Grid.Search.Fixed.3Reg$args$boot.type,
    Grid.Search.Fixed.3Reg$args$nboot,
    round(Grid.Search.Fixed.3Reg$stat,digits=1),
    format(Grid.Search.Fixed.3Reg$PvalBoot,nsmall=3),
    round(TimeUsed.Grid.Search.Fixed.3Reg[["elapsed"]],digits=1),
    format(round(Grid.Search.Fixed.3Reg$PercentsSSR,digits=1),nsmall=1)),
    deparse.level=0)
```

```
for (i in 1:6) {
    if (simul.Result.HSTest[i,1]==simul.Result.HSTest[i,10]) {
        simul.Result.HSTest[i,10]<- simul.Result.HSTest[i,9]
        simul.Result.HSTest[i,9]<-NA
    }
}
options(warn=0)
colnames(simul.Result.HSTest)<-c("nthresh","ngridTh","boot type","nboot",
    "supLM","P-value","Seconds","L $\\%$","M $\\%$","U $\\%$")
tab.simul.Result.HSTest<-xtable(simul.Result.HSTest,align=c("|l|l|r|l|r|r|r|r|r|r|r|"),
    digits=c(0,0,0,0,0,1,3,1,1,1,1))
## chunk number 94: tab.simul.Result.HSTest
print(tab.simul.Result.HSTest,hline.after=c(-1,0,4,nrow(tab.simul.Result.HSTest)),
    floating=FALSE,quote=FALSE,include.colnames=TRUE,include.rownames=FALSE,
    sanitize.text.function = function(x){x})
## chunk number 95: make.setarTest.ssim
ssim.setarTest1vs<-setarTest(ssim,m=3,nboot=1000,trim=0.05,
    test="1vs",hpc="none",check=FALSE)
ssim.setarTest2vs3<-setarTest(ssim,m=3,nboot=1000,trim=0.05,
    test="2vs3",hpc="none",check=FALSE)
## chunk number 96: make.setarTest.ssim.table
ssim.setarTest<-c(round(ssim.setarTest1vs$Ftests[1,1],digits=1),
    round(ssim.setarTest1vs$CriticalValBoot[1,1:4],digits=1),
    round(ssim.setarTest1vs$PvalBoot[1],digits=3))
ssim.setarTest<-rbind(ssim.setarTest,
    c(round(ssim.setarTest1vs$Ftests[1,2],digits=1),
    round(ssim.setarTest1vs$CriticalValBoot[2,1:4],digits=1),
    round(ssim.setarTest1vs$PvalBoot[2],digits=3)))
ssim.setarTest<-rbind(ssim.setarTest,
    c(round(ssim.setarTest2vs3$Ftests[1,3],digits=1),
    round(ssim.setarTest2vs3$CriticalValBoot[1,1:4],digits=1),
    round(ssim.setarTest2vs3$PvalBoot[1],digits=3)))
colnames(ssim.setarTest)<-c("F-test","90\\%","95\\%",
    "97.5\\%","99\\%","P-value")
rownames(ssim.setarTest)<-c("1vs2","1vs3","2vs3")
tab.ssim.setarTest<-xtable(ssim.setarTest,align=c("|l|l|l|l|l|l|l|"),
    digits=c(0,1,1,1,1,1,3))
## chunk number 97: tab.ssim.setarTest
print(tab.ssim.setarTest,hline.after=c(-1,0,nrow(tab.ssim.setarTest)),
    floating=FALSE,quote=FALSE,include.colnames=TRUE,include.rownames=TRUE,
    sanitize.text.function = function(x){x})
## chunk number 98: fig.ssim.setarTest
par(mfrow=c(2,1))
plot(ssim.setarTest1vs)
plot(ssim.setarTest2vs3)
```