Advanced Methods for Estimating the Probability of Informed Trading

Dissertation

zur Erlangung des Grades
eines Doktors der Wirtschaftswissenschaft
(Doctoris rerum politicarum)
der Fakultät für Empirische Humanwissenschaften und Wirtschaftswissenschaft der Universität des Saarlandes

vorgelegt von
Dipl.-Kfm. Andreas Recktenwald

Saarbrücken 2019

Tag der Disputation:	14. August 2019
Dekan:	Univ.-Prof. Dr. Stefan Strohmeier
Erster Berichterstatter:	Univ.-Prof. a.D. Dr. Ralph Friedmann
Zweiter Berichterstatter:	PD Dr. Stefan Klößner

Contents

List of Figures

List of Tables

List of Code Chunks

Important Abbreviations

(A)LACD (Asymmetric) logarithmic autoregressive conditional duration

BMW Bayerische Motorenwerke AG

cdf cumulative distribution function

CON Continental AG

DAI Daimler AG

e. g. for example (Latin: exempli gratia)

EHO Static model by Easley, Hvidkjaer, and O'Hara (2002)

EKOP Static model by Easley, Kiefer, O'Hara, and Paperman (1996)

F Ford Motor Company

FPE Floating point exception

GM General Motors Company

HMC Honda Motor Co. Ltd

i. e. that is (Latin: id est)

iid independently and identically distributed

JCI Johnson Controls International

MLE maximum-likelihood estimation

NYSE New York Stock Exchange

pdf probability density function

PIN-HMM Our new dynamic approach utilizing hidden Markov chains

PIN-ALACD Dynamic approach by Tay, Ting, Tse, and Warachka (2009)

pmf probability mass function

TM Toyota Motor Corp.

VOW Volkswagen AG

Xetra Electronic trading system operated by Frankfurt Stock Exchange

Important Symbols

General

$\mathscr{B}$	Bad-news state
$\text{card}(\cdot)$	Cardinality of a set
d	Index for trading days
D	Total number of trading days
$\exp(\cdot)$	Exponential function
$\mathbb{E}(\cdot)$	Expectation operator
$F(\cdot)$	Cumulative distribution function
$f(\cdot)$	Probability density/mass function
$\mathscr{G}$	Good-news state
$\mathscr{L}(\cdot)$	(Log-) Likelihood function
$\log(\cdot)$	Natural logarithm function
$T_{c,m}$	Market-specific official closing (regular trading)
$T_{o,m}$	Market-specific official opening (regular trading)
$\mathscr{M}$	Tuples of daily aggregated buys and sells
$\mathbb{N}_0$	Natural numbers including 0
$\mathscr{N}$	No-news state
OI_d	Order imbalance on trading day d
PIN	Probability of informed trading
Q	Set of possible states for trading days
q	Index for condition of trading day
$\mathbb{R}$	Real numbers
$\mathbb{R}_{>0}$	Positive real numbers
$\mathbb{R}_{\geq 0}$	Positive real numbers including zero
$S(\cdot)$	Survivor function

Static Models

α	Probability of a news-event
B_d	Aggregated number of buys on trading day d
δ	Probability of bad-news given a news-event
ϵ	Intensity of uninformed buys and sells in the EKOP model
ϵ_b	Intensity of uninformed buys in the EHO model
ϵ_s	Intensity of uninformed sells in the EHO model
μ	Intensity of informed buys and sells in the EKOP and EHO model
S_d	Aggregated number of sells on trading day d
θ	Vector of model parameters in the EKOP model
θ_{ext}	Vector of model parameters in the EHO model

Dynamic Models

α_j	Autoregressive parameter in ALACD specifications ($j \in \{-1, 1\}$)
$\alpha_q(d)$	Forward term for being in state q on trading day d (Forward-Backward algorithm)
$a_{m,n}$	Transition probability to switch from state m to state n
$\beta_q(d)$	Backward term for being in state q on trading day d (Forward-Backward probability)
β_j	Coefficient of lagged logarithmic durations in ALACD specifications ($j \in \{-1, 1\}$)
δ_i	Parameters determining the condition of trading days in the approach by Tay, Ting, Tse, and Warachka (2009) ($i \in \{1, \dots, 4\}$).
$\mathcal{F}_{i-1,d}$	Information set upon the $(i-1)th$ transaction on trading day d
$\mathcal{I}_d$	Total number of transactions on trading day d
$\lambda_{T^q_{j,i,d}}(\cdot)$	Intensity function of $T^q_{j,i,d}$
$\Lambda_{T^q_{j,i,d}}(\cdot)$	Cumulative intensity function of $T^q_{j,i,d}$
$\nu_{j,j}$, $\nu_{j,-j}$	Intercepts in ALACD specifications ($j \in \{-1, 1\}$)
$\mathcal{O}_d$	Set of observations for trading day d
$\boldsymbol{\mathcal{O}}$	Sequence of observation sets
$\boldsymbol{\omega}$	Stationary distribution of the Markov Chain for conditions of trading days
$\pi_{q,d}$	Probability that trading day d resides in state q

θ_{dyn}	Vector of model parameters
$T^{q}_{j,i,d}$	Random duration of i-th transaction of trade direction j on trading day d with state q
$x_{i,d}$	Observed duration of i-th transaction on trading day d
$y^{q}_{i,d}$	Random trade direction of i-th transaction on trading day d with condition q
ζ_j	Coefficient of lagged signed logarithmic volume in ALACD specifications ($j \in \{-1, 1\}$)

1 Introduction

For a long time, financial markets were only analyzed and investigated from a macroeconomic point of view. Actual trading activities, like the arrival of buyer- and seller-initiated transactions, the pricing process or the bid-ask spread were treated as a *black box*. However, in the last decades, the focus of research has tended more and more to the microstructure of markets. The strong interest in the trading process, in which a market maker sets bid and ask prices according to his beliefs and interacts with market attendees by buying from and selling equities to them, has led to many contributions (see Hasbrouck 1988). One can differentiate between two main directions of research which are separated from each other. According to Hasbrouck (1988), publications can be grouped into the topics of *inventory-control* and *information-asymmetry (adverse selection)*.

The first deals with the inventory of market markers, since this aspect of trading is likely to impact security prices and liquidity (see Madhavan 2000). When market makers are holding more (less) equities than desired, hence their inventory is too great (small), prices are reduced (increased) to enhance trading activities and to ensure liquidity. These models assume that market makers gain profits by buying shares at the bid and selling them at the ask price to compensate their inventory risk (see Manaster and Mann 1996). However, since this work presents and analyzes several approaches to calculate the probability of informed trading (PIN) which, in general, can be specified as the expected proportion information-based trading occupies of total trading, we concentrate on models dealing with information-asymmetry.

Adverse selection models analyze trading environments in which market makers and market attendees do not exhibit identical levels of information (see Manaster and Mann 1996). These models drop the efficient market hypothesis which states that information is universally shared (e.g., see Roll 1984). General assumption is that market participants can be split into two disjoint groups, namely uninformed and informed traders.[1] Uninformed traders are active on every trading day and do not have access to private information. Informed traders have knowledge of nonpublic information which gives them an advantage over market makers as well as noise traders (see Copeland and Galai 1983). Hence, they can better anticipate the future security price.

One of the first papers dealing with the market constellation of a market maker and the two

[1] We will use the terms *uninformed traders* and *noise traders* interchangeable for the remainder of this work.

groups of market participants, informed and uninformed, is the one by Bagehot (1971)[2]. In this work, the competition between the market maker and attendees with knowledge of private, price-relevant news is called a *"heads I win, tails you lose"* game in favor of the latter. The market maker always loses against this group of traders but tries to compensate his losses with bid-ask spreads (see Glosten and Milgrom 1985, Easley and O'Hara 1987).

If private information hits the trading process, revisions in beliefs about asset values do not only incorporate the arrival of new information. Traders with private information exploit their advantage and buy or sell securities as soon as their actual prices are below or above the true ones. This leads to an increased amount of transactions with higher order size in the corresponding trade direction. Hence, signed order flow plays a prominent role in this field of research (see Madhavan 2000).

Capturing the extent of private information of transactions is essential for analyzing the trading process. Hence, the probability of informed trading plays an important role in models which address information-asymmetry. By modeling the trading process on markets with heterogeneously informed groups of traders and the market maker's adaptations of bid and ask prices according to his beliefs, transaction data can be used to make inferences about PIN (see Daley and Vere-Jones 2010).

The basic model for the probability of informed trading was established by Easley, Kiefer, O'Hara, and Paperman (1996) (EKOP). It constitutes the baseline framework for several further approaches which adapt its general assumptions. While the EKOP model only differentiates between uninformed and informed transactions, one natural extension is to split the noise trading rate to take account of different arrival rates of buyer- and seller-initiated orders (EHO model, see Easley, Hvidkjaer, and O'Hara 2002).[3] Arrival rates of buyer- and seller-initiated orders of uninformed traders are assumed to follow homogeneous Poisson processes. Informed traders initiate buy-orders only on good-news days and sell-orders only on bad-news days, where the order arrivals also follows a homogeneous Poisson process. The state of a trading day, either no-news, good-news or bad-news, is realized according to respective constant probabilities. We assign the EKOP and EHO model to the class of *static models* with constant model parameters and constant probability of informed trading. Hence, the probability of informed trading does not change in the time span under consideration.

The PIN measure in models, which we label with the attribute *static*, represents the probability that market participants which possess private information enter the market at the beginning of a trading day. In other words, the probability of informed trading in these models can be interpreted as the risk of the market maker that he has to face a counterpart in the trading process which is even better informed than he is himself.

[2] Walter Bagehot (1826-1877) was an economist, journalist, and editor of the journal The Economist. Following The Economist's weekly commentary title "Bagehot", Jack Treynar used Bagehot as pen name; his 1971 analysis was reprinted under his real name in the Financial Analysts Journal 1995.

[3] Consequently, a next step would be to allow for different rates of informed buys (in case of good-news) and informed sells (in case of bad-news). However, to the best of our knowledge, this has not been done in the literature so far.

Research in the area of static models is still very active. Recently, new model extensions have been published. Pöppe, Aitken, Schiereck, and Wiegand (2016) propose a modification of the baseline EKOP model which allows to estimate the probability of informed trading even for short horizons. A variation of the standard PIN measure called *volume-synchronized probability of informed trading (VPIN)* was established by Easley, López de Prado, and O'Hara (2012). It is applicable to high-frequency trading by splitting trading days by volume time. Hence, days consist of short-term intraday volume buckets with constant amount of total size of transactions.

The model by Easley, Engle, O'Hara, and Wu (2008) is the first which allows for time-varying arrival rates of buys and sells and therefore time-varying estimates of the probability of informed trading. However, parameters determining the state probabilities of trading days are still not allowed to vary over time. Hidden Markov models were introduced for the first time in the context of PIN by Yin and Zhao (2015). One can think of this model as an intermediate step between the static approaches and the more advanced and complex settings which allow to calculate varying estimates of PIN even for very short intraday intervals. We label the latter as *dynamic models*.

While the EKOP model represents the baseline in terms of static models, the model by Tay, Ting, Tse, and Warachka (2009) is its counterpart in the field of dynamic models. The general assumptions from the EKOP model are still valid, but the modeling of the trading process is much more advanced. High-frequency data is utilized in place of the aggregated number of buys and sells in a predefined time interval[4], interactions between consecutive buys and sells are allowed and the trade durations and transaction sizes are taken into account. An extension of the dynamic setup was introduced by Preve and Tse (2013) who generalized the possible types of trading days. Instead of only no-news, good-news and bad-news, they refine each state and distinguish between order-flow shock and non-order-flow shock conditions. Hence, in total the authors double the total number of possible different states of trading days. This constellation of possible conditions had already been established in terms of static models by Duarte and Young (2009).

In the dynamic approaches, the probability of informed trading is modeled as the unobservable realized proportion of transactions insiders occupy of the total number of buys and sells in a unit time interval. Hence, the perspective and interpretation in static and dynamic models in terms of the PIN measure are different. Nevertheless, we go along with the recent literature in this field of research and do not introduce different symbols for the probability of informed trading, but use PIN for both types of models.

Dynamic models comprise a very young field of research for the probability of informed trading with a very limited number of empirical contributions. Hence, empirical applications of static approaches are still predominant nowadays. Just to name a few, Henry (2006) investigates the relationship between short selling and information-based trading, while the connection between investor protection, adverse selection and PIN is analyzed by Brockman and Chung (2008). In which way the probability of informed trading influences herding behavior is studied

[4] In literature, typically one trading day is defined as unit time interval.

in the work by Zhou and Lai (2009). Aslan, Easley, Hvidkjaer, and O'Hara (2011) employ PIN to investigate the linkage of microstructure, accounting and asset pricing and intend to determine firms which have high information risk. Seasonality of PIN estimates are examined in the work by Kang (2010). Additionally, various papers link the probability of informed trading to illiquidity measures (e.g., Duarte and Young (2009) and Li, Wang, Wu, and He (2009)), and bid-ask spreads (e.g., Lei and Wu (2005) and Chung and Li (2003)). Whether PIN is only a substitute for liquidity or it really captures the asymmetry of information and therefore reflects information-based trading is currently heavily discussed in the literature (e.g., see Aktas, de Bodt, Declerck, and van Oppens 2007, Duarte and Young 2009, Mohanram and Rajgopal 2009, Fuller, Van Ness, and Van Ness 2010). The VPIN measure gained attention for predicting the *flash crash* on May 6, 2010 (e.g., see Andersen and Bondarenko 2014).[5]

At the beginning, empirical applications in the area of information-asymmetry were limited and only data for short periods were utilized (e.g., see Glosten and Harris 1988). Estimations were computationally expensive and very time-consuming. Since computing power increased dramatically over the past decades, the number of empirical works has grown and it is nowadays even possible to apply models to high-frequency data over long periods with reasonable effort.

Due to the widespread usage of the static PIN measure in empirical literature, many researchers focused on analyzing its technical (computational) properties. Recently, several papers were published proposing improvements in the estimation procedure of model parameters and the probability of informed trading. Typically, parameter estimation is conducted with the maximum-likelihood method, which tries to find the parameter values which maximize the corresponding (log-)likelihood function. Since there are no closed-form solutions for this task in the static PIN models, iterative optimization methods need to be utilized. However, there is the chance that these methods do not converge, which means that they do not find a maximum at all, or that they land in a local instead of a global maximum. Therefore, it is crucial to initialize the iterative methods with appropriate starting values.

The original factorizations of likelihood functions in static PIN models are very inefficient in terms of stability and execution time. Furthermore, due to the inability to handle moderate or large values of aggregated daily buyer- and seller-initiated transactions, with the original likelihood formulation PIN can only be estimated for ancient trading data or very infrequently traded stocks. Easley, Hvidkjaer, and O'Hara (2010) present a more robust formulation of the likelihood function which reduces the occurrence of over- and underflow errors for moderately traded equities. The most recent likelihood factorization for the PIN framework assuming static arrival rates by Lin and Ke (2011) can even handle daily buys and sells data of very heavily traded stocks. In addition, Lin and Ke (2011) showed in their simulation study that if the factorization by Easley, Hvidkjaer, and O'Hara (2010) is used in numerical maximizations, one is confronted with floating-point exception (i.e., over- and underflow errors). Moreover, the authors state that estimates of the probability of informed trading are downward-biased, especially for frequently traded stocks.

[5] We will not delve any deeper into this direction of research in this work.

As mentioned above, results and convergence of optimizations depend to a huge degree on the chosen method for the generation of starting values. Yan and Zhang (2012), Gan, Chun, and Johnstone (2015) and Ersan and Alıcı (2016) study the generation of initial values for maximizations in static PIN models. A brute force grid search technique which delivers several sets of starting values is established by Yan and Zhang (2012). Despite its simplicity this method is very time-consuming. Both works, Gan, Chun, and Johnstone (2015) and Ersan and Alıcı (2016), offer more advanced methods by harnessing hierarchical agglomerative clustering (HAC), which is a bottom-up clustering technique starting with one cluster for each observation and sequentially merging them to bigger ones[6], to determine initial choices for the model parameters. While the former delivers only one set of initial values, the latter returns several sets but still only a small fraction of the amount delivered by grid search technique. Hence, both recent and more sophisticated algorithms for the creation of starting values improve optimization procedures at least in terms of computing time.

This thesis includes contributions to the literature of both types of PIN models, static and dynamic. We introduce and empirically estimate posterior probabilities of the conditions of trading days in static models which assume constant parameters. With this extension, we introduce the understanding of the probability of informed trading in the dynamic approaches to the static framework. Furthermore, we are the first to establish a confidence interval for the probability of informed trading.

While these are extensions to yet existing models, we propose a new dynamic approach which generalizes one of the central ideas of previous PIN models. Our new model for the probability of informed trading is based on and extends the work by Tay, Ting, Tse, and Warachka (2009). By utilizing hidden Markov chains for the modeling of trading days' conditions, we are able to model dependencies between consecutive days, but also include the independence of states as a special case. Taking account of the time-consuming diffusion of information into the market, allowing for dependencies is very reasonable, especially in terms of short intraday periods where the condition of the current period highly likely may be influenced by its predecessor. In addition, we relax the common assumption of Poisson processes for the arrivals of buys and sells and, conditional on the past process trajectory, allow the usage of any distribution with positive support for the interarrival times of buy and sell orders, while we concentrate on Weibull distributions. Overall, we provide a general framework which can easily be extended in several aspects. For example, the constellation of trading days' conditions by Preve and Tse (2013) could be incorporated or more generalized distributions could be utilized for waiting times of buys and sells. We are not the first to incorporate hidden Markov chains in PIN models, but our modeling differs from the intention in Yin and Zhao (2015) where the hidden states of the market are modeled as tuples of aggregated numbers of daily buys and sells, instead of labeling the state of a trading day as either no-, good- or bad-news.

This dissertation is structured as follows:

Chapter 2 presents the general framework of all PIN models discussed in this work. Proposed assumptions are valid for static as well as dynamic models.

[6]For a more detailed description of hierarchical agglomerative clustering see the beginning of section 4.3.2.

Chapter 3 gives in-depth explanations of the theoretical setting in the most prominent static models for the probability of informed trading, namely the approaches by Easley, Kiefer, O'Hara, and Paperman (1996) and Easley, Hvidkjaer, and O'Hara (2002). The last section of this chapter introduces posterior probabilities of conditions of trading days in the context of the static PIN models.

The fourth chapter concentrates on the numerical issues related with the models presented in chapter 3. A brief introduction to the areas of floating-point representation and over- and underflow errors is included. Factorizations of likelihood functions by Easley, Hvidkjaer, and O'Hara (2010) and Lin and Ke (2011) are discussed and analyzed by applying them to different synthetic datasets. Algorithms for initial values by Yan and Zhang (2012), Gan, Chun, and Johnstone (2015) and Ersan and Alıcı (2016) are presented and the quality of starting values provided by the different techniques is compared with the help of a simulation study. At the end of the chapter, confidence intervals for the probability of informed trading are introduced.

The next chapter is dedicated to the **pinbasic** R package which offers fast and stable implementations of the theoretical constructs in the context of static models previously discussed, as well as functions for visualizations of estimation results and posterior probabilities. Along with the source code, we give detailed information about the usage of corresponding functions in the **pinbasic** package.

The framework of dynamic models which use high-frequency transaction data to estimate the probability of informed trading is founded in chapter 6. We explain the theory of the approach by Tay, Ting, Tse, and Warachka (2009) and our new model utilizing hidden Markov chains for the sequence of trading days' conditions in detail. An intensive discussion of the much more advanced and complex method of calculating the probability of informed trading is given. In addition, we provide an introduction to (hidden) Markov chains and the forward-backward algorithm[7] which is used for optimization purposes in our new model.

Source code of all functions which are involved in the estimation of both dynamic models is displayed in chapter 7. Each function comes with a detailed explanation of how it integrates in the workflow for optimizations.

A description of our raw datasource used for estimations can be found in chapter 8. We show in detail which steps are necessary to achieve prepared datasets for optimizations. Since all symbols under consideration belong to the automobile industry, either as manufacturer or supplier, time ranges of central interest in our analyses are placed around the beginnings of scrappage programs in the United States and Germany.

Source code of implementations and explanations about the usage of functions concerning data preparation steps can be found in the consecutive chapter.

Chapter 10 covers empirical applications of the static PIN models utilizing data of equities over a range of four years (2007 - 2010), for nine stocks either listed on New York Stock Exchange (NYSE) or Xetra. This chapter concludes with the results for posterior probabilities of trading

[7] A detailed description of the forward-backward algorithm can be found in section 6.2.2.

days' states and confidence intervals for PIN whose theory is explained in chapter 4. Results of both newly introduced extensions to the literature of the probability of informed trading are presented for a subset of symbols and time ranges.

The eleventh chapter presents empirical applications of the dynamic models which are covered with this work. We also switch from the common unit time interval of trading days to very short intraday periods. The behavior of the probability of informed trading is analyzed for very special trading days. We choose *General Motors* from the NYSE-listed equities and investigate the last two trading days before its delisting after June 1, 2009. From the stocks listed on Xetra we pick *Volkswagen AG* and take a closer look on trading days around the price bubble in October 2008.

We will conclude the introduction with some words about the computational aspects concerning this work:

All calculations for which results are presented were conducted using the statistical programming language R. Some parts which are critical in terms of execution time have been implemented in C++ with the help of the **Rcpp** and **RcppArmadillo** packages (see Eddelbuettel and François 2011, Eddelbuettel and Sanderson 2014). Hence, all code chunks display source code or function calls which belong to the R or C++ programming language.

At the time of writing the statistical programming language R is available in version 3.5.2. Additional packages used for computations are mentioned in the text and corresponding authors are listed in the references.

2 General Framework of Models for the Probability of Informed Trading

We distinguish between two different types of models for estimating the probability of informed trading, *static* and *dynamic*. Static models assume parameters to be constant over the whole range spanned by the underlying data, whereas they are supposed to be time-varying in dynamic models. Furthermore, the PIN measure has a different intention in each model type. In the static models, the probability of informed trading is the probability that insiders with access to superior information enter the market, evaluated before the beginning of a trading period, while it is the unobservable realized fraction of insider trading in the dynamic approaches. Although the two groups of models have a different view of the probability of informed trading, an a priori probability in the static ones and an a posteriori in the dynamic approaches, we go with the literature and use the same notation in both cases.

In the sequential microstructure models for estimating the probability of informed trading, the exchange of equities takes place over $d = 1, \dots, D$ trading periods. The unit time interval is usually specified as one trading day for both model types, but it can also be an intraday interval of arbitrary length for dynamic models. Since we set trading days as reference for the majority of analyses in dynamic setup but also deliver intraday estimates of the probability of informed trading, the terms *trading periods* and *trading days* are used interchangeably.[8]

No market activities are permitted in which a risk-neutral and competitive market maker is not involved. Hence, market participants can only trade with the market maker and it is not possible for them to execute transactions among each other. The market maker determines and updates bid and ask prices utilizing the information he gathered so far for a trading day. Trading with the market maker is possible at every timestamp t during regular market hours starting at $T_{o,m}$ and ending at $T_{c,m}$: $t \in \left[T_{o,m}, T_{c,m}\right]$ with finite $T_{c,m}$, where the index m represents the specific marketplace under consideration. The beginning of official trading may vary depending on the chosen bourse m. For instance, the New York Stock Exchange starts

[8] Dynamic models introduced by Tay, Ting, Tse, and Warachka (2009) and Preve and Tse (2013) also specify the unit time interval as one trading day.

regular trading at 9:30 a.m., whereas the German electronic trading system XETRA opens earlier at 9:00 am. Likewise, the upper bound $T_{c,m}$ of the official trading interval may also vary according to the marketplace under consideration.

Each trading day can reside in one of three possible states of the set $Q = \{\mathcal{N}, \mathcal{G}, \mathcal{B}\}$.[9] The elements of the set Q, which represent the conditions of trading days, are no-news ($\mathcal{N}$), good-news ($\mathcal{G}$) and bad-news ($\mathcal{B}$). They are unobservable and determined by nature before the beginning of each trading period. Trading periods on which private information influence the market activities are called *information events*.

Market participants are split in two disjoint groups, informed and uninformed traders. Traders holding private information are solely active on information events. In addition, they are assumed to be risk neutral and competitive. They buy (sell) if positive (negative) signals hit the market, which is the case on good-news (bad-news) trading days, as visualized in figure 2.1. The contrary group of traders, the uninformed market attendees, are active on every trading day for various reasons (e.g., diversification, liquidity reasons).

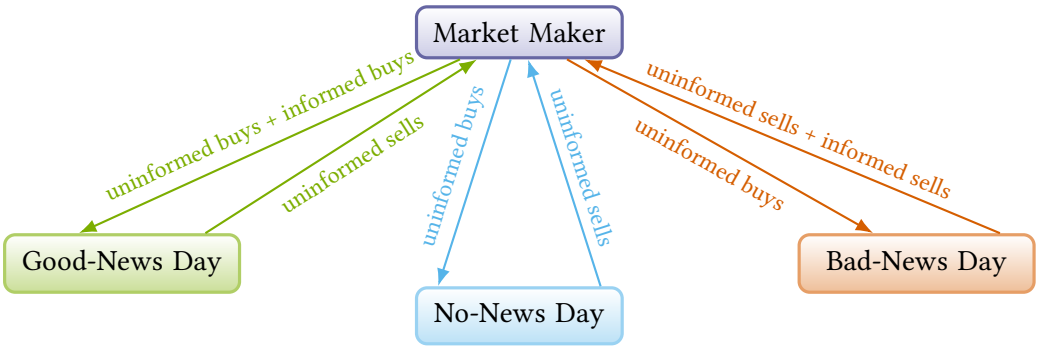

Figure 2.1: Trading activities on trading days in the static PIN framework with respect to their conditions.

In general, the probability of informed trading (PIN) can be defined as the relation of the expected number of transactions due to private information to the expected total number of trades,

$$\text{PIN} = \frac{\text{Expected number of information-based transactions}}{\text{Expected total number of transactions}}. \tag{2.1}$$

Although both types of models, static and dynamic, share the same structure of the formula to calculate the probability of informed trading, the PIN variable has varying intentions in the different approaches.

[9] Duarte and Young (2009) introduced an extended setting for static models. The number of possible conditions for trading days is doubled due to order-flow shocks which can influence each state. Hence, there are no-news, good-news and bad-news trading days, each with and without shocks. This setup was adopted by Preve and Tse (2013) in terms of dynamic models.

As mentioned before, in static models for the probability of informed trading PIN is interpreted as the probability that insiders hit the market at the beginning of trading periods and is constant over the whole range of the underlying data. It also reflects the market maker's initial beliefs about conditions of trading periods, as explained in Easley, Kiefer, O'Hara, and Paperman (1996, p.1421). Therefore the expected number of information-based transactions and the expected total number of transactions in (2.1) are based on a priori information at the beginning of each trading period.

For dynamic models, which aim to estimate the (latent) realized fraction of insider trading, PIN varies over time and is typically reported on, but not restricted to, a daily basis.[10] In contrast to static models, corresponding (observed) high-frequency trading data in each trading period is utilized to calculate the expectations in the numerator and denominator in equation (2.1).

[10] See section 11.5 for analyses of intraday estimates of the probability of informed trading.

3 Static Models

This chapter deals with static models for estimating the probability of informed trading. The word *static* takes into account that these models assume constant arrival rates for daily buys and sells. Additionally, probability parameters are also assumed to be constant over the whole range spanned by the data. Therefore these models do not enable to estimate the probability of informed trading on a daily basis.

The static models have several assumptions in common. Firstly, the sequence of trading days is assumed to be discrete and independent, whereas the time during a trading day is supposed to be continuous. Conditions of trading days are not observable and determined by nature before official market opening. *Information events*, days on which private, price-relevant information enters the market, occur with probability α. This is good information with probability $1 - \delta$ and news with negative direction with probability δ. Hence, unconditional probabilities of no-news, good-news and bad-news conditions are given by:

$$\Pr(\mathscr{N}) = 1 - \alpha \tag{3.1a}$$

$$\Pr(\mathscr{G}) = \alpha(1 - \delta) \tag{3.1b}$$

$$\Pr(\mathscr{B}) = \alpha\delta. \tag{3.1c}$$

Furthermore, conditional on a given information state of a trading day, buys and sells are supposed to follow latent independent homogeneous Poisson processes with constant intensities in a unit time interval. A Poisson process is a point process which is often defined on the positive line. According to Daley and Vere-Jones (2003), "we shall understand by a point process some method of randomly allocating points to the real line." However, in terms of arrivals of buys and sells, we assume the Poisson processes in static PIN models to be defined on the positive half-line.

Before we give a formal definition of Poisson processes, we need to introduce another class of stochastic processes, namely *counting processes*, which our definition of Poisson processes is based on. According to Ross (1996), a stochastic process $\{N(t), t \geq 0\}$ is called a counting process if it represents the total number of events that have occurred up to time t and it satisfies:

1. $N(t) \geq 0$.
2. $N(t)$ is integer-valued.
3. if $s < t$ then $N(s) \leq N(t)$.

4. For $s < t$, $N(t) - N(s)$ equals the number of events that have occurred in the interval $(s, t]$.

If the number of events that occur in disjoint time intervals are independent, the counting process exhibits *independent increments*. Furthermore, a counting process has stationary increments if the distribution of the number of events that happen in any time interval only depend on its length.[11]

Hence a formal definition of homogeneous Poisson processes is given by (see Ross 1996):

Definition 3.1. *The counting process $\{N(t), t \geq 0\}$ is said to be a homogeneous Poisson Process having rate λ, $\lambda > 0$, if:*

1. $N(0) = 0$.
2. *The process has independent increments.*
3. *The number of events in any interval of length t is Poisson[12] distributed with mean λt. That is, for all $s, t \geq 0$,*

$$\Pr(N(t+s) - N(s) = n) = \exp(-\lambda t)\frac{(\lambda t)^n}{n!}, \quad n = 0, 1, 2, \ldots.$$

It follows from the third property of definition 3.1 that homogeneous Poisson processes have stationary increments and that $\mathbb{E}(N(t)) = \lambda t$. Let $X_n, n \geq 1$ denote the time between the n-th and $(n-1)$-th event. It can be shown that the waiting times or interarrival times X_n are independently and identically distributed (iid) exponentially distributed (for the two just mentioned properties of Poisson processes see Ross 1996).[13]

In terms of models for the probability of informed trading, an event represents either the arrival of a buy or sell. Therefore, we will use the terms *transaction* and *event* interchangeable for the remainder of this thesis.

[11] Both properties are described in Ross (1996).

[12] The discrete Poisson distribution is defined by the following cumulative distribution function (cdf) and probability mass function (pmf) (e.g., see Johnson, Kemp, and Kotz 2005, p. 156):

$$\text{cdf: } F(\lambda; n) = \exp(-\lambda)\sum_{k=0}^{n}\frac{\lambda^k}{k!} \quad \text{and} \quad \text{pmf: } f(\lambda; n) = \exp(-\lambda)\frac{\lambda^n}{n!},$$

with the number of arrivals $n \in \mathbb{N}_0$ and the intensity parameter/rate $\lambda \in \mathbb{R}_{>0}$. Mean and variance of Poisson distributions both equal λ.

[13] The exponential distribution is a continuous distribution which has cdf and probability density function (pdf) given by (e.g., see Johnson, Kotz, and Balakrishnan 1994, p. 494):

$$\text{cdf: } F(\lambda; x) = 1 - \exp\left(-\frac{x}{\lambda}\right) \quad \text{and} \quad \text{pdf: } f(\lambda; x) = \frac{1}{\lambda}\exp\left(-\frac{x}{\lambda}\right),$$

with waiting times $x \in \mathbb{R}_{\geq 0}$ and scale parameter $\lambda \in \mathbb{R}_{>0}$.

The arrivals of transactions, which are indeed observable, can be interpreted as a merging of the latent arrivals of buys and sells, N_B and N_S, respectively. One can think of the observable arrivals of transactions N_O as the outcome of a competition between the latent Poisson processes of buys and sells which is the first to arrive, whereat the waiting times of the latent Poisson processes determine the next trade's direction. Assuming that the current waiting time of the buys' point process is less than the sells' interarrival time, the observed transaction will be buyer-initiated and N_B increases by 1 as well as N_O, whereas N_S remains unchanged, as shown in figure 3.1.[14] After observing a transaction the waiting times of both latent point processes are reset and the race of buys and sells process begins anew.

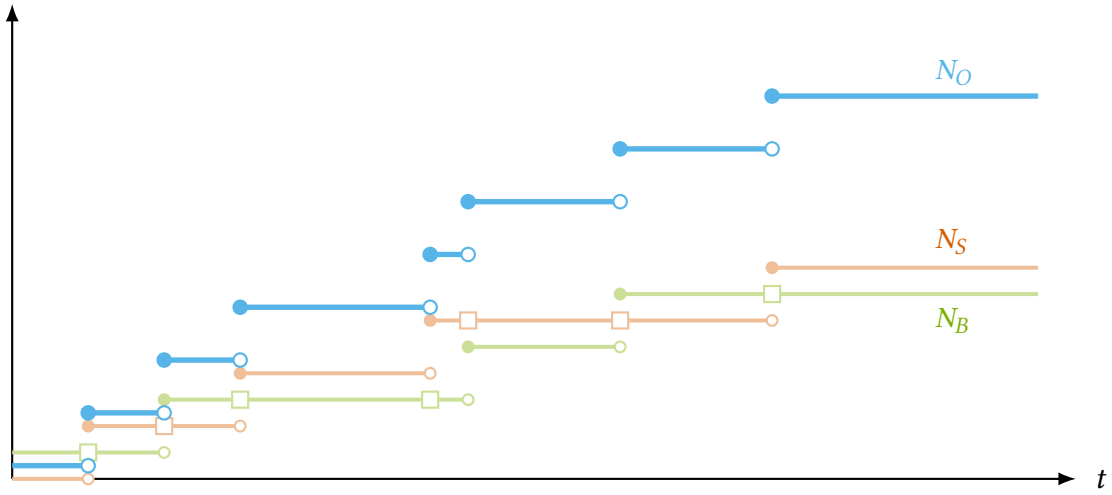

Figure 3.1: Exemplary paths of the latent Poisson processes of buys and sells and the observable (merged) Poisson process of transactions. The green-(red-)colored path represents the latent counting measure for buys (sells). The counting measure for the observable point process is displayed by the blue path. The combination of non-filled and filled circles are drawn whenever a trade arrived, which means that either the counting measure for buys or sells makes a jump. The non-filled squares are drawn when the waiting time of the point process is reset which involves a jump of the opposite trade direction. N_O increases by 1 every time a trade is observed, N_B and N_S only if the competition of waiting times is won.

The following theorem states that the merging of two homogeneous independent Poisson processes again yields a homogeneous Poisson process.[15]

Theorem 3.1. *(a) Suppose that* $\{N_1(t), t \geq 0\}$ *and* $\{N_2(t), t \geq 0\}$ *are independent Poisson processes with respective rates* λ_1 *and* λ_2*, where the process* $\{N_i(t), i = 1, 2\}$ *corresponds to type* i *arrivals. Let* $N(t) = N_1(t) + N_2(t)$, $t \geq 0$. *Then the merged process* $\{N(t), t \geq 0\}$ *is a Poisson*

[14] For most exchanges no data is available about the direction of transactions. Due to this reason algorithms like *Lee & Ready* are used to try to detect if a transaction is buyer- or seller-initiated (see Lee and Ready (1991)). More on this topic can be found in chapter 8.

[15] The theorem together with its proof can be found in Tijms (2003).

process with rate $\lambda = \lambda_1 + \lambda_2$. Denoting by Z_k the interarrival time between the $(k-1)$-th and k-th arrival in the merged process and letting $I_k = i$ if the k-th arrival in the merged process is a type i arrival, then for any $k = 1, 2, \ldots$,

$$\Pr(I_k = i \mid Z_k = t) = \frac{\lambda_i}{\lambda_1 + \lambda_2}, \quad i = 1, 2,$$

independently of t.

(b) Let $\{N(t), t \geq 0\}$ be a Poisson process with rate λ. Suppose that each arrival of the process is classified as being a type 1 arrival or type 2 arrival with respective probabilities p_1 and p_2, independently of all other arrivals. Let $N_i(t)$ be the number of type i arrivals up to time t. Then $\{N_1(t)\}$ and $\{N_2(t)\}$ are two independent Poisson processes having respective rates λp_1 and λp_2.

Using theorem 3.1 we can write down the probabilities of observing a buyer- or seller-initiated transaction for all three possible states of a trading day d. For a no-news day the probabilities of observing a buy or sell at each point in time can be written incorporating the corresponding intensities of the point processes of buys and sells,

$$\Pr\big(I_{k,d} = \text{Buy} \mid \mathcal{N}\big) = \frac{\lambda_{\text{uninf. Buys}}}{\lambda_{\text{uninf. Buys}} + \lambda_{\text{uninf. Sells}}} \tag{3.2}$$

$$\Pr\big(I_{k,d} = \text{Sell} \mid \mathcal{N}\big) = \frac{\lambda_{\text{uninf. Sells}}}{\lambda_{\text{uninf. Buys}} + \lambda_{\text{uninf. Sells}}}, \tag{3.3}$$

where $\lambda_{\text{uninf. Buys}}$ and $\lambda_{\text{uninf. Sells}}$ denote the intensities of uninformed buys and sells, respectively.

On a good-news day informed traders enter the market but only act as buyers. Therefore the rate of the buys' Poisson process is the sum of the intensities of uninformed and informed buyers, $\lambda_{\text{uninf. Buys}}$ and $\lambda_{\text{inf. Buys}}$, respectively,

$$\Pr\big(I_{k,d} = \text{Buy} \mid \mathcal{G}\big) = \frac{\lambda_{\text{uninf. Buys}} + \lambda_{\text{inf. Buys}}}{\lambda_{\text{uninf. Buys}} + \lambda_{\text{uninf. Sells}} + \lambda_{\text{inf. Buys}}} \tag{3.4}$$

$$\Pr\big(I_{k,d} = \text{Sell} \mid \mathcal{G}\big) = \frac{\lambda_{\text{uninf. Sells}}}{\lambda_{\text{uninf. Buys}} + \lambda_{\text{uninf. Sells}} + \lambda_{\text{inf. Buys}}}. \tag{3.5}$$

Informed sellers solely occur on a bad-news day. Similar to good-news days, the rate of informed sells, $\lambda_{\text{inf. Sells}}$, must be incorporated in the probabilities,

$$\Pr\big(I_{k,d} = \text{Buy} \mid \mathcal{B}\big) = \frac{\lambda_{\text{uninf. Buys}}}{\lambda_{\text{uninf. Buys}} + \lambda_{\text{uninf. Sells}} + \lambda_{\text{inf. Sells}}} \tag{3.6}$$

$$\Pr\big(I_{k,d} = \text{Sell} \mid \mathcal{B}\big) = \frac{\lambda_{\text{uninf. Sells}} + \lambda_{\text{inf. Sells}}}{\lambda_{\text{uninf. Buys}} + \lambda_{\text{uninf. Sells}} + \lambda_{\text{inf. Sells}}}. \tag{3.7}$$

Further, according to equation (2.1), at the beginning of the trading day, the probability of informed trading is given by

$$\text{PIN} = \frac{\alpha(1-\delta)\lambda_{\text{inf. Buys}} + \alpha\delta\lambda_{\text{inf. Sells}}}{\lambda_{\text{uninf. Buys}} + \lambda_{\text{uninf. Sells}} + \alpha(1-\delta)\lambda_{\text{inf. Buys}} + \alpha\delta\lambda_{\text{inf. Sells}}}. \tag{3.8}$$

Following sections introduce the static models by Easley, Kiefer, O'Hara, and Paperman (1996) and Easley, Hvidkjaer, and O'Hara (2002) for estimating the probability of informed trading. Although the main focus of research does not lie on the PIN variable in these two papers, they are very important milestones in the literature for the probability of informed trading. In fact, they investigate information-based differences in spreads of transactions and asset returns.

3.1 Model by Easley, Kiefer, O'Hara and Paperman (EKOP)

In the model developed by Easley, Kiefer, O'Hara, and Paperman (1996) arrival rates of buys and sells are assumed to be identical. However, it distinguishes between uninformed and informed transactions. Uninformed buys and sells each appear with rate ϵ, informed buys and sells each with rate μ. Both parameters ϵ and μ are non-negative real numbers but no assumption is made about their relation in magnitude. Hence, it is possible that μ, describing the intensity of informed traders, in certain cases can stride the rate of uninformed trading ϵ.

The intensity for the observable merged Poisson process varies with the states of trading days. For no-news days the rate of arrivals equals 2ϵ and for days on which private news enter the market, either positive or negative, $2\epsilon + \mu$. The scenario tree in figure 3.2 illustrates the probabilities for the potential states a trading day in the PIN framework can reside in. In addition, the mapping of the sets of arrival rates for buys and sells to the different trading days' conditions can be read directly from the graph.

Using theorem 3.1 in combination with equations (3.2)–(3.7) we can immediately write down the probabilities of observing a buy or sell for the k-th transaction on trading day d, with

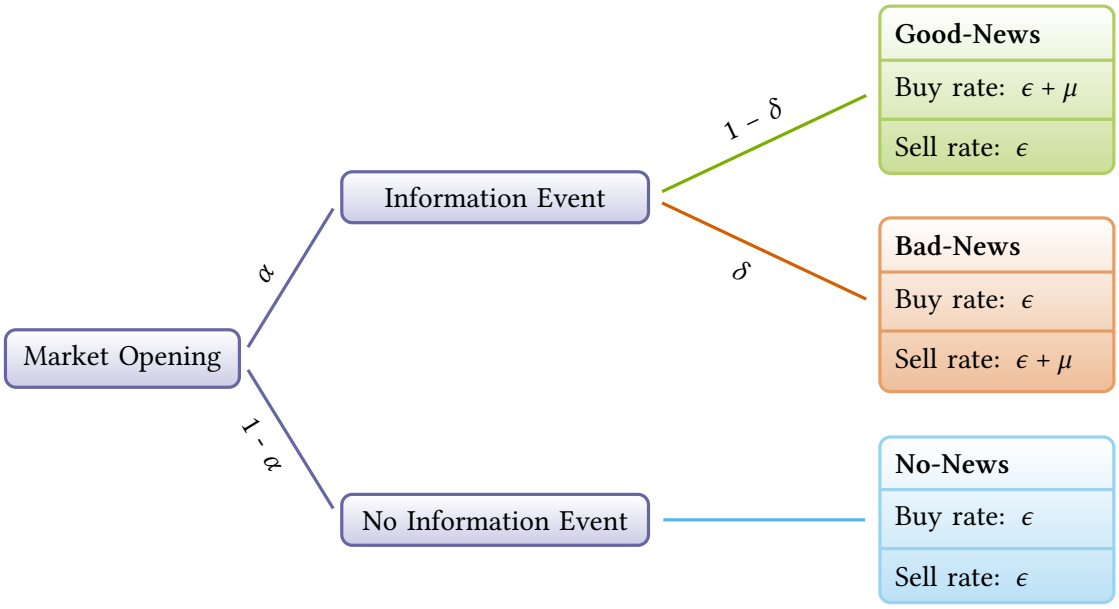

Figure 3.2: Scenario tree for the EKOP model with equal arrival rates of (un)informed buys and sells. Information events, days on which private information hit the market, occur with probability α. These private information has a negative direction with probability δ (Bad-News) and a positive direction with probability $1 - \delta$ (Good-News). On Bad-News (Good-News) days, informed traders only sell (buy). Uninformed traders buy and sell with rate ϵ, informed traders with rate μ.

respect to the condition of the trading day.

$$\Pr(I_{k,d} = \text{Buy} \mid \mathscr{N}) = \frac{\epsilon}{\epsilon + \epsilon} = \frac{1}{2} \tag{3.9a}$$

$$\Pr(I_{k,d} = \text{Sell} \mid \mathscr{N}) = \frac{\epsilon}{\epsilon + \epsilon} = \frac{1}{2} \tag{3.9b}$$

$$\Pr(I_{k,d} = \text{Buy} \mid \mathscr{G}) = \frac{\epsilon + \mu}{\epsilon + \epsilon + \mu} \tag{3.9c}$$

$$\Pr(I_{k,d} = \text{Sell} \mid \mathscr{G}) = \frac{\epsilon}{\epsilon + \epsilon + \mu} \tag{3.9d}$$

$$\Pr(I_{k,d} = \text{Buy} \mid \mathscr{B}) = \frac{\epsilon}{\epsilon + \epsilon + \mu} \tag{3.9e}$$

$$\Pr(I_{k,d} = \text{Sell} \mid \mathscr{B}) = \frac{\epsilon + \mu}{\epsilon + \epsilon + \mu}. \tag{3.9f}$$

According to equation (2.1) the probability of informed trading in the EKOP model can be

calculated as

$$\begin{aligned} \text{PIN} &= \frac{\alpha\delta\mu + \alpha(1-\delta)\mu}{2(1-\alpha)\epsilon + 2\alpha\delta\epsilon + 2\alpha(1-\delta)\epsilon + \alpha\delta\mu + \alpha(1-\delta)\mu} \\ &= \frac{\alpha\mu}{2\epsilon + \alpha\mu}, \end{aligned} \tag{3.10}$$

where PIN and the model parameters are constant over the range of the underlying data. We can think of the arrivals of buys and sells on no-news days as flipping a (fair) coin with equal probabilities for the two potential outcomes. Instead of seeing heads or tails when tossing the coin, the result will be either the arrival of buys or sells.

On information events the probability for an arrival of buys or sells increases depending on the direction of the private information. Informed buyers (sellers) enter the market if they receive a positive (negative) signal. The intensity of the point process for buys (sells) increases by the positive parameter μ, whereas the rate for sells (buys) remains on the level for no-news days. Therefore the probability of observing a transaction which is buyer- (seller-)initiated increases which causes a decrease in the probability for sells (buys).

For deriving the (log) likelihood function in the EKOP setting we can utilize the first and third property of Poisson processes in definition 3.1. Since in the PIN literature intensities of Poisson processes for buys and sells, ϵ and μ, represent expected arrivals per trading day, we can set the variables s and t to 0 and 1, respectively, to achieve an interval of length 1. Now we can adopt the equation in the third proportion of definition 3.1 for usage in the EKOP model for a trading day d,

$$\begin{aligned} \Pr(N_d(t+s) - N_d(s) = n) &= \\ \Pr(N_d(1) - N_d(0) = n) &= \\ \Pr(N_d(1) = n) &= \frac{\lambda^n}{n!}\exp(-\lambda), \end{aligned} \tag{3.11}$$

with the buys and sells Poisson processes N_d, respectively, arrival rate λ and the number of occurrences $n \in \mathbb{N}_0$ on trading day d.

The different types of trading days must be taken into account. If on trading day d no private information hit the market, it is free from information-based traders. Both point processes for buys and sells have identical intensities which equal ϵ. According to equation (3.11) the notation of the separated probabilities of observing B_d buys and S_d sells on a no-news trading day d is straightforward. Using part b) of theorem 3.1 and equations (3.9a) and (3.9b), the probability of observing a tuple of B_d buys and S_d sells can be written as

$$\underbrace{\exp(-\epsilon)\frac{\epsilon^{B_d}}{B_d!}}_{\text{Buys}}\ \underbrace{\exp(-\epsilon)\frac{\epsilon^{S_d}}{S_d!}}_{\text{Sells}}. \tag{3.12}$$

Similarly, equations (3.9c) – (3.9f) can be harnessed to get the probabilities of observing a tuple of B_d buys and S_d sells for the two remaining conditions of trading days.

On a good-news day d informed traders are active and buy equities which yields to an increase in the arrivals of buyer-initiated transactions captured by parameter μ,

$$\underbrace{\exp\big(-(\epsilon+\mu)\big)\frac{(\epsilon+\mu)^{B_d}}{B_d!}}_{\text{Buys}}\underbrace{\exp(-\epsilon)\frac{\epsilon^{S_d}}{S_d!}}_{\text{Sells}}. \tag{3.13}$$

Likewise to good-news days, the intensity of seller-initiated trades increase on a bad-news day d,

$$\underbrace{\exp(-\epsilon)\frac{\epsilon^{B_d}}{B_d!}}_{\text{Buys}}\underbrace{\exp\big(-(\epsilon+\mu)\big)\frac{(\epsilon+\mu)^{S_d}}{S_d!}}_{\text{Sells}}. \tag{3.14}$$

For most marketplaces we cannot directly identify the direction of trades. There is no flag in the data which tells us whether a trade is buyer- or seller-initiated. However, to distinguish between buys and sells is crucial in the EKOP model and in the models for the probability of informed trading in general. A well-established technique to classify transactions is the algorithm by Lee and Ready (1991), which checks if the actual price of a trade is closer to the corresponding bid or ask price.[16] Therefore, classification and aggregation of raw transactions data are essential steps in the workflow to get the daily number of buys and sells.

To receive estimates of the model parameters we utilize maximum-likelihood estimation (MLE).[17] Therefore, it is crucial to be able to calculate the (log-)likelihood function to perform numerical optimizations.

The likelihood of θ given an observed sequence of B_d buys and S_d sells on a trading day d can now be formulated, using equations (3.12–3.14), as weighted sum of these condition-specific probabilities,

$$\begin{aligned}\mathscr{L}(\theta \mid (B_d, S_d)) = {} & (1-\alpha)\left(\exp(-\epsilon)\frac{\epsilon^{B_d}}{B_d!}\exp(-\epsilon)\frac{\epsilon^{S_d}}{S_d!}\right) \\ & + \alpha(1-\delta)\left(\exp\big(-(\epsilon+\mu)\big)\frac{(\epsilon+\mu)^{B_d}}{B_d!}\exp(-\epsilon)\frac{\epsilon^{S_d}}{S_d!}\right) \\ & + \alpha\delta\left(\exp(-\epsilon)\frac{\epsilon^{B_d}}{B_d!}\exp\big(-(\epsilon+\mu)\big)\frac{(\epsilon+\mu)^{S_d}}{S_d!}\right),\end{aligned} \tag{3.15}$$

with parameter vector $\theta := \big(\alpha, \delta, \epsilon, \mu\big)$.

[16] For a detailed explanation of several classification algorithms see section 8.6.

[17] The maximum-likelihood method is used for all optimizations in this work.

Utilizing the independence of trading days, the probability of observing $\mathscr{M} = (B_d, S_d)_{d=1}^{D}$ for $d = 1, \dots, D$ trading days can be written as product of daily likelihoods,

$$\mathscr{L}(\theta \mid \mathscr{M}) = \prod_{d=1}^{D} \mathscr{L}(\theta \mid (B_d, S_d)). \tag{3.16}$$

Hence, the log-likelihood function for a total of D trading days can be formulated as

$$\log \mathscr{L}(\theta \mid \mathscr{M}) = \sum_{d=1}^{D} \log \mathscr{L}(\theta \mid (B_d, S_d)), \tag{3.17}$$

which yields the more explicit notation,

$$\begin{aligned} \log \mathscr{L}(\theta \mid \mathscr{M}) = \sum_{d=1}^{D} \log\Bigg(& (1-\alpha)\Bigg(\exp(-\epsilon)\frac{\epsilon^{B_d}}{B_d!}\exp(-\epsilon)\frac{\epsilon^{S_d}}{S_d!}\Bigg) \\ & + \alpha(1-\delta)\Bigg(\exp\big(-(\epsilon+\mu)\big)\frac{(\epsilon+\mu)^{B_d}}{B_d!}\exp(-\epsilon)\frac{\epsilon^{S_d}}{S_d!}\Bigg) \\ & + \alpha\delta\Bigg(\exp(-\epsilon)\frac{\epsilon^{B_d}}{B_d!}\exp\big(-(\epsilon+\mu)\big)\frac{(\epsilon+\mu)^{S_d}}{S_d!}\Bigg)\Bigg). \end{aligned} \tag{3.18}$$

In general, likelihood functions are very rarely used in maximum-likelihood estimation procedures compared to their logarithmic transformations. Therefore, for the remainder of this work we will refer to the log-likelihood function as likelihood function.

For maximizations of the likelihood function, using the formulation shown in equation (3.18) is very inefficient in terms of computation time and often raises overflow errors.[18] In calculations of equation (3.18) factorials of daily buys and sells need to be evaluated. Since the number of daily buys and sells can easily exceed values of several hundreds or thousands, calculations can easily get infeasible. Despite the fact that the likelihood function in equation (3.18) is theoretically finite, it is not ensured that results are representable by a computer, even if the number of daily buys or sells is small enough to get finite values for each single factorial term.[19] Additionally, the terms ϵ^{B_d}, ϵ^{S_d}, $(\epsilon+\mu)^{B_d}$ and $(\epsilon+\mu)^{S_d}$ are potential sources of overflow errors. Furthermore, single terms may be finite but products of those may not. In contrast to overflow errors induced by large values for daily buys and sells, the exponential terms in the likelihood function may introduce underflow errors (i.e., $\exp(-\epsilon)$ and $\exp\big(-(\epsilon+\mu)\big)$).[20] We will demonstrate and explain the numerical issues one may be confronted with by computing or estimating the probability of informed trading in static models in section 4.1.2.

[18] Overflow errors occur if the calculated number is too big in magnitude so that it cannot be longer represented by the machine/software.

[19] See section 4.1 for more details.

[20] Underflow errors occur if the number to represent is too small and vanishes to zero.

3.2 Model by Easley, Hvidkjaer and O'Hara (EHO)

This model is based on and extends the EKOP model presented in the previous section. In contrast to the simpler model, for the Poisson processes of uninformed buys and uninformed sells each is assumed to have a unique intensity. The expected number of uninformed buys equals ϵ_b, whereas the expected amount of uninformed sells is ϵ_s. The splitting of arrival rates for noise traders results in an extended parameter vector $\theta_{\text{ext}} := (\alpha, \delta, \epsilon_b, \epsilon_s, \mu)$. Hence, the EKOP model is nested in the extended setting for the special case that $\epsilon_b = \epsilon_s$.

The scenario tree in figure 3.2 does no longer hold for the EHO model. Relaxing the assumption of identical intensities for buys and sells initiated by noise traders implies an updated version displayed in figure 3.3.

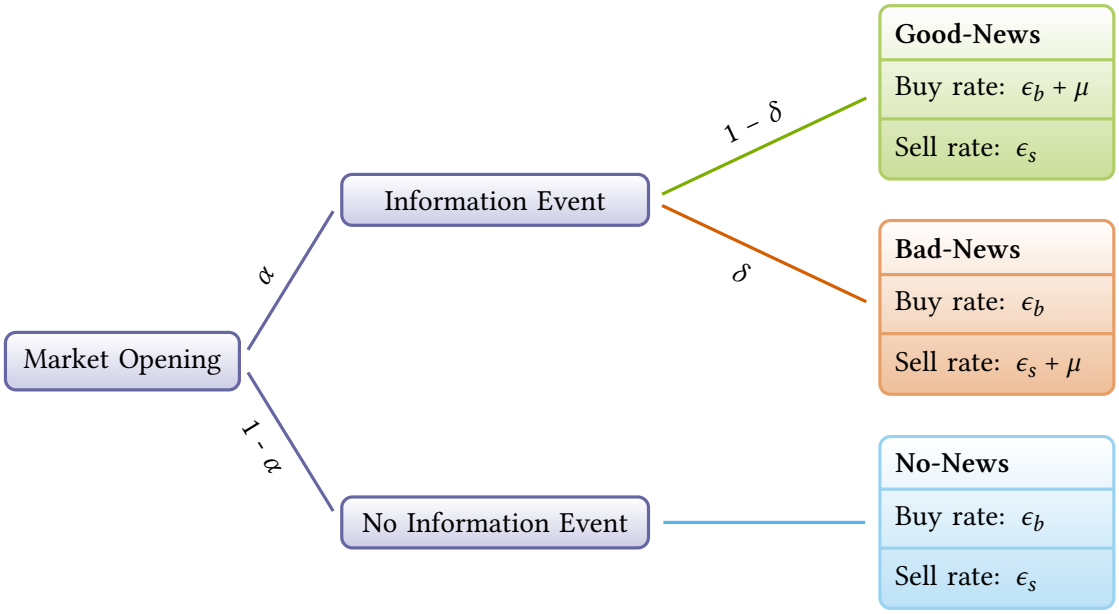

Figure 3.3: Scenario tree for the EHO model with different arrival rates of uninformed buys and uninformed sells. Information events, days on which private information hit the market, occur with probability α. These private information has a negative direction with probability δ (Bad-News) and a positive direction with probability $1 - \delta$ (Good-News). On Bad-News (Good-News) days, the informed traders only sell (buy). Uninformed traders buy equities with rate ϵ_b, and sell them with rate ϵ_s. Information-based buys and sells both have an intensity of μ.

Since the new structure of model parameters needs to be incorporated, the formula for computing the probability of informed trading slightly differs from equation (3.10) and can be written

as

$$\begin{aligned}\text{PIN} &= \frac{\alpha\delta\mu + \alpha(1-\delta)\mu}{(1-\alpha)(\epsilon_b+\epsilon_s) + \alpha\delta(\epsilon_b+\epsilon_s) + \alpha(1-\delta)(\epsilon_b+\epsilon_s) + \alpha\delta\mu + \alpha(1-\delta)\mu} \\ &= \frac{\alpha\mu}{\epsilon_b + \epsilon_s + \alpha\mu}. \end{aligned} \tag{3.19}$$

We can generalize equation (3.18) for deriving the likelihood function in the extended EHO setting. Hence, the likelihood of observing B_d buys and S_d sells on trading days d with $d = 1, \dots, D$ in the extended setting is given by

$$\begin{aligned}\log \mathscr{L}(\theta_{\text{ext}} \mid \mathscr{M}) = \sum_{d=1}^{D} \log\Bigg(&(1-\alpha)\left(\exp(-\epsilon_b)\frac{{\epsilon_b}^{B_d}}{B_d!}\exp(-\epsilon_s)\frac{{\epsilon_s}^{S_d}}{S_d!}\right) \\ &+ \alpha(1-\delta)\left(\exp\big(-(\epsilon_b+\mu)\big)\frac{(\epsilon_b+\mu)^{B_d}}{B_d!}\exp(-\epsilon_s)\frac{{\epsilon_s}^{S_d}}{S_d!}\right) \\ &+ \alpha\delta\left(\exp(-\epsilon_b)\frac{{\epsilon_b}^{B_d}}{B_d!}\exp\big(-(\epsilon_s+\mu)\big)\frac{(\epsilon_s+\mu)^{S_d}}{S_d!}\right)\Bigg). \end{aligned} \tag{3.20}$$

The probabilities that the k-th observed transaction on trading day d, $I_{k,d}$, is buyer- or seller-initiated, with respect to the trading day's condition, are given by:

$$\Pr\big(I_{k,d} = \text{Buy} \,\big|\, \text{No-News}\big) = \frac{\epsilon_b}{\epsilon_b + \epsilon_s} \tag{3.21a}$$

$$\Pr\big(I_{k,d} = \text{Sell} \,\big|\, \text{No-News}\big) = \frac{\epsilon_s}{\epsilon_b + \epsilon_s} \tag{3.21b}$$

$$\Pr\big(I_{k,d} = \text{Buy} \,\big|\, \text{Good-News}\big) = \frac{\epsilon_b + \mu}{\epsilon_b + \epsilon_s + \mu} \tag{3.21c}$$

$$\Pr\big(I_{k,d} = \text{Sell} \,\big|\, \text{Good-News}\big) = \frac{\epsilon_s}{\epsilon_b + \epsilon_s + \mu} \tag{3.21d}$$

$$\Pr\big(I_{k,d} = \text{Buy} \,\big|\, \text{Bad-News}\big) = \frac{\epsilon_b}{\epsilon_b + \epsilon_s + \mu} \tag{3.21e}$$

$$\Pr\big(I_{k,d} = \text{Sell} \,\big|\, \text{Bad-News}\big) = \frac{\epsilon_s + \mu}{\epsilon_b + \epsilon_s + \mu}. \tag{3.21f}$$

Contrary to the simpler model, no-news days do no longer represent (fair) coin-flipping scenarios for the direction of transactions. Whether buyer- or seller-initiated transactions prevail is determined by the relation of the parameters ϵ_b and ϵ_s. If $\epsilon_b > \epsilon_s$ ($\epsilon_b < \epsilon_s$) we expect the majority of observed trades to be buys (sells) on no-news days.

Optimizations may also suffer from overflow or underflow errors due to the same reasons mentioned for the EKOP model. Terms potentially leading computations to overflow are ${\epsilon_b}^{B_d}$, ${\epsilon_s}^{S_d}$, $(\epsilon_b+\mu)^{B_d}$, $(\epsilon_s+\mu)^{S_d}$ and all terms in which the factorial function is involved. Underflow errors can be caused by exponential functions if arguments are too small, as shown in section 4.1.

3.3 Posterior Probabilities

A priori probabilities in static models allow to determine the proportions each state, either no-news, good-news or bad-news, occupies of the total number of trading days. However, we do not receive any information about the state of each single trading day. In other words, we know how often the states occur *on average*, but we do not know the actual number of occurrences for each condition. By harnessing Bayes' theorem[21] we can extend the scope of static PIN models and introduce some aspects of the main research focus of dynamic models.

Utilizing the joint probability of observing B_d buys and S_d sells on trading day d given in (3.15), we can calculate the probability that a trading day d resides in no-news state, given that we have observed B_d buys and S_d sells. Since (3.15) fits the simpler EKOP model we need to apply little modifications and split the intensity of uninformed trading ϵ into the two variables ϵ_b and ϵ_s for the extended EHO setting.[22]

The posterior probability for a no-news day can then be written as

$$\begin{aligned}\Pr(\mathcal{N} \mid (B_d, S_d)) &= \frac{\Pr(B_d \mid \mathcal{N}) \Pr(S_d \mid \mathcal{N}) \Pr(\mathcal{N})}{\Pr(B_d, S_d)} \\ &= \frac{1-\alpha}{(1-\alpha) + \exp(-\mu)\left[\alpha(1-\delta)\left(1+\frac{\mu}{\epsilon_b}\right)^{B_d} + \alpha\delta\left(1+\frac{\mu}{\epsilon_s}\right)^{S_d}\right]}.\end{aligned} \tag{3.22}$$

Likewise, posterior probabilities for a good-news and bad-news trading day are given by

$$\begin{aligned}\Pr(\mathcal{G} \mid (B_d, S_d)) &= \frac{\Pr(B_d \mid \mathcal{G}) \Pr(S_d \mid \mathcal{G}) \Pr(\mathcal{N})}{\Pr(B_d, S_d)} \\ &= \frac{\alpha(1-\delta)\exp(-\mu)\left(1+\frac{\mu}{\epsilon_b}\right)^{B_d}}{(1-\alpha) + \exp(-\mu)\left[\alpha(1-\delta)\left(1+\frac{\mu}{\epsilon_b}\right)^{B_d} + \alpha\delta\left(1+\frac{\mu}{\epsilon_s}\right)^{S_d}\right]}\end{aligned} \tag{3.23}$$

and

$$\begin{aligned}\Pr(\mathcal{B} \mid (B_d, S_d)) &= \frac{\Pr(B_d \mid \mathcal{B}) \Pr(S_d \mid \mathcal{B}) \Pr(\mathcal{N})}{\Pr(B_d, S_d)} \\ &= \frac{\alpha\delta\exp(-\mu)\left(1+\frac{\mu}{\epsilon_s}\right)^{S_d}}{(1-\alpha) + \exp(-\mu)\left[\alpha(1-\delta)\left(1+\frac{\mu}{\epsilon_b}\right)^{B_d} + \alpha\delta\left(1+\frac{\mu}{\epsilon_s}\right)^{S_d}\right]}.\end{aligned} \tag{3.24}$$

[21] An introduction to Bayesian statistics can be found in the book by Lee (2012).

[22] Since the simpler EKOP model is nested in EHO model for $\epsilon_b = \epsilon_s$, formulas can easily be adapted.

Exemplarily, assuming low probability parameters α and δ, we can interpret this as few trading days in the sample period on which insiders triggered by positive private information enter the market. However, without posterior probabilities, we are not able to assign information events to specific trading days in the datasource. Utilizing equations (3.22) - (3.24) we can identify good-news days according to the magnitude of $\Pr(\mathcal{G} \mid (B_d, S_d))$ for each trading day. Hence, posterior probabilities deliver useful additional insights in the classification of trading days and help to extend the scope of analyses of static PIN models.

4 Maximum-Likelihood Estimation in Static Models

4.1 Numerical Issues

At the beginning of this chapter we explain and discuss the problems one would have to deal with if the original representation of the likelihood functions in equations (3.18) and (3.20) would be used.

Throughout the thesis we use the statistical programming language R for all calculations. In this section we investigate in detail the over- and underflow errors in computations of the likelihood function in the EKOP and EHO model mentioned in sections 3.1 and 3.2. Therefore, we discuss how numbers are stored by R and their maximum precision in terms of decimal places.[23] Furthermore, we simulate a dataset which represents an infrequently traded equity with low number of daily buys and sells. Finally, we inspect the results for the likelihood functions in the EKOP and EHO model for the simulated dataset.

4.1.1 Floating-Point Representation in R

The statistical programming language R uses the *IEEE 754* norm for floating-point arithmetic (see IEEE (1985)). By default, R uses double-precision for numeric values unless user-interaction happens specifying numbers as integers manually. A floating-point number (double) x is stored in *binary64* format and consists of sign s, mantissa m and exponent e

$$x = (-1)^s \cdot m \cdot 2^e, \tag{4.1}$$

As the name of the storing format suggests, double-precision floating numbers occupy 64 bits in memory. Sign s and mantissa (significand) m use 1 bit and 52 (+ 1 hidden) bits, respectively. The

[23] Explanations in section 4.1.1 hold for every programming language using double-precision for floats.

most significant bit for a normalized number is always 1 if working in binary and therefore it is sufficient to only store 52 bits explicitly. The remaining 11 bits are reserved for exponent e, as visualized in figure 4.1. Real-valued normalized numbers with exponent e and 52-bit mantissa are represented by

$$(-1)^s \cdot (1.m_{51}m_{50} \dots m_0)_2 \cdot 2^e,$$

with $m_i \in \{0, 1\}$, $i = 0, \dots, 51$ (e.g., see Goldberg 1991).

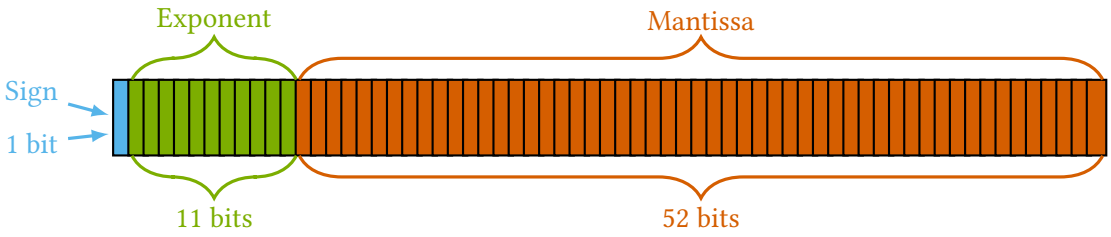

Figure 4.1: Floating-point representation in *binary64* format.

Hence, floating-point numbers exhibit an accuracy of 53 bits or approximately 16 decimal places. The conversion from base 2 to the decimal system leads to: $53log_{10}(2) \approx 15.95$ digits.

Furthermore, the *IEEE 754* standard distinguishes between normalized and denormal (subnormal) floating-point numbers. For a better understanding of the difference between these two types we take a look at some entries of the (invisible) `.Machine`[24] variable in R. This variable returns useful numerical characteristics of the machine R is currently running on.[25]

Code Chunk 4.1 (Machine characteristics):

```
# Print machine characteristics to console
.Machine[c("double.eps", "double.neg.eps",
           "double.xmin", "double.xmax", "double.base",
           "double.min.exp", "double.max.exp")]


$double.eps
[1] 2.22045e-16

$double.neg.eps
[1] 1.11022e-16

$double.xmin
[1] 2.22507e-308

$double.xmax
```

[24] The `.Machine` variable is shipped with the **base** package.
[25] Results of the code chunk were generated by R 3.5.2 using Linux and an Intel i5-4590 processor.

```
[1] 1.79769e+308

$double.base
[1] 2

$double.min.exp
[1] -1022

$double.max.exp
[1] 1024
```

The list elements `double.xmin` and `double.xmax` provide information about the range of normalized numbers. The largest normalized and smallest normalized non-zero floating-point number R can represent are stored in `double.xmax` and `double.xmin`, respectively. Both depend on the used base (radix) for floating-point representations which can be read from `double.base`. In addition, `double.neg.eps` influences `double.xmax`. To be specific,

$$\texttt{double.xmin} = \texttt{double.base}^{\texttt{double.min.exp}}$$

$$\texttt{double.xmax} = (1 - \texttt{double.neg.eps}) \cdot \texttt{double.base}^{\texttt{double.max.exp}},$$

where `double.min.exp` is the largest in magnitude negative integer i such that $\texttt{double.base}^i$ is positive and normalized.[26] The smallest positive power of `double.base` which overflows is given in `double.max.exp`. The value for `double.neg.eps` gives the smallest positive floating-point number `x` such that the difference `1 - x` is not identical to 1, $\texttt{1 - x} \neq 1$. Likewise, the list element `double.eps` represents the smallest positive floating-point number `x` such that the sum `1 + x` does not equal 1, $\texttt{1 + x} \neq 1$.

Denormal numbers fill the gap between zero and the smallest non-zero normalized floating-point number `double.xmin`, as shown in figure 4.2. They have the smallest possible exponent

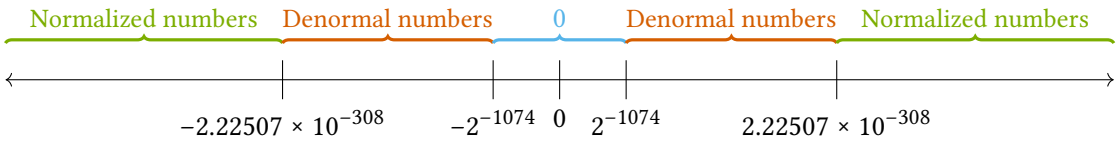

Figure 4.2: Visualization of normalized and denormal numbers (IEEE-754). The value of the smallest non-zero floating-point number can be read from the `double.x.min` value in code chunk 4.1. The smallest non-zero denormal number has an absolute value of 2^{-1074}.

for normalized numbers ($e = 1022$, `double.min.exp` value in code chunk 4.1), but at least one leading digit of the non-zero mantissa equals 0 (e.g., see Dooley and Kale 2006). The more leading digits are zero for the mantissa, the more precision is lost. Hence, the absolute value of the minimal non-zero denormal number is $2^{-1022-52} = 2^{-1074}$. In addition to the reduced

[26] The value of $\texttt{double.base}^i$ lies in the range from `double.xmin` to `double.xmax`.

precision, computation time increases (e.g., see Dooley and Kale 2006). Therefore, it is best to avoid computations in which denormal numbers are involved as long as full precision for *doubles* is needed.[27]

There are additional R packages like `Rmpfr` that enable the user to increase the number of significant bits for numeric values and some operations thereof. With the `Rmpfr` package values can be stored with arbitrary precision (see Maechler (2015)). This can be useful in situations where really high accuracy is essential but comes at the cost of speed. However, for evaluations of the likelihood functions in static PIN models factorials and exponentials need to be computed for (very) high numbers of buys and sells, if one is interested in frequently traded stocks. Hence, using such high-precision libraries results in dramatic slowdowns due to the (very) high precision which is required to avoid over- and underflows.

Since there is a fast and stable implementation of the likelihood function with which we can circumvent numerical issues by still using double-precision, we will not delve deeper into the field of arbitrary precision arithmetic. The most stable factorization of the model likelihood function is presented in section 4.2.2.

4.1.2 Over- and Underflow Errors

As previously mentioned, computations of factorials and exponentials is crucial for calculations of the likelihood functions given in equations (3.18) and (3.20). We will shortly demonstrate for which numbers these functions fail and either over- or underflow, before we simulate data, which may represent an infrequently traded stock, and analyze corresponding intermediate results of the likelihood functions in the EKOP and EHO model.

The statistical programming language R is equipped with the *factorial* function which relies on the *gamma* function[28] to evaluate the factorial of any positive real number.

Code Chunk 4.2 (Evaluation of factorial function):

```
# Define a vector 'x' with positive real-valued elements
x <- c(50, 100, 150, 170, 171)

# Compute the factorial for each element in 'x'
factorial(x)
```

```
Warning in factorial(x): value out of range in 'gammafn'
```

```
[1]  3.04141e+64 9.33262e+157 5.71338e+262 7.25742e+306          Inf
```

[27] Further numerical characteristics are listed on the help page of `.Machine` variable which can be accessed from the R console by `?.Machine`.

[28] The gamma function is defined as improper integral: $\Gamma(x) = \int_0^\infty s^{x-1} e^{-s} ds$.

We can see that R is not capable of returning finite values for the factorial function of numbers bigger than 170. This yields overflow errors for the factorial terms even for datasets of very infrequently traded stocks.

Evaluation of exponential function can be represented for exponents in the range of $[-745, 709]$. Smaller exponents yield a value of zero while larger exponents return infinite values.

Code Chunk 4.3 (Evaluation of exponential function):

```
small_exponents <- -742:-746
exp(small_exponents)
```

```
[1] 5.43472e-323 1.97626e-323 9.88131e-324 4.94066e-324  0.00000e+00
```

```
large_exponents <- 706:710
exp(large_exponents)
```

```
[1] 4.09170e+306 1.11224e+307 3.02338e+307 8.21841e+307          Inf
```

Intensity parameters or the sum thereof need to be bigger in magnitude than the threshold of 745 to introduce underflow errors. This happens only for more frequently traded stocks. However, for such equities computations of factorials of daily buys and sells as well as computations of terms where power operations are involved result in infinite values.

Below we will show how overflow errors strike the calculations of likelihood functions in the EKOP and EHO model. Code chunk 4.4 specifies model parameters for the EKOP and EHO case, while in code chunk 4.5 the seed of the random number generator is set to ensure that results are reproducible. Furthermore, a sequence of trading days' conditions is sampled in code chunk 4.6 while the subsequent code chunk 4.7 simulates aggregated daily buys and sells data. We set the length of the sequence of trading days to 60 which is a common recommendation in the literature to ensure the convergence of the likelihood functions in optimizations (e.g., see Easley, Kiefer, O'Hara, and Paperman 1996, Aktas, de Bodt, Declerck, and van Oppens 2007).

We simulate two sets of daily buys and sells. One set using the data generating process of the EKOP model and one taking into account the assumption from the EHO model that buys and sells initiated by uninformed traders differ in arrival rates. The intensities for noise traders in the EHO setting ϵ_b and ϵ_s equal 135 and 125, respectively. On average, conditional on an information event, we expect to observe 50 transactions driven by private information ($\mu = 50$).[29] In the data simulating procedure for the simpler setting we do not change the portion of informed traders, but use the mean of the values for ϵ_b and ϵ_s as intensity of noise trading.

[29] The simulated dataset for the EHO setting is shipped with the **pinbasic** package. To access it run `data('BSinfrequent')` in the console with **pinbasic** loaded.

Hence, the intensity of uninformed market attendees in the EKOP model is given by $\epsilon = 130$.[30]

Code Chunk 4.4 (Specifying intensities of buys and sells):

```
# intensity of uninformed buys
epsilon_b <- 135

# intensity of uninformed sells
epsilon_s <- 125

# intensity of noise trading in EKOP model
epsilon <- mean(c(epsilon_b, epsilon_s))

# intensity of informed trading
mu <- 50
```

At first, we set the seed for the R internal pseudo-random number generator to ensure that the results are reproducible.[31] The default (pseudo) random number generator in R is set to *Mersenne-Twister* developed by Matsumoto and Nishimura (1998), but there are various other options available.[32]

Code Chunk 4.5 (Setting seed for RNG):

```
# Specify state of random number generator
set.seed(123)
```

In the next step we need to sample conditions for trading days. We set the probability of an information event to 20% and assume that good-news and bad-news days are equally likely to occur. With α and δ fixed, the probabilities for the different types of trading days are given by

$$\Pr(\mathscr{N}) = 0.8$$

$$\Pr(\mathscr{G}) = \Pr(\mathscr{B}) = 0.1.$$

With the built-in *sample* function, we can now simulate a sequence of trading days' states by sampling with replacement and fetch the indices for each type.

[30] The chosen intensities are very small and only appropriate for very infrequently traded stocks. However, the goal of this section is to give the reader an insight into the numerical problems one can be confronted with when trying to calculate the likelihood function.

[31] The term *reproducible* implies that if code chunks are copy & pasted to the R console, identical vectors for buys and sells are sampled and therefore the results are equal to those presented in this work.

[32] More information about the available random number generators can be found on the help page of *set.seed* function (`?set.seed`).

Code Chunk 4.6 (Sampling a sequence of trading days):

```
# number of days
ndays <- 60

# probability parameters
alpha <- 0.2
delta <- 0.5

# actual sampling
states <- sample(c("no", "good", "bad"),
                 size = ndays, replace = TRUE,
                 prob = c(1 - alpha,
                          alpha * (1 - delta),
                          alpha * delta))

# indices for no-, good- and bad-news days
ind_no <- which(states == "no")
ind_good <- which(states == "good")
ind_bad <- which(states == "bad")
```

Now that we are able to identify the state every simulated trading day resides in, we can start to sample daily buys and sells data for both static models.[33] For computations of likelihood functions we incorporate the same parameters used for simulation, as can be seen in code chunk 4.8.

Likelihood functions given in equations (3.18) and (3.20) are implemented with *ll_issues* function in code chunk 4.9.[34] The values for model parameters can be passed with argument `param`, arguments `numbuys` and `numsells` take daily buys and sells data and logical argument `ext` is a switch between the simple and extended static model.

Code Chunk 4.7 (Sampling daily buys and sells data):

```
# initialize numeric vectors for buys and sells
buys_eho <-
  buys_ekop <-
  sells_eho <-
  sells_ekop <- numeric(ndays)
```

[33] To generate random numbers from Poisson distributions we use the *rpois* function which is shipped with the built-in **stats** package.

[34] This function is not included in the **pinbasic** package. However, the code is shown for better demonstration of the problematic parts of likelihood computations.

```
# drawing Poisson distributed random numbers for daily buys and sells
# according to the actual buy and sell intensities depending on the
# condition of the trading day
buys_eho[ind_no] <- rpois(length(ind_no), epsilon_b)
sells_eho[ind_no] <- rpois(length(ind_no), epsilon_s)
buys_ekop[ind_no] <- rpois(length(ind_no), epsilon)
sells_ekop[ind_no] <- rpois(length(ind_no), epsilon)

buys_eho[ind_good] <- rpois(length(ind_good), epsilon_b + mu)
sells_eho[ind_good] <- rpois(length(ind_good), epsilon_s)
buys_ekop[ind_good] <- rpois(length(ind_good), epsilon + mu)
sells_ekop[ind_good] <- rpois(length(ind_good), epsilon)

buys_eho[ind_bad] <- rpois(length(ind_bad), epsilon_b)
sells_eho[ind_bad] <- rpois(length(ind_bad), epsilon_s + mu)
buys_ekop[ind_bad] <- rpois(length(ind_bad), epsilon)
sells_ekop[ind_bad] <- rpois(length(ind_bad), epsilon + mu)
```

Code Chunk 4.8 (Specifying parameter vectors):

```
# parameter vector for EKOP model
pin_ekop_coef <- c(alpha, delta, epsilon, mu)

# parameter vector for EHO model
pin_eho_coef <- c(alpha, delta, epsilon_b, epsilon_s, mu)

# name the vector elements
names(pin_ekop_coef) <- c("alpha", "delta", "epsilon", "mu")
names(pin_eho_coef) <- c("alpha", "delta", "epsilon_b", "epsilon_s", "mu")
```

The *ll_issues* function returns a matrix with six columns. The first three columns contain the results for the components associated with no-news, good-news and bad-news days, respectively. The fourth column consists of daily likelihood values. The last two columns comprise of aggregated daily buys and sells. In tables 4.1 and 4.2 a column displaying trading days' conditions is added.

We evaluate the likelihood function in the simpler EKOP model with model parameters stored in `pin_ekop_coef` as well as data vectors `buys_ekop` and `sells_ekop`. The corresponding results for the first six days are displayed in table 4.1.

After computation we inspect the failure rate of daily likelihood values and assign the number of infinite function values or `NaN`[35] to `total_fail_ekop` (see code chunk 4.10).

[35] R returns `NaN` for some computations in which infinite values are involved: e.g., `Inf - Inf`, `Inf/Inf`, or for undefined division `0/0`.

Code Chunk 4.9 (Implementation of inefficient and unstable likelihood in static models):

```
ll_issues <- function(param, numbuys, numsells, ext = TRUE){
  if(!ext) param["epsilon_b"] <- param["epsilon_s"] <- param["epsilon"]

  no_news <- (1 - param["alpha"]) *
             exp(-param["epsilon_b"]) *
             (param["epsilon_b"]) ^ numbuys/
             factorial(numbuys) *
             exp(-param["epsilon_s"]) *
             (param["epsilon_s"]) ^ numsells/
             factorial(numsells)

  good_news <- param["alpha"] * (1 - param["delta"]) *
               exp(-(param["epsilon_b"] + param["mu"])) *
               (param["epsilon_b"] + param["mu"]) ^ numbuys/
               factorial(numbuys) *
               exp(-param["epsilon_s"]) *
               (param["epsilon_s"]) ^ numsells/
               factorial(numsells)

  bad_news <- param["alpha"] * param["delta"] *
              exp(-param["epsilon_b"]) *
              (param["epsilon_b"]) ^ numbuys/factorial(numbuys) *
              exp(-(param["epsilon_s"] + param["mu"])) *
              ((param["epsilon_s"] + param["mu"])) ^ numsells/
              factorial(numsells)

  ll_daily <- log(no_news + good_news + bad_news)

  res <- matrix(c(no_news, good_news, bad_news,
                  ll_daily,
                  numbuys, numsells),
                nrow = length(numbuys))

  colnames(res) <- c("No-News", "Good-News", "Bad-News",
                     "ll_daily",
                     "Buys", "Sells")
  res
}
```

Code Chunk 4.10 (EKOP likelihood evaluation):

```
# likelihood evaluation
ll_fail_ekop <- ll_issues(param = pin_ekop_coef, ext = FALSE,
                          numbuys = buys_ekop, numsells = sells_ekop)
```

```
# inspecting infinite values and NaN
total_fail_ekop <- sum(is.infinite(ll_fail_ekop[,"ll_daily"]) |
                       is.nan(ll_fail_ekop[,"ll_daily"]))

total_fail_ekop


[1] 36
```

No-News	Good-News	Bad-News	ll_daily	Buys	Sells	State
2.81573e-04	8.56258e-10	8.77608e-11	-8.17511	121	114	no
7.38283e-04	4.30422e-09	5.95969e-09	-7.21117	123	124	no
4.97065e-04	Inf	2.04202e-08	Inf	143	129	no
NaN	NaN	NaN	NaN	138	173	bad
NaN	NaN	NaN	NaN	182	126	good
4.54247e-04	Inf	9.97647e-10	Inf	140	120	no

Table 4.1: First six rows of the returned matrix from a call to *ll_issues* function in which argument `ext` is set to `FALSE`. Data vectors `buys_ekop` and `sells_ekop` as well as the parameter set stored in `pin_ekop_coef` are used for computation.

For the computation of the likelihood function in the EKOP model many warnings of the form `Warning in factorial(): value out of range in 'gammafn'` arise.[36] In total, for 36 days an infinite value or `NaN` is returned for `ll_daily` column. For more than half of the trading days in the simulated dataset daily likelihood values raise overflow errors or floating point exceptions (FPE) although the parameters of the simulation procedure are set very low to mimic daily buys and sells data of very infrequently traded stocks. Obviously, calculations under the assumption of identical trading intensities for noise trading are not feasible utilizing the likelihood formulation given in equation (3.18), even for infrequently traded equities.

By switching from the restrictive assumption of identical noise trader intensities to the extended EHO setting in code chunk 4.11 we get similar results, table 4.2 shows them for the first six days. The number of trading days for which daily likelihood values are not finite or `NaN` even increases.

Only for 22 trading days a real number is returned for `ll_daily`. The evaluations of likelihood functions in the EKOP and EHO model show that straightforward implementations of the formulations in equation (3.18) and (3.20) cannot be used in applications which try to estimate the probability of informed trading for any recent stock data.

[36] By default R prints these warnings to the console, but they are suppressed in this thesis to reduce the size of the output.

Code Chunk 4.11 (EHO likelihood evaluation):

```
# likelihood evaluation
ll_fail_eho <- ll_issues(param = pin_eho_coef, ext = TRUE,
                         numbuys = buys_eho, numsells = sells_eho)

# inspecting infinite values and NaN
total_fail_eho <- sum(is.infinite(ll_fail_eho[,"ll_daily"]) |
                      is.nan(ll_fail_eho[,"ll_daily"]))

total_fail_eho
```

```
[1] 38
```

Thus we have developed the **pinbasic** R package whose functionalities are designed to offer a fast and stable estimation of PIN in the static models presented and discussed in the previous chapter. The next chapter exhibits implemented functions together with an extensive explanation of the underlying source code.

No-News	Good-News	Bad-News	ll_daily	Buys	Sells	State
8.40579e-04	Inf	1.43642e-07	Inf	139	129	no
8.91695e-04	1.81207e-08	1.03251e-08	-7.02235	131	121	no
9.48740e-04	3.62061e-08	8.27168e-08	-6.96025	133	127	no
Inf	Inf	Inf	Inf	145	169	bad
NaN	NaN	NaN	NaN	185	135	good
Inf	Inf	Inf	Inf	145	119	no

Table 4.2: First six rows of the returned matrix from a call to *ll_issues* function in which argument `ext` is set to `TRUE`. Data vectors `buys_eho` and `sells_eho` as well as the parameter set stored in `pin_eho_coef` are used for computation.

4.2 Likelihood Factorizations

According to section 4.1.2, computation of equations (3.18) and (3.20) even fails for infrequently traded stocks. Historical data which has its origin decades ago and therefore the aggregated buys and sells data is very small may be an appropriate choice for these likelihood formulations. However, for any recent dataset more stable implementations are essential to achieve finite function values.

The PIN literature provides two widely used likelihood factorizations which try to minimize over- and underflow errors. Since the EKOP model is nested in the extended EHO setting, we

will concentrate on the latter for the remainder of this chapter.

4.2.1 EHO Factorization

Easley, Hvidkjaer, and O'Hara (2010) reformulated the likelihood function in the static model with different intensities for uninformed buys and uninformed sells. The authors rearranged the likelihood function and dropped the constant term $-\log(B_d!S_d!)$ so that an algebraically equivalent but more stable and robust factorization can be maximized,

$$\begin{aligned}\log \mathscr{L}\big(\theta_{\text{ext}} \mid \mathscr{M}\big) = \sum_{d=1}^{D}\Big(-\epsilon_b - \epsilon_s + M_d\big(\log x_b + \log x_s\big) + B_d \log\big(\mu + \epsilon_b\big) + S_d \log\big(\mu + \epsilon_s\big)\Big) \\ + \sum_{d=1}^{D} \log\Big((1-\alpha)x_s^{S_d - M_d} x_b^{B_d - M_d} + \alpha(1-\delta)\exp\big(-\mu\big)x_s^{S_d - M_d} x_b^{-M_d} \\ + \alpha\delta\exp\big(-\mu\big)x_b^{B_d - M_d} x_s^{-M_d}\Big),\end{aligned} \tag{4.2}$$

where $M_d = \min(B_d, S_d) + \dfrac{\max(B_d, S_d)}{2}$, $x_s = \dfrac{\epsilon_s}{\epsilon_s + \mu}$ and $x_b = \dfrac{\epsilon_b}{\epsilon_b + \mu}$. Technically speaking, due to the dropping of the constant term, the expression in equation (4.2) is no longer a likelihood function. However, we go with the literature and still use the term and do not introduce a new symbol.

The maximum likelihood estimator, denoted as $\hat{\theta}_{\text{ext}} = \big(\widehat{\alpha}, \hat{\delta}, \widehat{\epsilon_b}, \widehat{\epsilon_s}, \hat{\mu}\big)$, is then obtained by maximizing the likelihood function in equation (4.2),

$$\max_{\theta_{\text{ext}} \in \text{BFS}} \log \mathscr{L}\big(\theta_{\text{ext}} \mid \mathscr{M}\big), \tag{4.3}$$

where $\text{BFS} = \Big\{\theta_{\text{ext}} = \big(\alpha, \delta, \epsilon_b, \epsilon_s, \mu\big) \;\Big|\; \alpha, \delta \in [0,1] \text{ and } \epsilon_b, \epsilon_s, \mu \in [0, \infty)\Big\}$ denotes the set of basic feasible solutions (see Lin and Ke 2011). Estimates of the probability of informed trading are then calculated with (see equation (3.19))

$$\widehat{\text{PIN}} = \frac{\widehat{\alpha}\hat{\mu}}{\widehat{\epsilon_b} + \widehat{\epsilon_s} + \widehat{\alpha}\hat{\mu}}. \tag{4.4}$$

According to Easley, Hvidkjaer, and O'Hara (2010) the computation of the probability of informed trading benefits due to two facts. The computing efficiency is increased and the truncation errors (over- and underflow) are reduced. No evaluations of factorials are needed, additionally x_b and x_s are always weakly smaller than 1 which leads to more stable calculations of

the terms involving power operations. Instead of a straightforward implementation of equation (4.2), we exploit the relations

$$x^y = \exp\big(\log(x) * y\big) \text{ with } x > 0, y \in \mathbb{R} \text{ and} \tag{4.5}$$

$$\exp\big(x + y\big) = \exp(x) \cdot \exp\big(y\big) \text{ with } x, y \in \mathbb{R}, \tag{4.6}$$

which yield an increased stability of computations.

Applying only the first relation to equation (4.2), corresponding terms can be transformed as follows:

$$x_s^{S_d - M_d} x_b^{B_d - M_d} = \exp\big((S_d - M_d)\log x_s\big)\exp\big((B_d - M_d)\log x_b\big), \tag{4.7a}$$

$$\exp\big(-\mu\big)x_s^{S_d - M_d} x_b^{-M_d} = \exp\big(-\mu\big)\exp\big((S_d - M_d)\log x_s\big)\exp\big(-M_d \log x_b\big), \tag{4.7b}$$

$$\exp\big(-\mu\big)x_b^{B_d - M_d} x_s^{-M_d} = \exp\big(-\mu\big)\exp\big((B_d - M_d)\log x_b\big)\exp\big(-M_d \log x_s\big). \tag{4.7c}$$

Utilizing both relations for exponential terms, we finally get:

$$x_s^{S_d - M_d} x_b^{B_d - M_d} = \exp\big(B_d * \log(x_b) + S_d * \log(x_s) - M_d * \big(\log(x_b) + \log(x_s)\big)\big), \tag{4.8a}$$

$$\exp\big(-\mu\big)x_s^{S_d - M_d} x_b^{-M_d} = \exp\big(-\mu + S_d * \log(x_s) - M_d * \big(\log(x_b) + \log(x_s)\big)\big), \tag{4.8b}$$

$$\exp\big(-\mu\big)x_b^{B_d - M_d} x_s^{-M_d} = \exp\big(-\mu + B_d * \log(x_b) - M_d * \big(\log(x_b) + \log(x_s)\big)\big). \tag{4.8c}$$

There are still three clear-cut parts of the likelihood function which can be assigned to trading days' conditions. Using equations (4.8a) – (4.8c) we can write them as

$$\begin{aligned}
(1 - \alpha)\exp\big(B_d * \log(x_b) + S_d * \log(x_s) - M_d * \big(\log(x_b) + \log(x_s)\big)\big) &\quad \text{(no-news)}, \\
\alpha(1 - \delta)\exp\big(-\mu + S_d * \log(x_s) - M_d * \big(\log(x_b) + \log(x_s)\big)\big) &\quad \text{(good-news)} \quad \text{and} \\
\alpha\delta\exp\big(-\mu + B_d * \log(x_b) - M_d * \big(\log(x_b) + \log(x_s)\big)\big) &\quad \text{(bad-news)}.
\end{aligned} \tag{4.9}$$

In addition, we label the sum

$$\sum_{d=1}^{D}\Big(-\epsilon_b - \epsilon_s + M_d\big(\log x_b + \log x_s\big) + B_d \log\big(\mu + \epsilon_b\big) + S_d \log\big(\mu + \epsilon_s\big)\Big),$$

as *General Part*. The values for the constant, for maximization purposes irrelevant, component $\log(B_d!S_d!)$ are not included in any tables presented in this section.

We get very detailed results of likelihood evaluations by *ehofactr_mat*[37] function which returns a matrix consisting of daily values for each of the likelihood parts. The `param` arguments takes

[37] The *ehofactr_mat* function is not included in **pinbasic** package. It is presented and used in this work for demonstration purposes only.

values for model parameters, `numbuys` and `numsells` expect vectors of daily buys and sells and `ext` is again a logical which distinguishes between model settings.

To demonstrate how the reformulated likelihood functions handles data of infrequently traded stocks we apply *ehofactr_mat* function to the simulated dataset from section 4.1.2. For evaluations of equation (4.2) we utilize the sets of parameters which were already used for simulations (see code chunks 4.4, 4.6 and 4.8).

Code Chunk 4.12 (Detailed output of computations utilizing EHO likelihood factorization):

```
ehofactr_mat <- function(param = NULL, numbuys = NULL, numsells = NULL) {
  n <- length(numbuys)

  m <- pmin.int(numbuys, numsells) + 0.5 * pmax.int(numbuys,numsells)
  xs <- param["epsilon_s"]/(param["mu"] + param["epsilon_s"])
  xb <- param["epsilon_b"]/(param["mu"] + param["epsilon_b"])
  lxb <- log(xb)
  lxs <- log(xs)

  prob_no <- 1 - param["alpha"]
  prob_good <- param["alpha"] * (1 - param["delta"])
  prob_bad <- param["alpha"] * param["delta"]

  part1 <- -(param["epsilon_b"] + param["epsilon_s"]) +
            (lxb + lxs) * m + log(param["mu"] + param["epsilon_b"]) *
            numbuys + log(param["mu"] + param["epsilon_s"]) *
            numsells

  part2 <- log(prob_no * exp(lxb * numbuys + lxs * numsells - m * (lxb + lxs)) +
               prob_good * exp(-param["mu"] + numsells * lxs - m * (lxb + lxs)) +
               prob_bad * exp(-param["mu"] + numbuys * lxb - m * (lxb + lxs)))

  ll <- cbind(part1,
              prob_no * exp(lxb * numbuys + lxs * numsells - m * (lxb + lxs)),
              prob_good * exp(-param["mu"] + numsells * lxs - m * (lxb + lxs)),
              prob_bad * exp(-param["mu"] + numbuys * lxb - m * (lxb + lxs)),
              part1 + part2,
              numbuys,
              numsells)

  colnames(ll) <- c("General Part", "No-News", "Good-News", "Bad-News",
                    "ll_daily", "Buys", "Sells")

  ll
}
```

The intention of including tables 4.1, 4.2 and 4.3 in this work is to give the reader an idea of the magnitude of values occurring in computations of likelihood functions in static PIN models.

General Part	No-News	Good-News	Bad-News	ll_daily	Buys	Sells
1.00255e+03	1.58799e+18	4.01337e+14	2.71362e+14	1044.46295	139	129
9.27291e+02	1.17215e+17	2.38200e+12	1.35725e+12	966.59385	131	121
9.64160e+02	7.93051e+17	3.02646e+13	6.91429e+13	1005.37437	133	127
1.22027e+03	2.02602e+20	3.39105e+17	2.42364e+22	1271.81943	145	169
1.25478e+03	1.72007e+19	8.56555e+21	2.21318e+16	1305.28761	185	135
9.86789e+02	7.25000e+16	1.21347e+14	4.28312e+11	1025.61271	145	119

Table 4.3: First six rows of likelihood evaluation in the extended static model setting establishing EHO factorization. Synthetic EHO data representing an infrequently traded stock from section 4.1.2 is used for computation. We decided to stick with the layout and column names of table 4.2, however, the values in the column *ll_daily* are not directly comparable since the constant term $-\log(B_d!S_d!)$ is dropped.

For the simulated data, representing a (very) infrequent traded equity, EHO factorization is sufficient. Each daily likelihood value is finite and they sum up to 63791.61.

The conclusion of this section so far is that we now have a more stable likelihood formulation which enables us to estimate the probability of informed trading for equities with a small amount of daily buys and sells. It remains to be seen whether it can handle datasets from moderate or even very frequently traded stocks. The factorization by Easley, Hvidkjaer, and O'Hara (2010) has similar drawbacks as equations (3.18) and (3.20) but they are not that distinct.

To shortly demonstrate the failure of the reformulated likelihood function we simulate a new, more realistic, dataset of daily buys and sells representing a frequently traded equity according to the EHO model setup. The simulation procedure does not change in comparison to section 4.1.2, but the intensities do. Conditions of trading days are not altered or sampled again. Hence, we use the same type and order of trading days but set trading intensities to higher levels. We now expect to observe 2200 uninformed buys and 2000 uninformed sells per trading day. The amount of transactions due to private information on information events equals 800.[38] Thus, we get the parameter vector $\theta_{\text{ext}} = (0.2, 0.5, 2200, 2000, 800)$ which is used in the simulation of the dataset as well as the evaluation of the likelihood function.[39]

We can observe infinite results in each part of the likelihood function which can be assigned to either no-news, good-news or bad-news condition in table 4.4. Hence, for a (very) frequently traded stock the factorization discussed in this chapter is infeasible.

There is a subset BFS_{EHO} of BFS with

$$\text{BFS}_{\text{EHO}} = \left\{\theta_{\text{ext}} \in \text{BFS} \;\middle|\; \log \mathscr{L}\left(\theta_{\text{ext}} \mid \mathscr{M}\right) \text{ do not lead to FPE}\right\}, \tag{4.10}$$

[38] The simulated dataset is shipped with the **pinbasic** package. To access it run `data('BSfrequent')` in the console with **pinbasic** package loaded.

[39] Before drawing random numbers from Poisson distributions for daily buys and sells we set the seed of the internal random number generator with the *set.seed* function and the corresponding function call `set.seed(321)`.

General Part	No-News	Good-News	Bad-News	ll_daily	Buys	Sells
2.79568e+04	3.43872e+283	1.49889e+242	5.87415e+228	28609.65850	2279	2009
2.74969e+04	2.71489e+290	7.11521e+233	7.07083e+243	28165.63701	2166	2065
2.71451e+04	1.06598e+282	2.14765e+227	1.72743e+226	27794.52713	2180	2002
3.47072e+04	Inf	1.19803e+266	Inf	Inf	2248	2937
3.42059e+04	2.09597e+292	Inf	1.69836e+242	Inf	3062	2041
2.76391e+04	1.76514e+289	2.73385e+236	1.58934e+241	28305.12194	2194	2055

Table 4.4: First six rows of likelihood evaluation in the extended static model setting establishing EHO factorization. Synthetic EHO data represents a frequently traded stock and is simulated with parameter vector $\theta_{\text{ext}} = (0.2, 0.5, 2200, 2000, 800)$.

where $\log \mathscr{L}\left(\theta_{\text{ext}} \mid \mathscr{M}\right)$ can be accurately calculated as a floating-point number. After applying the transformations shown in equations (4.8a)-(4.8c) on the EHO factorization of the likelihood function, we see that is crucial to calculate the exponential terms without overflow. According to the results in code chunk 4.3, the exponential function does not overflow for values smaller than 710. Therefore we can specify the set BFS_{EHO} more precisely as

$$\text{BFS}_{\text{EHO}} = \left\{ \theta_{\text{ext}} \in \text{BFS} \;\middle|\; \max \left\{ \begin{array}{l} B_d * \log(x_b) + S_d * \log(x_s) \\ -\mu + S_d * \log(x_s) \\ -\mu + B_d * \log(x_b) \end{array} \right\} - M_d * \left(\log(x_b) + \log(x_s)\right) < 710 \right\}, \tag{4.11}$$

where $d \in \{1, 2, \ldots, D\}$.

This set of basic feasible solutions for the EHO likelihood factorization slightly differs from the one by Lin and Ke (2011), which only apply the first relation for the exponential function shown in equation (4.5). Hence, the authors must ensure that each exponential term in equations (4.7a) - (4.7c) does not overflow in evaluations of the likelihood function. This brings an additional condition into their set of basic feasible solutions, namely that $-M_d \cdot \min\left(\log(x_b), \log(x_s)\right)$ needs to be less than the upper bound 710 for every $d \in \{1, 2, \ldots, D\}$.

Computations of equations (4.8a) - (4.8c) are more stable than that of equations (4.7a) - (4.7c), which allows us to calculate the likelihood function for more parameter constellations and higher number of buys and sells.

At the end of this section we can summarize that the EHO factorization is sufficient for stocks with relative small number of daily buys and sells but does not return finite results if the daily number of transactions is large, as it is for the simulated dataset in this section.

4.2.2 Lin and Ke Factorization

A further alternative formulation of the likelihood function is presented in the work by Lin and Ke (2011). The factorization is applicable even for heavily traded stocks. The effectiveness and stability of the likelihood is caused by two principles (see Lin and Ke 2011, p. 629):

- In computing
$$\exp(x)\exp(y) \;\; (\text{or} \;\; x\exp(y)),$$
the expression of
$$\exp(x+y) \;\; (\text{or} \;\; \operatorname{sgn}(x)\exp(\log(|x|)+y))$$
is more stable than that of
$$\exp(x)\exp(y) \;\; (\text{or} \;\; x\exp(y)).$$

- In the computer arithmetic process, the absolute computing error of a function $f(x)$ increases with the absolute value of its first-order derivative.

To fortify the usefulness of these two principles the authors give the following example. Say, one intends to compute
$$\log\big(\exp(x)\exp(y)+\exp(z)\big)$$
with $x = 800$, $y = -400$ and $z = 900$. A threshold for the inputs of exponential function lies at 710, meaning that any larger value gets the exponential function to overflow and return infinite results (see code chunk 4.3).

At first, $\exp(x)\exp(y)$ would lead to an overflow error due to the fact that one input for the exponential is bigger than the threshold ($800 > 710$). Taking the first principle into account, we can compute the expression with $\exp(x+y)$ which gives 5.22147e+173. However, the expression $\exp(z)$ would still produce an infinite value and therefore the expression
$$\log\big(\exp(x+y)+\exp(z)\big)$$
is not computable.

The second principle states that one should avoid large input values for the exponential and small positive input values for the logarithmic function. Hence, the expression
$$m+\log\big(\exp(x+y-m)+\exp(z-m)\big)$$
with $m = \max(x+y, z) = 900$ is more stable and accurate. Irrelevant of the specified values for x, y and z, one of the terms $x+y-m$ and $z-m$ is always zero, whereas the remaining term is always less than 0. This yields small input values for the exponential functions, which in turn always sum up to a value greater than 1. Thus, we have no small positive input values for the natural logarithm. Using x, y and z as specified before we can compute the expression $\log\big(\exp(x)\exp(y)+\exp(z)\big)$ as[40]
$$\begin{aligned} &m+\log\big(\exp(x+y-m)+\exp(z-m)\big) = \\ &900+\log\big(\exp(800-400-900)+\exp(900-900)\big) = \\ &900+\log\big(\exp(-500)+1\big) = 900+7.12458e-218. \end{aligned}$$

[40] We incorporate the built-in *log1p* function which computes $\log(1+x)$ accurately even for $|x| \ll 1$.

Following the two principles mentioned by Lin and Ke (2011), a stable and efficient formulation of the likelihood is given by

$$\begin{aligned}\log \mathscr{L}(\theta_{\text{ext}} \mid \mathscr{M}) = \sum_{d=1}^{D}\Big(-\epsilon_b - \epsilon_s + B_d \log\big(\mu + \epsilon_b\big) + S_d \log\big(\mu + \epsilon_s\big) + e_{\max,d}\Big) + \\ \sum_{d=1}^{D} \log\Big((1-\alpha)\exp\big(e_{1,d} - e_{\max,d}\big) + \alpha(1-\delta)\exp\big(e_{2,d} - e_{\max,d}\big) + \\ \alpha\delta\exp\big(e_{3,d} - e_{\max,d}\big)\Big),\end{aligned} \tag{4.12}$$

where $e_{1,d} = -B_d \log\left(1 + \frac{\mu}{\epsilon_b}\right) - S_d \log\left(1 + \frac{\mu}{\epsilon_s}\right)$, $e_{2,d} = -\mu - S_d \log\left(1 + \frac{\mu}{\epsilon_s}\right)$, $e_{3,d} = -\mu - B_d \log\left(1 + \frac{\mu}{\epsilon_b}\right)$ and $e_{\max,d} = \max\big(e_{1,d}, e_{2,d}, e_{3,d}\big)$. Again, the constant term $-\log(B!S!)$ is dropped.

Similar to the preceding section dealing with the EHO factorization of the likelihood, it is essential to compute the terms involving exponentials without overflow. Analogous to equation (4.11), we can write the set of feasible solutions for the likelihood in equation (4.12) as

$$\text{BFS}_{\text{LK}} = \left\{\theta_{\text{ext}} \in \text{BFS} \;\middle|\; \max\big(e_{1,d} - e_{\max,d}, e_{2,d} - e_{\max,d}, e_{3,d} - e_{\max,d}\big) < 710\right\}, \tag{4.13}$$

with $d = 1, 2, \ldots, D$. Since the three terms $e_{1,d} - e_{\max,d}$, $e_{2,d} - e_{\max,d}$ and $e_{3,d} - e_{\max,d}$ are always weakly smaller than 0, BFS_{LK} is equal to BFS (see equation (4.3)) if the focus lies on the exponential function. Therefore using the likelihood in equation (4.12) reduces floating point exceptions in computations of the likelihood function.

An analogue of the *ehofactr_mat* function for EHO factorization is given by the *linkefactr_mat* function in code chunk 4.13.[41] Both functions for detailed output of likelihood function evaluations share their arguments.

Code Chunk 4.13 (Detailed output of computations utilizing likelihood factorization by Lin and Ke):

```
linkefactr_mat <- function (param = NULL, numbuys = NULL, numsells = NULL) {
  n <- length(numbuys)

  rat1 <- param["mu"]/param["epsilon_s"]
  rat2 <- param["mu"]/param["epsilon_b"]
  rat1log1p <- log1p(rat1)
  rat2log1p <- log1p(rat2)

  const1 <- log(param["mu"] + param["epsilon_s"])
```

[41] Likewise to *ehofactr_mat*, the *linkefactr_mat* function is not part of our **pinbasic** package. It is presented and used for demonstration purposes only. Chapter 5 deals with our package in detail.

```
  const2 <- log(param["mu"] + param["epsilon_b"])

  prob_no <- 1 - param["alpha"]
  prob_good <- param["alpha"] * (1 - param["delta"])
  prob_bad <- param["alpha"] * param["delta"]

  e2 <- -param["mu"] - numsells * rat1log1p
  e3 <- -param["mu"] - numbuys * rat2log1p
  e1 <- e2 + e3 + 2 * param["mu"]
  e_max <- pmax.int(e1, e2, e3)

  part1 <- -param["epsilon_b"] - param["epsilon_s"] +
            numbuys * const2 + numsells * const1 + e_max

  part2 <- log(prob_no * exp(e1 - e_max) + prob_good * exp(e2 - e_max) +
               prob_bad * exp(e3 - e_max))

  ll <- cbind(part1,
              prob_no * exp(e1 - e_max),
              prob_good * exp(e2 - e_max),
              prob_bad * exp(e3 - e_max),
              part1 + part2,
              numbuys,
              numsells)

  colnames(ll) <- c("General Part", "No-News", "Good-News", "Bad-News",
                    "ll_daily", "Buys", "Sells")
  ll
}
```

Using the synthetic EHO dataset from section 4.2.1 which imitates a frequently traded equity, evaluation[42] of Lin-Ke factorization results in all finite values for calculation and equals 1.75833e+06. Results for the first six days are displayed in table 4.5.

With Lin-Ke formulation we have a stable method to estimate PIN even for heavily traded stocks. But the question remains if it is able to handle even very extreme trading data. Consider a really heavily traded stock with trading days for which the total amount of transactions can literally "explode" and easily exceeds 10,000 or more.

For this reason, we simulate new datasets which exhibits really large values for daily buys and sells. The intensities used to simulate daily buys and sells data for an extraordinary frequently traded equity in the EHO setting are $\epsilon_{b,\text{extreme}} = 6600$, $\epsilon_{s,\text{extreme}} = 6000$ and $\mu_{\text{extreme}} = 2400$.[43] The conditions of trading days do not differ from the ones in the previous simulation procedures.

[42] Lin-Ke factorization is evaluated at the corresponding parameter vector which is already utilized in simulation.

[43] The simulated dataset for EHO setting is shipped with our **pinbasic** R package. To access it run `data('BSheavy')` in the console with **pinbasic** package loaded.

General Part	No-News	Good-News	Bad-News	ll_daily	Buys	Sells
2.86099e+04	8.00000e-01	3.48710e-42	1.36659e-55	28609.65850	2279	2009
2.81659e+04	8.00000e-01	2.09665e-57	2.08357e-47	28165.63701	2166	2065
2.77948e+04	8.00000e-01	1.61178e-55	1.29641e-56	27794.52713	2180	2002
3.56132e+04	1.44755e-82	4.21069e-128	1.00000e-01	35610.85291	2248	2937
3.50289e+04	7.79188e-66	1.00000e-01	6.31375e-116	35026.63683	3062	2041
2.83053e+04	8.00000e-01	1.23904e-53	7.20324e-49	28305.12194	2194	2055

Table 4.5: First six rows of likelihood evaluation in the extended static model setting establishing Lin-Ke factorization. Synthetic EHO data representing a frequently traded stock from section 4.2.1 is used for computation.

General Part	No-News	Good-News	Bad-News	ll_daily	Buys	Sells
9.80362e+04	8.00000e-01	1.83275e-152	1.82300e-163	98035.99893	6619	6026
9.73620e+04	8.00000e-01	1.16673e-158	5.38043e-168	97361.75202	6573	5995
9.84504e+04	8.00000e-01	2.59391e-157	3.43195e-151	98450.14447	6583	6110
1.19090e+05	7.74864e-192	0.00000e+00	1.00000e-01	119087.40147	6575	8440
1.19998e+05	5.58503e-175	1.00000e-01	0.00000e+00	119996.02696	9031	6066
9.72588e+04	8.00000e-01	6.57735e-157	1.19568e-171	97258.59688	6586	5970

Table 4.6: First six rows of likelihood evaluation in the extended static model setting establishing Lin-Ke factorization. Synthetic EHO data represents a heavily traded stock and is simulated with parameter vector $\theta_{\text{ext}} = (0.2, 0.5, 6600, 6000, 2400)$.

Computations do not return any `NaN` or infinite values, as can be read from table 4.6. The final result for the likelihood function is given by 6.12494e+06. It is obvious that the Lin-Ke factorization is feasible even for extremely high trading intensities.

4.3 Initial Values

The previous sections 4.2.1 and 4.2.2 give options to evaluate likelihood functions in the EHO model in a stable and effective way. However, we did not have to care about initial values for the evaluations beforehand because we knew the data generating process.

To get the maximum likelihood estimators of the EHO model parameters, the corresponding likelihood function must be maximized. Since the maximum likelihood estimators cannot be derived analytically, one needs to perform numerical optimization with iterative methods. There are plenty of different optimizers available in the R language and the vast majority of them demand a set of initial values from the user. Whether the optimizer converges as well as the corresponding computing time depend to a high degree on the choice of initial values. Moreover, if one uses a *bad* set of starting values, the optimizer may converge and find a maximum, but a local one instead of a global.

There is not *one* rule or algorithm delivering the best starting values which ensure the maximization to end in the global maximum for every optimization run. One possibility would be to perform an appropriate number of maximization runs with different sets of random starting values and then take the run with the highest final function value. This procedure can be cumbersome because it is unclear how many runs are really needed to reach the global maximum, instead of hitting one of possibly several local maxima. Furthermore, running several hundred or even thousands of optimizations can be very time-consuming.

4.3.1 Grid Search Algorithm

Yan and Zhang (2012) present an approach to generate initial values by grid search technique. To determine initial values for the non-probability parameters, ϵ_b, ϵ_s and μ, Yan and Zhang (2012) make use of the marginal distributions of buys B and sells S.[44] The marginal distributions can be written as

[44] More details are given in the appendix of Yan and Zhang (2012).

$$\begin{aligned}\Pr(B = N_B) = (1-\alpha)&\left(\exp(-\epsilon_b)\frac{{\epsilon_b}^{N_B}}{N_B!}\right)\\ &+ \alpha(1-\delta)\left(\exp\big(-(\epsilon_b+\mu)\big)\frac{\big(\epsilon_b+\mu\big)^{N_B}}{N_B!}\right)\\ &+ \alpha\delta\left(\exp(-\epsilon_b)\frac{{\epsilon_b}^{N_B}}{N_B!}\right)\end{aligned} \tag{4.14}$$

and

$$\begin{aligned}\Pr(S = N_S) = (1-\alpha)&\left(\exp(-\epsilon_s)\frac{{\epsilon_s}^{N_S}}{N_S!}\right)\\ &+ \alpha(1-\delta)\left(\exp(-\epsilon_s)\frac{{\epsilon_s}^{N_S}}{N_S!}\right)\\ &+ \alpha\delta\left(\exp\big(-(\epsilon_s+\mu)\big)\frac{\big(\epsilon_s+\mu\big)^{N_S}}{N_S!}\right),\end{aligned} \tag{4.15}$$

with $N_B, N_S \in \mathbb{N}_0$. It is obvious that equations (4.14) and (4.15) are weighted sums of densities of Poisson distributed random variables. We can utilize the linearity of the expectation operator and write the expected values for the marginal distributions as

$$\mathbb{E}(B) = \alpha(1-\delta)\mu + \epsilon_b, \tag{4.16}$$

$$\mathbb{E}(S) = \alpha\delta\mu + \epsilon_s. \tag{4.17}$$

These moment conditions for the expected values are then incorporated in the procedure of determining initial values for all five model parameters.

First step is to get initial values for the probabilities α and δ. To prevent the initial guesses for these two parameters to lie on the boundaries we go with Yan and Zhang (2012) and take a sub-interval of $[0, 1]$. Starting values for α and δ are limited to belong to a series of equidistant real-valued numbers in the range of 0.1 to 0.9, beginning and ending with the minimum and maximum, respectively.[45] In the next step, the sample averages $\overline{B}$ and $\overline{S}$ of the series of daily buys and sells replace the expectations $\mathbb{E}(B)$ and $\mathbb{E}(S)$ in equations (4.16) and (4.17). Since the term $\alpha(1-\delta)\mu$ in equation (4.16) is always positive, ϵ_b needs to be smaller than $\overline{B}$. Analogously, ϵ_s must be smaller than $\overline{S}$ in equation (4.17), since $\alpha\delta\mu$ is always greater than zero.

The average daily number of buys $\overline{B}$ is then multiplied by the same equally spaced series of values which are chosen as initial values for α and δ to generate starting values for the intensity

[45] These bounds are chosen according to the work by Yan and Zhang (2012). In principle, any value reasonably bigger than 0 can be chosen as lower bound and any value reasonably smaller than 1 as upper bound.

of uninformed buyers ϵ_b. The last step is to receive initial values for the intensity of sells initiated by noise traders ϵ_s and the intensity of transaction fulfilled by informed traders μ. Therefore, equations (4.16) and (4.17) are simultaneously solved.

A set of initial values for the EHO model, $\theta^0 = \left(\alpha^0, \delta^0, {\epsilon_b}^0, {\epsilon_s}^0, \mu^0\right)$, can then be calculated as,

$$\begin{aligned}
\alpha^0 &= \alpha_i, \\
\delta^0 &= \delta_j, \\
{\epsilon_b}^0 &= \gamma_k \overline{B}, \\
\mu^0 &= \frac{\overline{B} - {\epsilon_b}^0}{\alpha^0(1-\delta^0)}, \\
{\epsilon_s}^0 &= \overline{S} - \alpha^0 \delta^0 \mu^0,
\end{aligned}$$

where each of the three parameters α_i, δ_j and γ_k take on equally distanced values between 0.1 and 0.9, one at a time. Yan and Zhang (2012) choose the length of the series of initial values for α_i, δ_j and γ_k to be five. Hence, the starting values for each of the three parameters are 0.1, 0.3, 0.5, 0.7 and 0.9. This results in a total of $5^3 = 125$ potential sets of initial values.

However, not all combinations are feasible due to negative values for the intensity of uninformed sells ${\epsilon_s}^0$. In addition, Ersan and Alıcı (2016) recommend to exclude sets of starting values with *irrelevant* values for μ^0 which is the case if μ^0 exceeds the maximum number of daily buys or sells ($\mu^0 > \max(B_d, S_d)$, $d = 1, \ldots, D$). Otherwise, one would pick a *bad* starting value for the intensity of information-based transactions which lies above the largest amount of transactions of one trade direction, either buys or sells, over all trading days under consideration. Assuming that $\max(B_d, S_d) = 1000$ and is realized for buys on a good-news day. Even in the extreme situation that almost all buys are initiated by insiders, a starting value of 1500 for μ is not reasonable.

For demonstration purposes we generate sets of initial values for the synthetic dataset imitating a frequently traded stock from section 4.2.1 with the *initial_vals* function from the **pinbasic** R package and show six exemplary sets in table 4.7.[46] Detailed explanations of the functions the **pinbasic** package is equipped with as well as its source code can be found in chapter 5.

Code Chunk 4.14 (Initial values by grid search):

```
grid_init_demo <- initial_vals(numbuys = BSfrequent[,"Buys"], numsells = BSfrequent[,"Sells"],
                               method = "Grid")
```

[46] In the context of code chunk 5.2 in chapter 5 we give a detailed explanation of the function usage.

α^0	δ^0	${\epsilon_b}^0$	${\epsilon_s}^0$	μ^0
0.90	0.10	228.99	1856.83	2544.30
0.70	0.10	686.96	1907.72	2544.30
0.90	0.10	686.96	1907.72	1978.90
0.90	0.30	686.96	1398.86	2544.30
0.50	0.10	1144.93	1958.60	2544.30
0.70	0.10	1144.93	1958.60	1817.35

Table 4.7: Six exemplary sets of initial values for `BSfrequent` data generated by grid search technique. Infeasible sets of initial values have been deleted.

A total of 86 sets of initial values are removed due to negative starting values for the intensity of uninformed sells or an intensity of transactions initiated by insiders which exceeds the maximum amount of buys or sells in the data. Hence, a small proportion of 31.20% of the potential 125 sets of starting values are feasible for estimating the probability of informed trading.

We can see a similar picture if we increase the number of possible values for α_i, δ_j and γ_k. We allow these initial guesses to take on ten different values which yields a maximum of $10^3 = 1000$ sets of starting values. Only 309 sets of initial values are reasonable and could be used for optimizations.

It seems that the majority of execution time of the grid search technique is dissipated in generating infeasible sets of initial guesses. We will examine this potential disadvantage of the technique in section 4.3.4.

4.3.2 Hierarchical Agglomerative Clustering

A method which utilizes hierarchical agglomerative clustering (HAC) to generate starting values is proposed by Gan, Chun, and Johnstone (2015). Daily order imbalance $\text{OI}_d := B_d - S_d$, $d = 1, \dots, D$, serves as a criterion to assign trading days to three clusters representing no-news, good-news and bad-news condition. HAC is a bottom-up clustering technique in which at the beginning of the algorithm all order imbalances OI_d illustrate a cluster of their own. For instance, if trading data for a quarter of a year is used to estimate the probability of informed trading, roughly 60 clusters exist when the algorithm is initialized.

Gan, Chun, and Johnstone (2015) use the complete-linkage clustering, which is one of several methods of HAC, to sequentially merge small clusters to bigger ones. In general, HAC combines two clusters with the shortest distance, with the definition of *shortest distance* varying with the chosen method.[47]

[47] Gan, Chun, and Johnstone (2015) mention that they tested different agglomerative clustering methods but that the complete-linkage method performed marginally better than the others. For a description of the other available methods, for example, single-linkage or centroid-linkage, see Everitt, Landau, Leese, and Stahl (2011).

For complete-linkage clustering, the distance $\mathrm{D}(X, Y)$ between two clusters X and Y can be written as

$$\mathrm{D}(X, Y) = \max_{x \in X, y \in Y} d(x, y), \tag{4.18}$$

where $d(x, y)$ is the distance between the cluster elements $x \in X$ and $y \in Y$. Gan, Chun, and Johnstone (2015) use the euclidean norm as measure for $d(x, y)$.

The following is a step-by-step instruction how to use the clustering algorithm to generate initial values for the parameters in the EHO model (see Gan, Chun, and Johnstone 2015, p. 1809).

I. Calculate a series of daily order imbalances, OI_d with $d = 1, \dots D$, and use the daily order imbalances, buys and sells as inputs for the following steps.

II. Perform HAC on the daily order imbalances using the complete-linkage clustering.[48] Stop the algorithm when there are three clusters left.

III. The mean of daily order imbalances of the clusters serve as a criterion to assign them to no-news, good-news and bad-news. The cluster with the highest mean, $C_{\mathscr{G}}$, is assumed to consist of good-news trading days. Likewise the cluster with the lowest mean, $C_{\mathscr{B}}$, adheres bad-news trading days. The remaining cluster, $C_{\mathscr{N}}$, is then defined as the no-news cluster.

IV. Compute the average daily buys $\overline{B}_c$ and sells $\overline{S}_c$ for $c \in \{C_{\mathscr{N}}, C_{\mathscr{G}}, C_{\mathscr{B}}\}$ as

$$\overline{B}_c = \frac{\sum\limits_{d \in c} B_d}{\mathrm{card}(\{d \in c\})} \quad \text{and}$$
$$\overline{S}_c = \frac{\sum\limits_{d \in c} S_d}{\mathrm{card}(\{d \in c\})},$$

with $d = 1, 2, \dots, D$ and the card function represents the cardinality of a set. Then, assign each cluster a weight w_c which is calculated as the proportion this cluster occupies of the total number of trading days D. Hence, the cluster weights sum up to 1.

V. With the help of the classification of trading days and the cluster weights from the third and fourth step, we are able to compute initial values for the intensities of uninformed buys and sells as weighted sums of average buys B_c and sells S_c, respectively,

$$\epsilon_b{}^0 = \frac{w_{C_{\mathscr{B}}}}{w_{C_{\mathscr{B}}} + w_{C_{\mathscr{N}}}} \overline{B}_{C_{\mathscr{B}}} + \frac{w_{C_{\mathscr{N}}}}{w_{C_{\mathscr{B}}} + w_{C_{\mathscr{N}}}} \overline{B}_{C_{\mathscr{N}}}$$
$$\epsilon_s{}^0 = \frac{w_{C_{\mathscr{G}}}}{w_{C_{\mathscr{G}}} + w_{C_{\mathscr{N}}}} \overline{S}_{C_{\mathscr{G}}} + \frac{w_{C_{\mathscr{N}}}}{w_{C_{\mathscr{G}}} + w_{C_{\mathscr{N}}}} \overline{S}_{C_{\mathscr{N}}}.$$

[48] According to Gan, Chun, and Johnstone (2015), we use the *hclust* function to perform this task (see Müllner 2013).

For ${\epsilon_b}^0$ we only consider average buys of the no-news and bad-news cluster. On trading days which belong to one of both clusters, buys are solely initiated by uninformed traders. The corresponding weights represent the proportion each cluster occupies of the sum of no-news and bad-news trading days. Analogously, we calculate the initial value ${\epsilon_s}^0$ for the intensity of uninformed sells.

VI. The intensity of informed trading is then calculated as the weighted sum of the intensities of informed buys μ_b^0 and sells μ_s^0 [49] [50]

$$\begin{aligned}
\mu_b^0 &= \overline{B}_{C_{\mathscr{G}}} - {\epsilon_b}^0 \\
\mu_s^0 &= \overline{S}_{C_{\mathscr{B}}} - {\epsilon_s}^0 \\
\mu^0 &= \frac{w_{C_{\mathscr{G}}}}{w_{C_{\mathscr{G}}} + w_{C_{\mathscr{B}}}} \mu_b^0 + \frac{w_{C_{\mathscr{B}}}}{w_{C_{\mathscr{G}}} + w_{C_{\mathscr{B}}}} \mu_s^0,
\end{aligned}$$

where μ_b^0 and μ_s^0 represent the buys and sells initiated by informed traders on good-news and bad-news trading days, respectively. The weights in μ^0 are then set according to the occurrence of these two trading states if concentrating on trading days on which private information enters the market.

VII. Cluster sizes are utilized to compute starting values for the probability of an information event α and the probability of bad news given that private information enter the market δ,

$$\begin{aligned}
\alpha^0 &= w_{C_{\mathscr{G}}} + w_{C_{\mathscr{B}}} \\
\delta^0 &= \frac{w_{C_{\mathscr{B}}}}{\alpha^0}.
\end{aligned}$$

VIII. An initial estimate of the probability of informed trading is given by

$$\text{PIN} = \frac{\alpha^0 \mu^0}{{\epsilon_b}^0 + {\epsilon_s}^0 + \alpha^0 \mu^0}.$$

In contrast to the brute force grid search method discussed in the previous section, the HAC algorithm returns only a single vector of initial values. Furthermore, no computation time is spent in generating infeasible sets of values which are then immediately discarded.

Likewise to the previous section, we generate the set of starting values with the help of our *initial_vals* function using the identical dataset as we did for the grid search. The results are show in table 4.8.

[49] A splitting of the intensity of informed trading is not present in the EHO model. However, one could extend the existing model with this feature and already had a suitable technique for generating initial values.

[50] It is not ensured that μ_b^0 and μ_s^0 are positive. Hence, if μ_b^0 or μ_s^0 are negative we set them to 0.

Code Chunk 4.15 (Initial values by HAC algorithm):

```
cluster_init_demo <- initial_vals(numbuys = BSfrequent[,"Buys"],
                                  numsells = BSfrequent[,"Sells"],
                                  method = "HAC")
```

The results of the complete-linkage clustering (Step *II*) are visualized with a dendogram in figure 4.3.[51] The good-news and bad-news cluster each consist of 6 trading days. The remaining

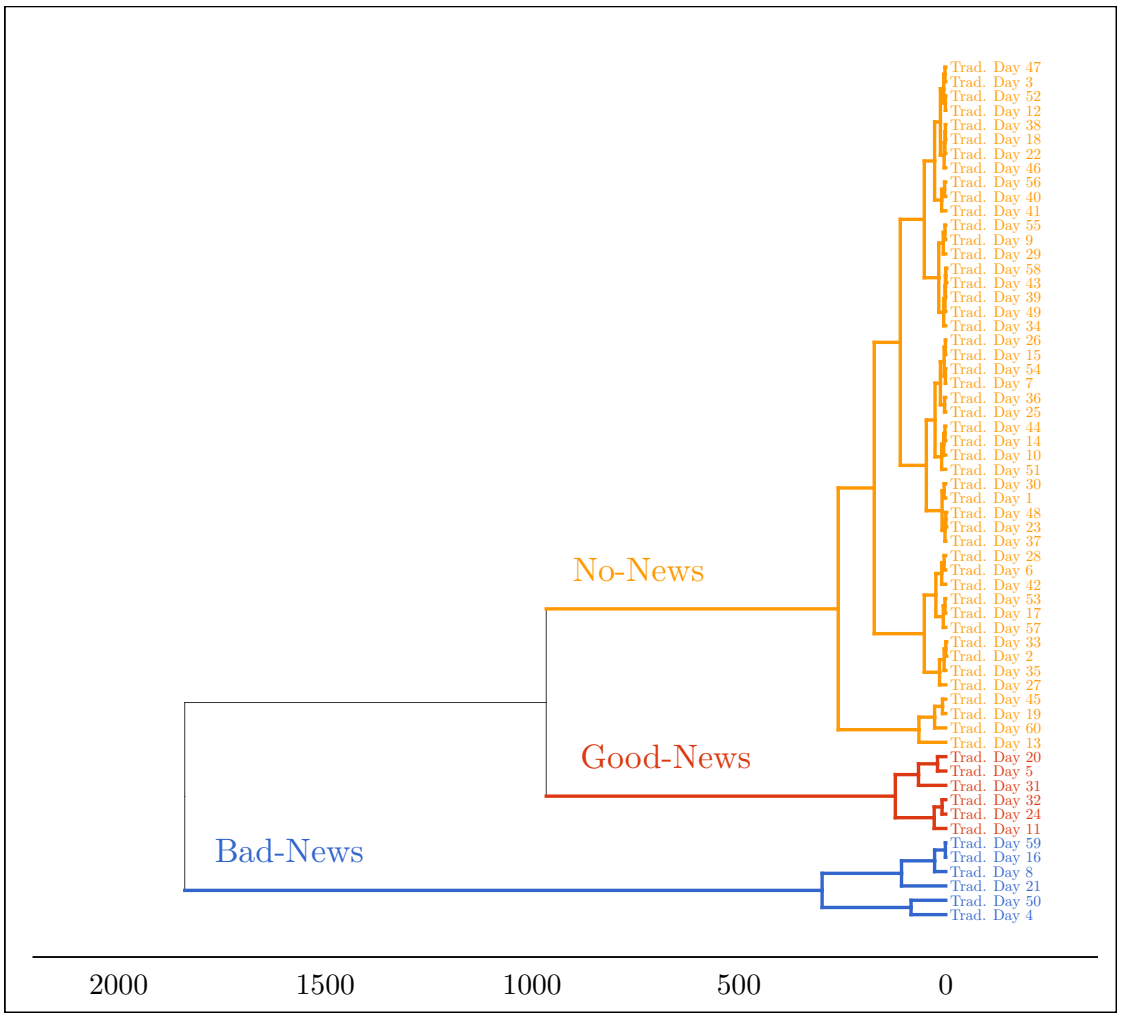

Figure 4.3: Dendogram of HAC complete-linkage clustering for `BSfrequent` data.

[51] To generate the dendogram plot, we utilize the object returned by the *hclust* function. The function stops if all observations are merged into one big cluster. We then utilize the *cutree* function from the **stats** package to *cut* the returned object from *hclust* into three groups representing no-news, good-news and bad-news clusters.

48 trading days are marked as no-news days by the algorithm. Hence, the weights for the no-news, good-news and bad-news cluster are given by $w_{C_\mathcal{N}} = 0.80$, $w_{C_\mathcal{G}} = w_{C_\mathcal{B}} = 0.10$, respectively.

α^0	δ^0	${\epsilon_b}^0$	${\epsilon_s}^0$	μ^0
0.20	0.50	2210.44	2001.96	816.38

Table 4.8: Set of initial values for `BSfrequent` data generated by HAC algorithm.

The results of the algorithm proposed by Gan, Chun, and Johnstone (2015) are pretty close to the true parameters for every entry. The initial estimate of the probability of informed trading equals 0.04 and the true probability of informed trading is given by 0.04. Hence, the starting values generated by HAC algorithm seem to be a good choice for optimization. We will further investigate the performance of the technique in section 4.3.4.

4.3.3 Refined HAC Algorithm

A third option for generating initial values for maximization of the likelihood function in the EHO model was presented by Ersan and Alıcı (2016). Similar to the algorithm discussed in section 4.3.2, hierarchical agglomerative clustering is utilized for generating starting values. However, instead of stopping the HAC algorithm once three clusters are left, the number of groups is not fixed. The refined HAC algorithm is stopped when $j + 1$ clusters are left, whereat j is a positive integer with $j \leq D - 1$[52] and D denotes the number of total trading days.

In contrast to the HAC algorithm by Gan, Chun, and Johnstone (2015), daily absolute order imbalance is used to assign trading days to one of the $j+1$ groups. The clusters are then ordered by their average absolute order imbalance and distributed to a *no-event* and *event* cluster. To achieve multiple vectors of starting values instead of a single one, the two types of groups are build as

$$CL_i^{\mathcal{N}} = \bigcup_{k=1}^{i} CL_k,$$

$$CL_i^{\mathcal{E}} = \bigcup_{k=i+1}^{j+1} CL_k,$$

where $i = 1, \ldots, j$ and $CL_i^{\mathcal{N}}$ and $CL_i^{\mathcal{E}}$ represent the no-event cluster and event cluster. At this point, we are able to obtain initial values α^0 and μ^0. The initial guess for the probability of an information event is calculated, in a very similar way to the HAC algorithm by Gan, Chun, and Johnstone (2015), as the proportion the event cluster $CL_i^{\mathcal{E}}$ occupies of the total number of

[52] In the work by Ersan and Alıcı (2016) j was chosen to be an integer in the range from 1 to 10.

trading days D. The initial intensity of informed trading equals the difference in averages of the absolute order imbalances of no-event and event group.[53]

In the next step, $CL_i^{\mathscr{E}}$ is split according to the signs of the average order imbalances of its members in a good-news and bad-news group. The remaining starting values for the two intensities ϵ_b and ϵ_s and the probability δ are calculated according to section 4.3.2.

Hence, we get a total of j vectors of initial values. MLE is performed for each of the j vectors and the best result among all maximization runs is kept while discarding the remaining. Again, we use the simulated dataset from section 4.2.1 for demonstration. According to the results in the work by Ersan and Alıcı (2016) a value of $j = 5$ is a good compromise between accuracy and speed.

Again, we use the `BSfrequent` dataset and compute initial values by calling the *initial_vals* functions from our **pinbasic** package. The results are shown in table 4.9.

Code Chunk 4.16 (Initial values by refined HAC algorithm):

```
# 'num_clust' arguments takes the number 'j' of clusters
hac_ref_demo <- initial_vals(numbuys = BSfrequent[,"Buys"],
                             numsells = BSfrequent[,"Sells"],
                             method = "HAC_Ref",
                             num_clust = 5)
```

α^0	δ^0	${\epsilon_b}^0$	${\epsilon_s}^0$	μ^0
0.83333	0.12000	2192.06250	2001.96296	237.66000
0.26667	0.37500	2204.94000	2001.96296	485.36932
0.20000	0.50000	2210.44444	2001.96296	594.06250
0.13333	0.25000	2210.44444	2056.03448	694.75962
0.10000	0.00000	2210.44444	2085.81667	741.94444

Table 4.9: Sets of initial values for `BSfrequent` data generated by refined HAC algorithm. MLE is performed for each of the sets and best result is kept.

At the end of this section we want to mention a critical aspect in the design of this method. Dividing trading periods in a no-event and an event cluster according to their absolute order imbalance is a potential pitfall. Assuming a parameter constellation such that the intensity of uninformed sellers is approximately equal to the sum of the intensity of uninformed buyers and the intensity of informed traders, more precisely: $\epsilon_s \approx \epsilon_b + \mu$. Trading days on which

[53] In the work by Ersan and Alıcı (2016) this relation is not directly mentioned. However, at this point in the algorithm , we solely have information about the two groups of trading days and the absolute order imbalance at hand. Since the EHO model does not separate informed buy rate from informed sell rate, the difference in averages of the absolute order imbalances of no-event and event group is appropriate to capture the intensity of trading due to private information.

informed traders buy equities then show an absolute order imbalance, $|\overline{B} - \overline{S}|$, close to 0 and will be erroneously assigned to the no-event cluster. Bad-news and no-news trading periods exhibit absolute order imbalances of approximately 2μ and μ, respectively, and will belong to the event cluster. Therefore we will get the boundary solution $\delta = 1$ for the initial value of the probability of bad-news days. The same problematic arises if the intensity of uninformed buyers is very close to the sum of the intensity of uninformed sellers and the intensity of transactions initiated by insiders. Following the argumentation just described, the algorithm by Ersan and Alıcı (2016) will return the boundary solution of 0 as initial guess for δ.

4.3.4 Simulation Study

With our simulation study in this section we investigate the quality of the results of the algorithms for generating initial values presented in the previous sections 4.3.1 – 4.3.3. This section must not be misinterpreted that the grid search and HAC method are two different estimators for the PIN variable. They are two methods to generate initial values for the optimization of the same likelihood function.

The main question we want to answer with our simulation study is if it is necessary to conduct several optimizations of the likelihood function with different sets of starting values as it is done by the grid search and refined HAC methods, or if it sufficient to even use only one set of starting values as proposed by the HAC approach. A possible decrease in the number of optimization runs will reduce the execution time to estimate the probability of informed trading.

At first, we explain the two simulation studies in the papers by Gan, Chun, and Johnstone (2015) and Ersan and Alıcı (2016), which differ in several aspects from our simulation procedure. In the former the five parameters of the EHO model are determined by four parameters $a, b, c, d \in (0, 1)$ according to:

$$\alpha = a, \delta = b,$$
$$\epsilon_b = (1 - c)dk, \epsilon_s = (1 - c)(1 - d)k, \mu = ck,$$

where k represents the total trading activity, which is fixed at 2500 for each simulated dataset. Overall, Gan, Chun, and Johnstone (2015) simulate 1000 datasets of daily buys and sells, whereas each has a length of 60 trading days. The probability a of an information event and the conditional probability of bad-news b are independent random variables following a standard uniform distribution. The remaining simulation parameters c and d are the percentage of informed market participants and the percentage of buyer-initiated transactions among the trades initiated by uninformed market attendees, respectively. Both parameters, c and d, are also drawn independently from a standard uniform distribution. The results show that maximum likelihood estimation utilizing the HAC algorithm offers the same accuracy as the brute force grid search algorithm in terms of the mean absolute error. However, the HAC method is much faster. This simulation incorporates different intensities for noise trading but allows for a very

large difference in the trading rates. For real data, it is a reliable assumption that the frequencies of buys and sells are relatively close to each other for the majority of trading days.

In the simulation study by Ersan and Alıcı (2016) the total trading intensity k is also set to 2500 for all trading days. The number of total transactions is distributed to noise trading and informed trading with the help of one parameter, p_1, which again follows a standard uniform distribution. The noise trading intensities, ϵ_b and ϵ_s, are assumed to be identical,

$$\mu = p_1 k, \epsilon_b = \epsilon_s = (1 - p_1)\frac{k}{2}.$$

Ersan and Alıcı (2016) increase the number of simulation runs to 5000, whereas the amount of trading days for each dataset is not changed in comparison to the simulation study by Gan, Chun, and Johnstone (2015). According to Ersan and Alıcı (2016), conditions of trading days are determined by using two sequences of binomial values, each with a length of 60. One decides about the occurrence of an information event, the other about its direction. The actual condition of a trading day is then given as the product of the corresponding values in both series. There are no-news days with probability p_2 and information events with probability $(1 - p_2)$. Likewise, on information events, the direction of private news is negative with probability p_3 and positive with probability $(1 - p_3)$. Similar to p_1, the probabilities p_2 and p_3 are independently drawn from a standard uniform distribution.

Due to the restrictive assumption for the intensity parameters of noise trading, the simulation design by Ersan and Alıcı (2016) does reflect the setting in the simple EKOP model. However, the restriction is not mentioned in the presentation of the concept, as can be seen in Ersan and Alıcı (2016).

Our simulation study increases the total number of simulation runs dramatically. We simulate 100,000 datasets of daily buys and sells and stick to the common length of 60 trading days per dataset. In contrast to the previous mentioned simulation procedures, we do not fix the total trading intensity to a certain level over all datasets. We assume that it is uniformly distributed in the range from 100 to 10,000. With this less restrictive assumption, we intend to capture the performance and quality of the different algorithms for infrequently traded as well as for really heavily traded stocks.

The determination of the intensity parameters is done in two steps. Firstly, we split the amount of total transactions in a noise trading and an information-based part. Therefore, for simulation run i, we uniformly draw a value which lies in the range from 10% to 90% of the total trading intensity assigned to the i-th dataset (TT_i). This value is classified as the noise trading intensity for the i-th run (NT_i). The intensity of the informed market participants is then given as the difference between TT_i and NT_i. In the second step, we distribute the noise trading intensity to uninformed buyers and uninformed sellers. We agree with Ersan and Alıcı (2016) that the disparity between uninformed buys and sells should be limited and relatively small. To incorporate this into our simulation procedure we make use of a jitter parameter γ which is drawn from a uniform distribution in the interval $[-0.1, 0.1]$ in each simulation run. The i-th intensity of uninformed buys $\epsilon_{b,i}$ is set to $\mathrm{NT}_i \cdot (0.5 + \gamma)$ and accordingly the i-th intensity of uninformed

sells $\epsilon_{s,i}$ is computed as $\text{NT}_i - \epsilon_{b,i}$. This ensures that the proportion of noise trading each of the intensities of uninformed traders occupies lies between 40% and 60%. The probability parameters α and δ are determined the same way as in Gan, Chun, and Johnstone (2015) and are drawn from a standard uniform distribution.

For certain constellations of the model parameters exists an identification problem. This is the case if only one unique condition for all trading days in the run is drawn or if there are no insiders in the market ($\mu = 0$). Assuming we have solely one trading state in simulation run i, then the buys B_d and sells S_d are iid Poisson distributed with parameters $\epsilon_{b,i}$ and $\epsilon_{s,i}$ and only these two distribution parameter are identifiable and not the set of model parameters and the probability of informed trading.

As one can see, the following parameter sets could all generate identical distributions of daily buys and sells:

$$\text{Only no-news: } \left\{\left(\alpha = 0, \delta, \epsilon_b = \epsilon_{b,i}, \epsilon_s = \epsilon_{s,i}, \mu \mid \delta \in [0,1], \mu > 0\right)\right\} \tag{4.19}$$

$$\text{Only good-news: } \left\{\left(\alpha = 1, \delta = 0, \epsilon_b = \epsilon_{b,i} - \mu, \epsilon_s = \epsilon_{s,i}, \mu \mid \mu \in \left[0, \epsilon_{b,i}\right]\right)\right\} \tag{4.20}$$

$$\text{Only bad-news: } \left\{\left(\alpha = 1, \delta = 1, \epsilon_b = \epsilon_{b,i}, \epsilon_s = \epsilon_{s,i} - \mu, \mu \mid \mu \in \left[0, \epsilon_{s,i}\right]\right)\right\} \tag{4.21}$$

$$\text{No informed activity: } \left\{\left(\alpha, \delta, \epsilon_b = \epsilon_{b,i}, \epsilon_s = \epsilon_{s,i}, \mu = 0 \mid \alpha, \delta \in [0,1]\right)\right\} \tag{4.22}$$

Due to the explanations above and the expressions in equations (4.19) – (4.22), we restrict the range of true values of α, δ and μ to prevent our simulation from identification problems.[54] The two probability parameters are uniformly drawn from the interval $[0.1, 0.9]$. According to the restricted range of noise trading in each simulation run mentioned above, the minimum possible value for the intensity of informed trading equals 10% of the total trading intensity and the maximum value cannot exceed 90% of the total trading intensity. If there is still only one type of trading days in a simulation run, we discard this sequence of conditions. The states of trading days are then simulated again until there are at least two different types of trading days.

For each simulated dataset we run MLE with Lin-Ke factorization and initial values generated by the different algorithms discussed in this work. We utilize the *nlminb* function from the **stats** package for all likelihood maximizations in our simulation study which is a bounds constrained quasi-Newton method, as Nash (2014) states. The optimizer is part of the *PORT* library by Bell Laboratories (see Gay 1990). We implemented the *pin_est_core* function in the **pinbasic** package which can be harnessed, in combination with the *initial_vals* function mentioned in the previous sections, to perform the optimizations for the three methods for generating initial values.[55]

[54] With the restricted range for the parameters α, δ and μ there are no sets of parameter estimates which fit any of the expressions in (4.19) – (4.22) for all approaches.

[55] A detailed description of the usage of the *pin_est_core* function is given in chapter 5. Its source code can be found in code chunk 5.6.

The refined HAC algorithm delivers 2395 sets of initial values for which an optimization is not possible due to infinite likelihood function values. In such cases a warning is printed to the console and *nlminb* returns the starting values as *best* set of parameters.

Exemplary, consider one simulated dataset with summary statistics given in table 4.10. Corresponding sets of starting values which are returned by the refined HAC algorithm can be found in table 4.11. The data for this simulation run is generated according to the parameter set given in table 4.12.

	Buys	Sells
Min.	496	568
1st Qu.	515	628
Median	533	5320
Mean	937	3333
3rd Qu.	553	5427
Max.	5477	5576

Table 4.10: Summary statistics of a simulated dataset of daily buys and sells for which the refined HAC algorithm returns only infeasible vectors of initial values. Evaluations of the likelihood at those sets yield infinite function values.

α	δ	ϵ_b	ϵ_s	μ
0.85000	1.00000	937.46667	598.77778	3679.99346
0.65000	1.00000	937.46667	622.90476	4762.16484
0.60000	1.00000	937.46667	1004.45833	4206.19444
0.41667	1.00000	937.46667	1961.25714	2942.00571
0.10000	1.00000	937.46667	3091.37037	1993.27778

Table 4.11: Sets of initial values achieved with refined HAC method for simulated dataset presented in table 4.10.

α	δ	ϵ_b	ϵ_s	μ
0.5937	0.8362	531	623	4813

Table 4.12: Parameter set which is used to simulate daily buys and sells data for which optimization is not possible if initial values generated by the refined HAC algorithm are incorporated.

We will pick one trading day of the simulated dataset and demonstrate why starting values displayed in table 4.11 yield infinite likelihood function values. On the sixth trading day buyer-initiated transactions dominate the market with a total number of 5413 compared to a very small amount of sells which equals 630. Values of e_1, e_2 and e_3 in the Lin-Ke factorization are shown in table 4.13.

Due to the initial guesses for δ equaling 1 in every vector of starting values the (numerical)

e1	e2	e3
-7391.67509	-4918.90775	-12310.58284
-8411.42805	-6121.06972	-14532.49776
-8177.62026	-5243.33618	-13420.95643
-7110.66045	-3519.28429	-10629.94475
-5856.40758	-2306.77293	-8163.18052

Table 4.13: Values of e_1, e_2 and e_3 in Lin-Ke likelihood factorization evaluated at initial values displayed in table 4.11. Data for buys and sells of the eighth trading day of the simulated data is used. Maximum values are highlighted in red.

result of the expression

$$(1-\alpha)\exp(e_1 - e_{\max}) + \alpha(1-\delta)\exp(e_2 - e_{\max}) + \alpha\delta\exp(e_3 - e_{\max})$$

is 0 for each row in table 4.11. Since $1-\delta$ is identical to 0, the term $\alpha(1-\delta)\exp(e_2 - e_{\max})$ is also 0 irrespective of e_2 and $e_{\max}$. The two remaining expressions, $(1-\alpha)\exp(e_1 - e_{\max})$ and $\alpha\delta\exp(e_3 - e_{\max})$ are 0 due to underflows of the exponential function. From table 4.13 we see that values in the e_2-column are the maximum in each row. Hence, to evaluate the likelihood, we need to compute $e_1 - e_{\max} = e_1 - e_2$ and $e_3 - e_{\max} = e_3 - e_2$ with maximum values of -2290.35833 and -5856.40758, respectively. Using this large negative values as arguments for the exponential function lead to underflows. Therefore, computations of the term log(0) are involved in likelihood function evaluations and induce infinite function values. The remaining algorithms for generation of initial values do not suffer from infinite likelihood function values and the amount of simulated datasets for which the refined HAC algorithm returns infeasible starting values is not negligible. Additionally, there is the potential pitfall of this algorithm due to dividing the trading days in no-event and event clusters according to their absolute order imbalance as described in section 4.3.3. Therefore we decided to exclude this approach from further analyses.

For two synthetic datasets the grid search technique generates 107 infeasible sets of starting values so that only 18 out of 125 possible sets can be used for optimizations. In the remaining simulation runs less sets must be deleted. The clustering method by Gan, Chun, and Johnstone (2015) also produces feasible initial values for all simulated datasets of daily buys and sells incorporated in our simulation study.

For the vast majority of optimizations, initial values delivered by grid search and HAC method achieve very similar final likelihood function values and model parameter estimates. The former yields (slightly) higher likelihood values in 99596 simulation runs, the latter in 369, for 35 runs results are identical. However, by taking the maximum final likelihood function value in each run and computing the mean difference for both algorithms, it is obvious that the overall performance is very similar. Grid search and HAC algorithm gain values of 1.8372e-10 and 2.6874e-01, respectively.

Furthermore, we calculate mean errors as well as mean absolute errors for the probability of informed trading and the five model parameters. The results are shown in tables 4.14 and 4.15, respectively. We use two-sided Student's t-tests to check for the significance of (absolute) mean errors.[56]

	Grid Search	HAC
α	-0.00028	-0.00026
δ	0.00013	0.00013
ϵ_b	-0.02386	-0.01141
ϵ_s	-0.00837	-0.00543
μ	-0.01442	-0.29729$\star$
PIN	-0.00155$\star$	-0.00156$\star$

Table 4.14: Mean errors of algorithms for generating initial values in the EHO setup. Errors are computed as the difference between parameter estimates, which are returned by MLE incorporating Lin-Ke factorization and either grid search or HAC method, and actual values. Significance is checked with two-sided t-tests, whereat $\star$ signals significance at the 5% level.

The mean errors of α and δ as well as those of the intensity of noise trading, ϵ_b and ϵ_s, do not significantly differ from 0 for initial values by the grid search and HAC method. Additionally, we see that all trading intensities are on average underestimated, with the mean error for the intensity of informed trading being significant if initial guesses by HAC are utilized.

However, the focus lies on the quality of estimates of the probability of informed trading. Comparing the entries for the bias of the estimates of PIN shows that the mean error is significant for all methods. Values are very small and nearly identical but the method by Yan and Zhang (2012) performs little better. Further information we can extract from table 4.14 is that PIN estimates are significantly lower than actual values for both approaches for generating sets of initial values.

Mean absolute errors for all model parameters and the probability of informed trading display significance, as can be seen from table 4.15. In addition, it is obvious that the values for the mean absolute errors for both methods are on a very similar level. The comparison of HAC and grid search in terms of mean absolute errors confirms the good performance of the clustering algorithm.

Similar to the simulation study by Ersan and Alıcı (2016) we analyze the occurrence of estimates with a large bias (see table 4.16). We assign a large bias to the estimates of the probability parameters α and δ in the i-th simulation run if the magnitude of the difference between estimate and true value exceeds 0.25.

$$|\widehat{\alpha}_i - \alpha_i| > 0.25 \text{ and } |\widehat{\delta}_i - \delta_i| > 0.25$$

[56] For the various significance tests we use the built-in function *t.test*.

	Grid Search	HAC
α	0.04537⋆	0.04539⋆
δ	0.07319⋆	0.07319⋆
ϵ_b	3.88221⋆	3.89756⋆
ϵ_s	3.87681⋆	3.88366⋆
μ	10.31550⋆	10.60065⋆
PIN	0.01956⋆	0.01957⋆

Table 4.15: Mean absolute errors of algorithms for generating initial values in the EHO setup. Absolute errors are computed as the magnitude of the difference between the parameter estimates, which are returned by MLE incorporating Lin-Ke factorization and either grid search or HAC method, and actual values. Significance is checked with two-sided t-tests. Likewise to table 4.14, ⋆ signals significance at the 5% level.

The noise trader intensity estimates have a large bias if the absolute difference is larger than 1% of the total trading intensity in the i-th run.

$$|\widehat{\epsilon_{b,i}} - \epsilon_{b,i}| > 0.01 \cdot \mathrm{TT}_i \quad \text{and} \quad |\widehat{\epsilon_{s,i}} - \epsilon_{s,i}| > 0.01 \cdot \mathrm{TT}_i$$

Likewise, the intensity of informed trading has a large bias if the absolute difference is larger than 2% of the total trading intensity in the i-th run.

$$|\widehat{\mu_i} - \mu_i| > 0.02 \cdot \mathrm{TT}_i$$

Since PIN is a probability parameter like α and δ, we apply the same rule for indicating estimates with a large bias.

$$|\widehat{\mathrm{PIN}}_i - \mathrm{PIN}_i| > 0.25$$

By analyzing the results for large biases in table 4.16 we see again the similarity of the grid search and HAC approach. The parameter for the probability of bad news exhibit the highest rate of large bias for both approaches with little more than 2%. About 0.01% of the PIN estimates deviate more than 0.25 in magnitude from the true values if initial values are generated by either grid search or HAC.

Figures 4.4 and 4.5 display heatmap-like plots in which model parameter estimates as well as the probability of informed trading, resulting from MLE with either initial values by grid search or HAC, are plotted against their corresponding actual values. Darker areas represent higher concentration of points. For the probability parameters α and δ plots look very similar for grid search and HAC method. We see higher concentration of estimates in a more or less thin band around the hypothetical red line which represents points for which estimates are identical to actual values. Furthermore, the graphs exhibit no systematic variation.

Estimates of the noise trading intensities exhibit high concentration around red lines for both algorithms. We also see a high concentration of estimates around the red line in the plots which belong to the intensity of informed trading. However, initial values by HAC yield in

	Grid Search	HAC
α	0.006	0.008
δ	2.197	2.197
ϵ_b	0.231	0.233
ϵ_s	0.242	0.242
μ	0.544	0.559
PIN	0.009	0.011

Table 4.16: Summary of frequencies (in %) of parameter estimates marked with large bias. The rules for assigning a large bias to an estimate are: $|\widehat{\alpha} - \alpha| > 0.25, |\hat{\delta} - \delta| > 0.25, |\widehat{\epsilon_b} - \epsilon_b| > 0.01 \cdot$ Total Trading Intensity,
$|\widehat{\epsilon_s} - \epsilon_s| > 0.01 \cdot$ Total Trading Intensity, $|\hat{\mu} - \mu| > 0.02 \cdot$ Total Trading Intensity,
$|\widehat{\text{PIN}} - \text{PIN}| > 0.25$.

some simulation runs estimates of the intensity of informed trading which are far too low. The estimates resulting from initial values by the grid search do not exhibit such behavior.

Trading days are always clustered in three groups representing the possible conditions, $\mathscr{N}$, $\mathscr{G}$ and $\mathscr{B}$, by HAC. However, there are 2252 simulation runs in which only two types of trading days are realized and therefore there are actually only two clusters. The HAC method will then randomly split one into two sub-clusters. Hence, the initial value for the intensity of informed trading, which is calculated as the weighted sum of the intensity of informed buys on days in the good-news cluster and informed sells in the bad-news cluster[57], is (far) too low. However, the optimizer used in this work, *nlminb*, seems to be able to handle such starting values for μ in most situations.

To give an example, in one simulation run the true values for α and δ are about 0.17 and 0.16, respectively. The sequence of states in the corresponding simulation run solely consists of 50 no-news and 10 good-news trading days. The actual value for the intensity of informed trading equals 456. The approach by Gan, Chun, and Johnstone (2015) returns initial values for the two probability parameters α and δ of about 0.95 and 0.87, respectively. The two initial values represent a high occurrence of bad-news trading days. Hence, good-news days are split in a good-news and no-news cluster and the actual no-news days are labelled as bad-news. This results in a starting value for μ which equals 60. The estimates returned by the `nlminb` function for α, δ and μ are 0.16, 0 and 437, respectively.

The plots for the probability of informed trading look very similar again. One can think of a funnel which is wider for mid-ranged actual values of PIN and gets thinner in the corners. Optimizations with initial values by HAC yield some more estimates of PIN which are far too low (e.g., we can see estimates around 0 when the true value lies around 0.25). As described above the method by Gan, Chun, and Johnstone (2015) struggles to generate *good* initial values for μ if there are only two different labels in the sequence of trading days' conditions in a

[57] See section 4.3.2 for the corresponding expressions.

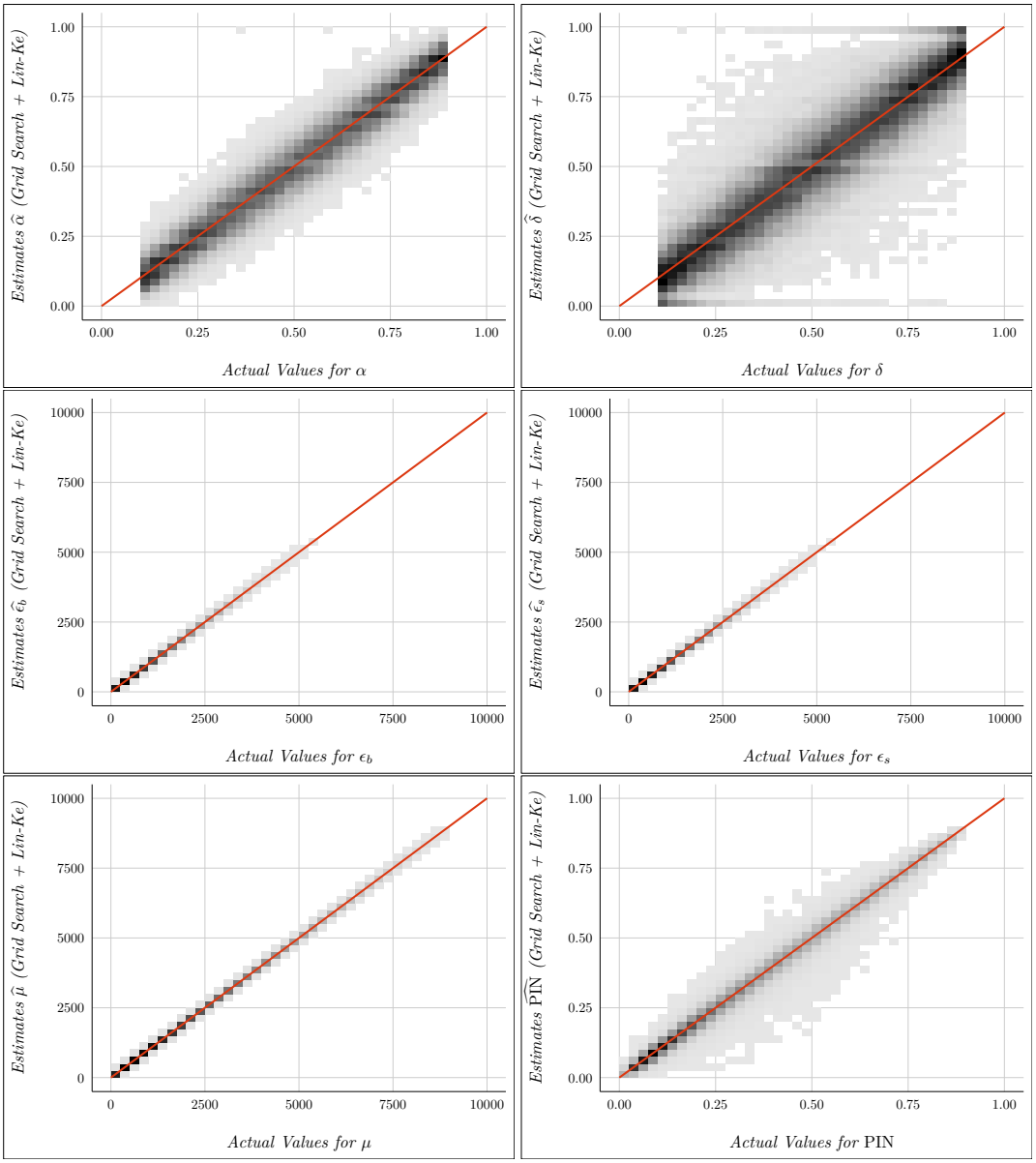

Figure 4.4: Estimates of all model parameters and the probability of informed trading, received by MLE + brute force grid search, plotted against actual values. Red lines represent hypothetical points for which estimates are identical to actual values. Darker areas display higher concentration of points. Binwidths are 0.025 in vertical and horizontal direction for parameters α, δ and PIN and 250 for the remaining intensity parameters.

simulation run. If the optimizer is not able to find solution with a substantially increased value for $\hat{\mu}$, estimates of the probability of informed trading are in consequence also too low.

In summary, the various results in our simulation study show that the performance of both

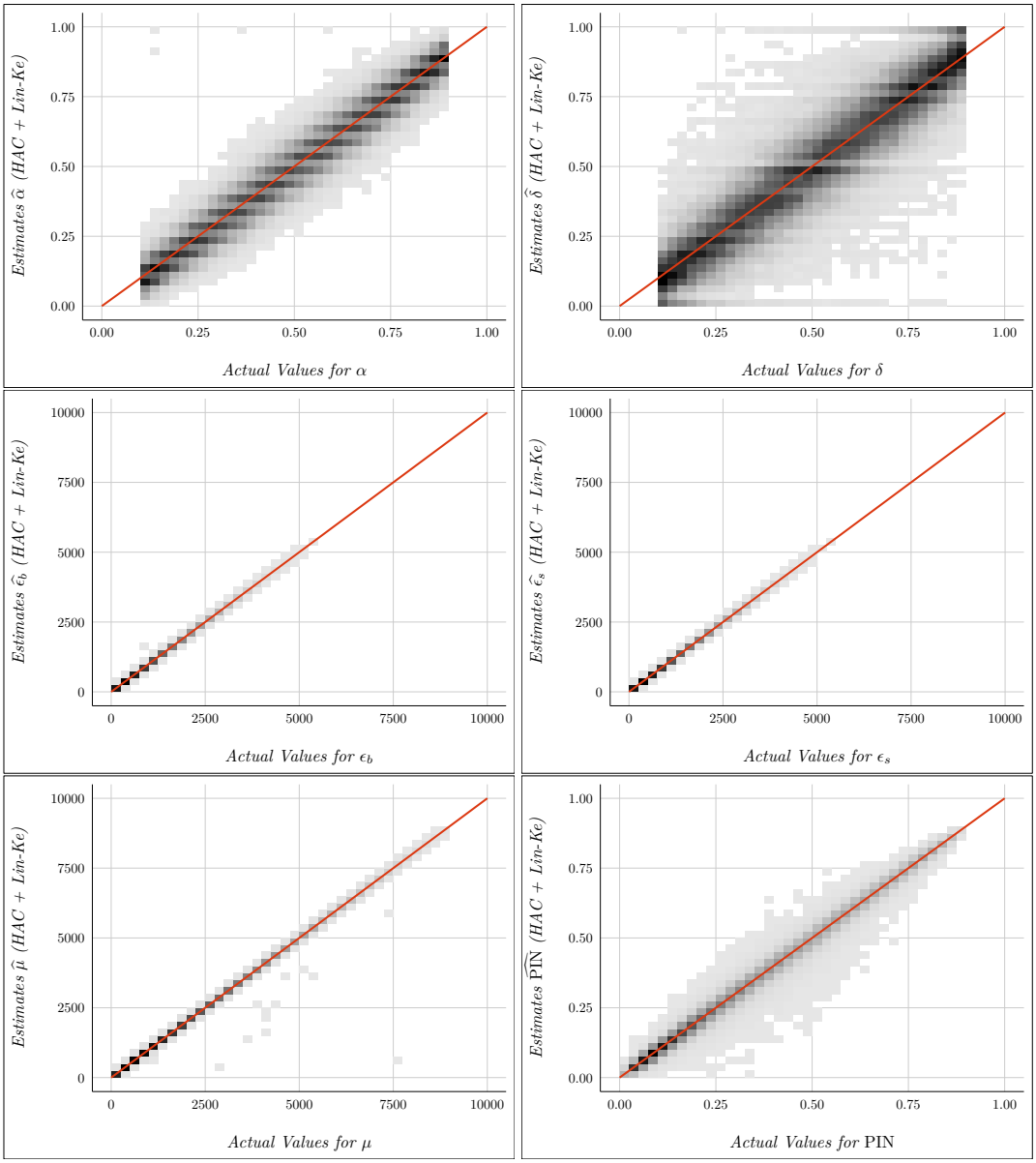

Figure 4.5: Estimates of all model parameters and the probability of informed trading, received by MLE + HAC algorithm, plotted against actual values. Red lines represent hypothetical points for which estimates are identical to actual values. Darker areas display higher concentration of points. Binwidths are 0.025 in vertical and horizontal direction for parameters α, δ and PIN and 250 for the remaining intensity parameters.

algorithms for the generation of starting values is very similar. Due to the nearly identical values of the (absolute) mean error for the probability of informed trading and the tremendous reduction in computing time (one set of initial values and one optimization run vs at most 125 sets of initial values and optimizations), we suggest the clustering algorithm by Gan, Chun,

and Johnstone (2015) as reference. The results in this section show that it is sufficient to use only one set of initial values.

Hence, the most stable and efficient method for estimating PIN in static models is a combination of the likelihood factorization by Lin and Ke (2011) and the HAC algorithm to generate sets of starting values by Gan, Chun, and Johnstone (2015).

Furthermore, as mentioned earlier in section 4.3.1, we investigate the frequency of non-feasible combinations of initial values generated by brute force grid search algorithm by calculating sets of initial values for 100,000 simulated datasets. We assume that the parameters α_i, δ_j and γ_k can each take on five different values, $0.1, 0.3, 0.5, 0.7, 0.9$, which is common practice in the literature. The results of the simulation fortify our previous suggestion. On average, 68.22 combinations of starting values exhibit a negative value for ϵ_s or possess an intensity of informed trading which is too large to be reasonable for the underlying data. Only 45.42% of calculated sets are feasible for optimization purposes.

4.4 Confidence Intervals

To the best of our knowledge, we are the first to present confidence intervals for the probability of informed trading. We conduct calculations of those confidence intervals with the help of Monte-Carlo simulation techniques by utilizing model parameter estimates to simulate n datasets of daily buys and sells according to the procedure presented in code chunks 4.6 and 4.7 in section 4.1.2.

We then perform optimizations for every simulated dataset to receive parameter estimates and use the results to compute $\widehat{\text{PIN}}$. Hence, finally we receive a sequence of n PIN estimates for whose empirical distribution we can calculate any quantile and therefore are in the position to return confidence intervals with any desired level. This procedure is similar to the *type 2 Monte Carlo intervals* as described in the work by Buckland (1983).

The computation of confidence intervals benefits to a huge degree from the computing efficiency in estimations of the probability of informed trading. On the hand, there is a very stable factorization of the likelihood function by Lin and Ke (2011) which we described in section 4.2.2, on the other hand computation times can be considerably reduced by incorporation the initial value algorithm by Gan, Chun, and Johnstone (2015).

5 R Package: pinbasic

In the **pinbasic** package we have implemented utilities for fast and stable estimation of the probability of informed trading. Since the EKOP model is nested in the EHO model, functionalities can also be be applied to this simpler model structure, if needed. State-of-the-art factorization of the model likelihood function as well as hierarchical agglomerative clustering algorithm for generating initial values for optimizations are provided. In total, two different likelihood factorizations and three methods for generating sets of starting values are implemented according to chapter 4.

The probability of informed trading can be estimated for arbitrary length of daily buys and sells data with *pin_est* in code chunk 5.1. Since *pin_est* and its underlying functions share many arguments, we will not discuss same arguments separately for each function but highlight arguments which are unique for different functions.

Code Chunk 5.1 (Source code of function *pin_est* in the **pinbasic** package):

```
function(numbuys = NULL, numsells = NULL, nlminb_control = list(), confint = FALSE,
    ci_control = list(), posterior = TRUE) {
    if (is.null(numbuys))
        stop("Missing data for 'numbuys'")
    if (is.null(numsells))
        stop("Missing data for 'numsells'")
    if (length(numbuys) != length(numsells))
        stop("Unequal lengths for 'numbuys' and 'numsells'")
    init_vals <- initial_vals(numbuys = numbuys, numsells = numsells, method = "HAC")
    res <- pin_est_core(numbuys = numbuys, numsells = numsells, factorization = "Lin_Ke",
        init_vals = init_vals, nlminb_control = nlminb_control, confint = confint,
        ci_control = ci_control, posterior = posterior)
    res
}
```

The arguments `numbuys` and `numsells` take vectors of daily buys and sells. Fine-tuning of the optimizer, *nlminb*, is possible via the `nlminb_control` argument.[58] With the logical argument `confint` one can specify if confidence intervals for the probability of informed trading should be

[58] For details see the help page of the *nlminb* function in the **stats** package.

computed according to section 4.4. The `ci_control` list specifies further details of calculations of confidence intervals which we will explain later in the context of code chunk 5.15. The `posterior` flag switches the calculation of posterior probabilities of the conditions of tradings either on or off.

A call to the *pin_est* function involves evaluations of the *initial_vals* and *pin_est_core* functions. The corresponding source code of these functions is shown in code chunks 5.2 and 5.6, respectively. If *pin_est* is called initial values for optimization are generated by the HAC algorithm which has proven to be the best choice for this task (see section 4.3.4). Furthermore, Lin-Ke likelihood factorization is utilized since it is the most stable known in PIN literature. The **pinbasic** package harnesses the built-in *nlminb* function from the **stats** package for all optimization purposes.

Code Chunk 5.2 (Source code of function *initial_vals* in the **pinbasic** package):

```
function(numbuys = NULL, numsells = NULL, method = "HAC", length = 5, num_clust = 5,
    details = FALSE) {
    if (is.null(numbuys))
        stop("Missing data for 'numbuys'")
    if (is.null(numsells))
        stop("Missing data for 'numsells'")
    if (length(numbuys) != length(numsells))
        stop("Unequal lengths for 'numbuys' and 'numsells'")
    meth <- match.arg(method, choices = c("HAC", "HAC_Ref", "Grid"))
    res <- switch(meth, Grid = {
        init_grid_search(numbuys = numbuys, numsells = numsells, length = length,
            details = details)
    }, HAC = {
        init_hac(numbuys = numbuys, numsells = numsells)
    }, HAC_Ref = {
        init_hac_ref(numbuys = numbuys, numsells = numsells, j = num_clust)
    })
    res
}
```

The **pinbasic** package offers the *initial_vals* function for generating set(s) of starting values which can be used in optimization routines estimating the probability of informed trading. The algorithm for calculating initial values can be specified via the `method` argument by which the user can choose one of the three discussed methods from section 4.3. Brute force grid search algorithm can be chosen via `'Grid'`, for HAC or refined HAC algorithm `method` needs to equal `"HAC"` or `"HAC_Ref"`, respectively. According to the specified string for the `method` argument, one of the specialized functions *init_grid_search*, *init_hac* or *init_hac_ref* is internally called. The source codes for each of the functions are presented in code chunks 5.3, 5.4 and 5.5. By default, *initial_vals* uses the HAC algorithm and therefore calls *init_hac*.

Code Chunk 5.3 (Source code of function *init_grid_search* in the **pinbasic** package):

```
function(numbuys = NULL, numsells = NULL, length = 5, details = FALSE) {
    max_obs <- max(c(numbuys, numsells))
    avg_buys <- mean(numbuys)
    avg_sells <- mean(numsells)
    alpha <- seq(from = 0.1, to = 0.9, length.out = length)
    delta <- alpha
    gamma <- alpha
    epsilon_b <- gamma * avg_buys
    mat <- expand.grid(alpha, delta, epsilon_b)
    colnames(mat) <- c("alpha", "delta", "epsilon_b")
    mu <- (avg_buys - mat[, "epsilon_b"])/(mat[, "alpha"] * (1 - mat[, "delta"]))
    epsilon_s <- avg_sells - mat[, "alpha"] * mat[, "delta"] * mu
    mat <- cbind(mat, epsilon_s, mu)
    neg_eps <- which(mat[, "epsilon_s"] < 0)
    irrelevant_mu <- which(mat[, "mu"] > max_obs)
    rem_lines <- unique(c(neg_eps, irrelevant_mu))
    if (length(rem_lines) > 0) {
        mat <- mat[-rem_lines, ]
    }
    res <- data.matrix(mat)
    if (details) {
        res_list <- vector("list", 4)
        names(res_list) <- c("inits", "neg_eps", "irr_mu", "rem")
        res_list[["inits"]] <- res
        res_list[["neg_eps"]] <- length(neg_eps)
        res_list[["irr_mu"]] <- length(irrelevant_mu)
        res_list[["rem"]] <- length(rem_lines)
        return(res_list)
    } else return(res)
}
```

Code Chunk 5.4 (Source code of function *init_hac* in the **pinbasic** package):

```
function(numbuys = NULL, numsells = NULL) {
    N <- length(numbuys)
    order_imb <- numbuys - numsells
    clust <- hclust(dist(order_imb), method = "complete")
    clust3 <- cutree(clust, k = 3)
    cluster_means <- numeric(3)
    for (k in 1:3) {
        cluster_means[k] <- mean(order_imb[clust3 == k])
    }
    max_ind <- which.max(cluster_means)
    min_ind <- which.min(cluster_means)
```

```
    no_ind <- (1:3)[(!((1:3) %in% c(max_ind, min_ind)))]
    cluster_bad_ind <- which(clust3 == min_ind)
    cluster_good_ind <- which(clust3 == max_ind)
    cluster_no_ind <- which(clust3 == no_ind)
    cluster_bad <- cbind(numbuys[cluster_bad_ind], numsells[cluster_bad_ind])
    cluster_good <- cbind(numbuys[cluster_good_ind], numsells[cluster_good_ind])
    cluster_no <- cbind(numbuys[cluster_no_ind], numsells[cluster_no_ind])
    weights <- c(nrow(cluster_bad)/N, nrow(cluster_good)/N, nrow(cluster_no)/N)
    clusters <- vector("list", 3)
    clusters[[1]] <- cluster_bad
    clusters[[2]] <- cluster_good
    clusters[[3]] <- cluster_no
    mean_daily_buys <- sapply(clusters, function(x) mean(x[, 1]))
    mean_daily_sells <- sapply(clusters, function(x) mean(x[, 2]))
    mat <- cbind(mean_daily_buys, mean_daily_sells, weights)
    colnames(mat) <- c("mean_daily_buys", "mean_daily_sells", "weight")
    rownames(mat) <- c("BadNews", "GoodNews", "NoNews")
    mat[which(is.nan(mat))] <- 0
    if (mat["BadNews", "weight"] == 0)
        mu_s <- 0
    if (mat["GoodNews", "weight"] == 0)
        mu_b <- 0
    alpha <- mat["GoodNews", "weight"] + mat["BadNews", "weight"]
    delta <- mat["BadNews", "weight"]/alpha
    eps_b_helper <- mat["BadNews", "weight"] + mat["NoNews", "weight"]
    eps_s_helper <- mat["GoodNews", "weight"] + mat["NoNews", "weight"]
    eps_b <- (mat["BadNews", "weight"]/eps_b_helper) * mat["BadNews", "mean_daily_buys"] +
        (mat["NoNews", "weight"]/eps_b_helper) * mat["NoNews", "mean_daily_buys"]
    eps_s <- (mat["GoodNews", "weight"]/eps_s_helper) * mat["GoodNews", "mean_daily_sells"] +
        (mat["NoNews", "weight"]/eps_s_helper) * mat["NoNews", "mean_daily_sells"]
    if (mat["GoodNews", "weight"] > 0)
        mu_b <- mat["GoodNews", "mean_daily_buys"] - eps_b
    if (mat["BadNews", "weight"] > 0)
        mu_s <- mat["BadNews", "mean_daily_sells"] - eps_s
    if (mu_b < 0)
        mu_b <- 0
    if (mu_s < 0)
        mu_s <- 0
    mu <- (mat["GoodNews", "weight"]/alpha) * mu_b + (mat["BadNews", "weight"]/alpha) *
        mu_s
    res <- matrix(data = c(alpha, delta, eps_b, eps_s, mu), nrow = 1, ncol = 5)
    colnames(res) <- c("alpha", "delta", "epsilon_b", "epsilon_s", "mu")
    res
}
```

Code Chunk 5.5 (Source code of function *init_hac_ref* in the **pinbasic** package):

```
function(numbuys = NULL, numsells = NULL, j = 5) {
    N <- length(numbuys)
    ordered_clusters <- vector("list", j + 1)
    order_imb <- numbuys - numsells
    abs_order_imb <- abs(order_imb)
    res <- matrix(data = NA, nrow = j, ncol = 5)
    colnames(res) <- c("alpha", "delta", "epsilon_b", "epsilon_s", "mu")
    clust <- hclust(dist(abs_order_imb), method = "complete")
    clustj <- cutree(clust, k = j + 1)
    cluster_means <- numeric(j + 1)
    for (k in 1:(j + 1)) {
        cluster_means[k] <- mean(abs_order_imb[clustj == k])
    }
    cluster_means_ord_ind <- order(cluster_means, decreasing = FALSE)
    for (k in 1:j) {
        no_event_ind <- clustj %in% cluster_means_ord_ind[1:k]
        event_ind <- clustj %in% cluster_means_ord_ind[(k + 1):(j + 1)]
        alpha <- sum(event_ind)/N
        mu <- mean(abs_order_imb[event_ind]) - mean(abs_order_imb[no_event_ind])
        good_ind <- bad_ind <- numeric()
        for (l in cluster_means_ord_ind[(k + 1):(j + 1)]) {
            mean_event_cluster <- mean(order_imb[clustj == l])
            if (mean_event_cluster > 0)
                good_ind <- c(good_ind, l) else bad_ind <- c(bad_ind, l)
        }
        good_news_ind <- clustj %in% good_ind
        bad_news_ind <- clustj %in% bad_ind
        cluster_no <- cbind(numbuys[no_event_ind], numsells[no_event_ind])
        cluster_good <- cbind(numbuys[good_news_ind], numsells[good_news_ind])
        cluster_bad <- cbind(numbuys[bad_news_ind], numsells[bad_news_ind])
        weights <- c(nrow(cluster_bad)/N, nrow(cluster_good)/N, nrow(cluster_no)/N)
        clusters <- vector("list", 3)
        clusters[[1]] <- cluster_bad
        clusters[[2]] <- cluster_good
        clusters[[3]] <- cluster_no
        mean_daily_buys <- sapply(clusters, function(x) mean(x[, 1]))
        mean_daily_sells <- sapply(clusters, function(x) mean(x[, 2]))
        mat <- cbind(mean_daily_buys, mean_daily_sells, weights)
        colnames(mat) <- c("mean_daily_buys", "mean_daily_sells", "weight")
        rownames(mat) <- c("BadNews", "GoodNews", "NoNews")
        mat[which(is.nan(mat))] <- 0
        if (mat["BadNews", "weight"] == 0) {
            mu_s <- 0
        }
        if (mat["GoodNews", "weight"] == 0) {
            mu_b <- 0
```

```
        }
        delta <- mat["BadNews", "weight"]/alpha
        eps_b_helper <- mat["BadNews", "weight"] + mat["NoNews", "weight"]
        eps_s_helper <- mat["GoodNews", "weight"] + mat["NoNews", "weight"]
        eps_b <- (mat["BadNews", "weight"]/eps_b_helper) * mat["BadNews", "mean_daily_buys"] +
            (mat["NoNews", "weight"]/eps_b_helper) * mat["NoNews", "mean_daily_buys"]
        eps_s <- (mat["GoodNews", "weight"]/eps_s_helper) * mat["GoodNews",
            "mean_daily_sells"] + (mat["NoNews", "weight"]/eps_s_helper) * mat["NoNews",
            "mean_daily_sells"]
        if (mat["GoodNews", "weight"] > 0)
            mu_b <- mat["GoodNews", "mean_daily_buys"] - eps_b
        if (mat["BadNews", "weight"] > 0)
            mu_s <- mat["BadNews", "mean_daily_sells"] - eps_s
        res[k, 1:5] <- c(alpha, delta, eps_b, eps_s, mu)
    }
    return(res)
}
```

In addition, there are method-specific arguments `length`, `num_clust` and `details`. The `length` argument is relevant for grid search in which `length` determines the grid width of the interval $[0.1, 0.9]$. This influences the amount of possible initial values for the probability parameters α and δ as well as γ. If `details` is set to `TRUE` and `method = 'Grid'` a list is returned with elements representing a matrix with sets of starting values, the number of sets removed due to negative values for the intensity of uninformed sells and guesses for the intensity of informed trading that are larger than the highest values of aggregated buy or sell orders in the data. Otherwise, solely a matrix of initial values is returned. Function argument `num_clust` determines the number of clusters trading data is grouped into if `method = "HAC_Ref"`.

The *pin_est* function is a user-friendly wrapper around the real workhorse function, *pin_est_core*, presented in code chunk 5.6.

Code Chunk 5.6 (Source code of function *pin_est_core* in the **pinbasic** package):

```
function(numbuys = NULL, numsells = NULL, factorization = "Lin_Ke", init_vals = NULL,
    lower = rep(0, 5), upper = c(1, 1, rep(Inf, 3)), num_best_res = 1, only_converged = TRUE,
    nlminb_control = list(), confint = FALSE, ci_control = list(), posterior = TRUE) {
    if (is.null(init_vals))
        stop("No initial values provided!")
    if (is.null(lower) || is.null(upper))
        stop("Lower or upper bounds missing!")
    if (length(numbuys) != length(numsells))
        stop("Unequal lengths for 'numbuys' and 'numsells'")
    factr <- match.arg(factorization, choices = c("Lin_Ke", "EHO"))
    mat <- matrix(data = NA, nrow = nrow(init_vals), ncol = ncol(init_vals) +
        4)
    colnames(mat) <- c(colnames(init_vals), "loglike", "PIN", "Convergence",
```

```
    "Iterations")
opt_message <- numeric(nrow(init_vals))
fn <- function(x) pin_ll(param = x, numbuys = numbuys, numsells = numsells,
    factorization = factr)
par_names <- c("alpha", "delta", "epsilon_b", "epsilon_s", "mu")
if (nrow(mat) == 1)
    num_best_res <- 1
ci_con <- list(n = 10000, seed = NULL, level = 0.95, ncores = 1)
names_ci <- names(ci_con)
ci_con[(nam_ci <- names(ci_control))] <- ci_control
if (length(noNms <- nam_ci[!nam_ci %in% names_ci]))
    warning("unknown names in control: ", paste(noNms, collapse = ", "))
if (ci_con$ncores < 1)
    stop("Set valid number of cpu cores for 'ncores'")
nlminb_con <- list(eval.max = 1000, iter.max = 500, trace = 0, abs.tol = 0,
    rel.tol = 1e-10, x.tol = 1.5e-08, xf.tol = 2.2e-14, step.min = 1, step.max = 1,
    sing.tol = 1e-10)
names_nlminb <- names(nlminb_con)
nlminb_con[(nam_nlminb <- names(nlminb_control))] <- nlminb_control
if (length(noNms_nlminb <- nam_nlminb[!nam_nlminb %in% names_nlminb]))
    warning("unknown names in control: ", paste(noNms_nlminb, collapse = ", "))
for (i in 1:nrow(mat)) {
    tmp <- nlminb(start = init_vals[i, ], objective = function(x) -fn(x),
        lower = lower, upper = upper, control = nlminb_con)
    mat[i, par_names] <- tmp$par
    mat[i, "loglike"] <- -tmp$objective
    mat[i, "PIN"] <- pin_calc(param = tmp$par)
    mat[i, "Convergence"] <- as.integer(tmp$convergence)
    mat[i, "Iterations"] <- as.integer(tmp$iterations)
    opt_message[i] <- tmp$message
}
if (nrow(mat) > 1) {
    mat <- mat[order(mat[, "loglike"], decreasing = TRUE), ]
    start_vals <- init_vals[order(mat[, "loglike"], decreasing = TRUE),
        ]
    if (only_converged) {
        converged <- as.logical(!mat[, "Convergence"])
        mat <- mat[converged, ]
        start_vals <- start_vals[converged, ]
    }
} else start_vals <- matrix(init_vals[1, ], nrow = 1)
if (num_best_res == "all")
    num_best_res <- nrow(mat)
if (num_best_res > 1) {
    mat_list <- vector("list", num_best_res)
    names(mat_list) <- paste0("Best", 1:num_best_res)
    for (i in 1:num_best_res) {
        mat_list[[paste0("Best", i)]] <- vector("list", 7)
```

```
            names(mat_list[[paste0("Best", i)]]) <- c("Results", "ll", "pin",
                "conv", "message", "iterations", "init_vals")
            if (confint)
                mat_list[[paste0("Best", i)]][["confint"]] <- numeric(2)
            tmp_res <- summary_car(param = mat[i, par_names], numbuys = numbuys,
                numsells = numsells, factorization = factorization, lower = lower,
                upper = upper)
            if (!is.null(tmp_res)) {
                mat_list[[paste0("Best", i)]][["Results"]] <- tmp_res
            } else {
                mat_list[[paste0("Best", i)]][["Results"]] <- matrix(data = NA,
                  ncol = 4, nrow = 5)
                colnames(mat_list[[paste0("Best", i)]][["Results"]]) <- c("Estimate",
                  "Std. error", "t value", "Pr(> t)")
                rownames(mat_list[[paste0("Best", i)]][["Results"]]) <- c("alpha",
                  "delta", "epsilon_b", "epsilon_s", "mu")
                mat_list[[paste0("Best", i)]][["Results"]][, "Estimate"] <- mat[i,
                  par_names]
            }
            mat_list[[paste0("Best", i)]][["ll"]] <- mat[i, "loglike"]
            mat_list[[paste0("Best", i)]][["pin"]] <- mat[i, "PIN"]
            mat_list[[paste0("Best", i)]][["conv"]] <- mat[i, "Convergence"]
            mat_list[[paste0("Best", i)]][["message"]] <- opt_message[i]
            mat_list[[paste0("Best", i)]][["iterations"]] <- mat[i, "Iterations"]
            mat_list[[paste0("Best", i)]][["init_vals"]] <- start_vals[i, ]
            if (confint) {
                mat_list[[paste0("Best", i)]][["confint"]] <- pin_confint(param = mat[i,
                  par_names], numbuys = numbuys, numsells = numsells, lower = lower,
                  upper = upper, n = ci_con$n, seed = ci_con$seed, level = ci_con$level,
                  ncores = ci_con$ncores)
            }
            if (posterior) {
                mat_list[[paste0("Best", i)]][["posterior"]] <- posterior(param = mat[i,
                  par_names], numbuys = numbuys, numsells = numsells)
            }
        }
    } else {
        mat_list <- vector("list", 7)
        names(mat_list) <- c("Results", "ll", "pin", "conv", "message", "iterations",
            "init_vals")
        if (confint)
            mat_list[["confint"]] <- numeric(2)
        tmp_res <- summary_car(param = mat[1, par_names], numbuys = numbuys,
            numsells = numsells, factorization = factorization, lower = lower,
            upper = upper)
        if (!is.null(tmp_res)) {
            mat_list[["Results"]] <- tmp_res
        } else {
```

```
            mat_list[["Results"]] <- matrix(data = NA, ncol = 4, nrow = 5)
            colnames(mat_list[["Results"]]) <- c("Estimate", "Std. error", "t value",
                "Pr(> t)")
            rownames(mat_list[["Results"]]) <- c("alpha", "delta", "epsilon_b",
                "epsilon_s", "mu")
            mat_list[["Results"]][, "Estimate"] <- mat[i, par_names]
        }
        mat_list[["ll"]] <- mat[1, "loglike"]
        mat_list[["pin"]] <- mat[1, "PIN"]
        mat_list[["conv"]] <- mat[1, "Convergence"]
        mat_list[["message"]] <- opt_message[1]
        mat_list[["iterations"]] <- mat[1, "Iterations"]
        mat_list[["init_vals"]] <- start_vals[1, ]
        names(mat_list[["init_vals"]]) <- c("alpha", "delta", "epsilon_b", "epsilon_s",
            "mu")
        if (confint) {
            mat_list[["confint"]] <- pin_confint(param = mat[1, par_names],
                numbuys = numbuys, numsells = numsells, lower = lower, upper = upper,
                n = ci_con$n, seed = ci_con$seed, level = ci_con$level, ncores = ci_con$ncores)
        }
        if (posterior) {
            mat_list[["posterior"]] <- posterior(param = mat[1, par_names],
                numbuys = numbuys, numsells = numsells)
        }
    }
    mat_list
}
```

The *pin_est_core* function is much more flexible than *pin_est*. The user is able to specify additional arguments and change the settings of optimization runs. The `init_vals` argument takes matrices of initial values which were produced by *initial_vals* function. It is also possible to provide a user-defined matrix of starting values in which each column consists of one or more choices for each model parameter. The ordering of the columns needs to be α, δ, ϵ_b, ϵ_s, μ or the matrix needs column names which equal the internally used names for the model parameters, `"alpha"`, `"delta"`, `"epsilon_b"`, `"epsilon_s"` and `"mu"`. Factorization of the likelihood function can be specified by the `factorization` argument which offers two choices, `"Lin_Ke"` and `"EHO"`. This argument is passed to the *pin_ll* function which then calls either *linke* or *eho*. By default, Lin-Ke factorization is utilized. The two latter functions are both implemented in the C++ language with the help of the **Rcpp** package to increase computing speed and therefore reduce execution time.[59] The corresponding R and C++ source codes can be found in the code chunks 5.7, 5.8 and 5.9. Whether the given parameter vector `param` is valid is checked with the *param_check* function in code chunk 5.10.

Lower and upper bounds for maximizations can be set via the `lower` and `upper` arguments of the *pin_est_core* function. The standard is that the smallest value allowed for all model parameters

[59] An introduction to the **Rcpp** package and its syntax is given in Eddelbuettel and François (2011).

is 0. Probability parameters must not be larger than 1, whereas the upper bound for the intensities is set to infinity. The amount of output is controlled by the `num_best_res` and `only_converged` arguments. Only the results for the best optimization run are returned if `num_best_res = 1`. If one is interested in the results for every maximization, whether it has converged or not, `num_best_res` should be specified as the string `"all"` while `only_converged` is set to `FALSE`, so that no results of runs are dismissed.

Code Chunk 5.7 (Source code of function *pin_ll* in the **pinbasic** package):

```
function(param = NULL, numbuys = NULL, numsells = NULL, factorization = "Lin_Ke") {
    if (is.null(numbuys))
        stop("No number of daily buys given!")
    if (is.null(numsells))
        stop("No number of daily sells given!")
    if (length(numbuys) != length(numsells))
        stop("Buys and Sells length differ!")
    param <- param_check(param)
    factr <- match.arg(factorization, choices = c("Lin_Ke", "EHO"))
    switch(factr, Lin_Ke = {
        linke(param = param, numbuys = numbuys, numsells = numsells)
    }, EHO = {
        eho(param = param, numbuys = numbuys, numsells = numsells)
    })
}
```

Code Chunk 5.8 (Source code of C++ function *linke* in the **pinbasic** package):

```
double linke(NumericVector param, NumericVector numbuys, NumericVector numsells) {
  double n_par  = -numbuys.length() * (param[2] + param[3]);

  double rat1 = param[4]/param[3];
  double rat2 = param[4]/param[2];

  double rat1log1p = log1p(rat1);
  double rat2log1p = log1p(rat2);

  double const1 = log(param[4] + param[3]);
  double const2 = log(param[4] + param[2]);

  double prob_no = 1.0 - param[0];
  double prob_good = param[0] * (1.0 - param[1]);
  double prob_bad = param[0] * param[1];
```

```
  NumericVector e1 = -param[4] - numsells * rat1log1p;
  NumericVector e2 = -param[4] - numbuys * rat2log1p;
  NumericVector e3 = e2 + e1 + 2.0 * param[4];

  NumericVector e_max0 = pmax(e1, e2);
  NumericVector e_max = pmax(e_max0, e3);

  double part1 = n_par + sum(numbuys) * const2 +
                 sum(numsells) * const1 + sum(e_max);

  double part2 = sum(log(prob_no * exp(e3-e_max) + prob_good * exp(e1 - e_max) +
                         prob_bad * exp(e2 - e_max)));


  double ll = part1 + part2;

  return(ll);
}
```

Code Chunk 5.9 (Source code of C++ function *eho* in the **pinbasic** package):

```
double eho(NumericVector param , NumericVector numbuys, NumericVector numsells) {
  double n_par = -numbuys.length() * (param[3] + param[2]);

  NumericVector m = pmin(numbuys, numsells) + 0.5 * pmax(numbuys, numsells);

  double helper_sum1 = param[4] + param[3];
  double helper_sum2 = param[4] + param[2];

  double xs = param[3]/helper_sum1;
  double xb = param[2]/helper_sum2;
  double log_xs = log(xs);
  double log_xb = log(xb);

  double exp_mu = exp(-param[4]);
  NumericVector xs_helper = exp((numsells - m) * log_xs);
  NumericVector xb_helper = exp((numbuys - m) * log_xb);

  double prob_no = 1.0 - param[0];
  double prob_good = param[0] * (1.0 - param[1]);
  double prob_bad = param[0] * param[1];
```

```
  double part1 = n_par +  (log_xb + log_xs) * sum(m) +
                 log(param[4] + param[2]) * sum(numbuys) +
                 log(param[4] + param[3]) * sum(numsells);

  double part2 = sum(log(prob_no * xs_helper * xb_helper +
                         prob_good * exp_mu * xs_helper * exp(-m * log_xb) +
                         prob_bad * exp_mu * xb_helper * exp(-m * log_xs)));

  double ll = part1 + part2;
  return(ll);
}
```

Code Chunk 5.10 (Source code of function *param_check* in the **pinbasic** package):

```
function(param = NULL) {
    if (is.null(param))
        stop("No parameter vector provided!")
    if (!(length(param) == 5))
        stop("Parameter vector need to have 5 elements)!")
    if (is.null(names(param))) {
        names(param) <- c("alpha", "delta", "epsilon_b", "epsilon_s", "mu")
    }
    if (!is.null(names(param)) && !all(names(param) %in% c("alpha", "delta",
        "epsilon_b", "epsilon_s", "mu"))) {
        names(param) <- c("alpha", "delta", "epsilon_b", "epsilon_s", "mu")
    }
    param
}
```

A list with seven to nine elements is returned by *pin_est* and *pin_est_core*, if the `num_best_res` argument equals 1, depending on the `confint` and `posterior` flags. If `num_best_res` is larger than 1 a list of list is returned where the first layer represents optimization runs. For each of them a list with seven to nine elements is returned. The names of the list elements of the first layer begin with the string `Best` and end with an integer in the range from 1 to the value of `num_best_res`. Those list elements are returned in descending order according to the final values of the likelihood function.

The `Results` slot provides a summary including parameter estimates, standard errors, t-values and corresponding p-values for significance tests which is computed with the help of the auxiliary function *summary_car* with the corresponding source code in code chunk 5.11. The `bound_hit` function (see code chunk 5.12) checks for estimates which either hit lower or upper bounds or are so close to them that they are ignored for summary statistics to circumvent numerical instabilities in the computation of the Hessian matrix.

Code Chunk 5.11 (Source code of function *summary_car* in the **pinbasic** package):

```
function(param = NULL, numbuys = NULL, numsells = NULL, factorization = NULL,
    lower = NULL, upper = NULL, eigentol = 1e-12) {
    boundary <- bound_hit(param, lower, upper)
    vcov <- vcov_car(param = param, numbuys = numbuys, numsells = numsells,
        factorization = factorization, lower = lower, upper = upper, eigentol = eigentol)
    if (is.null(vcov))
        return(NULL)
    if (any(diag(vcov) < 0) | any(is.infinite(vcov))) {
        warning("Infeasible Variance-Covariance Matrix with negative variances or infinite entries!")
        return(NULL)
    }
    std_err <- t_vals <- p_vals <- numeric(length(param))
    std_err[!boundary] <- sqrt(diag(vcov))
    std_err[boundary] <- NA
    t_vals[!boundary] <- param[!boundary]/std_err[!boundary]
    p_vals[!boundary] <- 2 * stats::pnorm(-abs(t_vals[!boundary]))
    t_vals[boundary] <- NA
    p_vals[boundary] <- NA
    results <- cbind(Estimate = param, `Std. error` = std_err, `t value` = t_vals,
        `Pr(> t)` = p_vals)
    results
}
```

Code Chunk 5.12 (Source code of function *bound_hit* in the **pinbasic** package):

```
function(param = NULL, lower = NULL, upper = NULL) {
    low.hit <- abs(lower - param) < 1e-10
    upper.hit <- abs(upper - param) < 1e-10
    bound.hit <- (low.hit | upper.hit)
    bound.hit
}
```

Standard deviations of parameter estimates are calculated as diagonal elements of the corresponding covariance matrix returned by the *vcov_car* function in code chunk 5.13. The Hessian matrix, which is essential to receive the covariance matrix, is returned by the built-in *optimHess* function from the **stats** package. The *vcov_car* function introduces the `eigentol` argument which sets the eigenvalue tolerance for the Hessian matrix. It controls when the Hessian is treated as numerically singular.

Code Chunk 5.13 (Source code of function *vcov_car* in the **pinbasic** package):

```
function(param = NULL, numbuys = NULL, numsells = NULL, factorization = NULL,
    lower = NULL, upper = NULL, eigentol = 1e-12) {
    fun <- function(par) {
        pin_ll(par, numbuys = numbuys, numsells = numsells, factorization = factorization)
    }
    bound.hit <- bound_hit(param, lower, upper)
    names.param <- names(param)
    param.bound <- param[bound.hit]
    join_param <- function(x) c(param.bound, x)[names.param]
    hess <- tryCatch(stats::optimHess(fn = function(x) fun(join_param(x)), par = param[!bound.hit]),
        error = function(e) e)
    if (is.matrix(hess)) {
        if (any(is.nan(hess)) | any(is.na(hess)) | any(is.infinite(hess))) {
            warning("NaN, NA or infinite values in Hesse matrix")
            return(NULL)
        }
    } else return(NULL)
    hess_eigen <- abs(eigen(hess, symmetric = TRUE, only.values = TRUE)$values)
    vcov_mat <- matrix(0, length(param[!bound.hit]), length(param[!bound.hit]))
    rownames(vcov_mat) <- colnames(vcov_mat) <- names(param[!bound.hit])
    if (min(hess_eigen) > (eigentol * max(hess_eigen))) {
        vcov_mat <- solve(-hess)
        vcov_mat <- (vcov_mat + t(vcov_mat))/2
    } else {
        vcov_mat <- NULL
        warning("Singular Hesse matrix")
    }
    vcov_mat
}
```

The list returned by *pin_est_core* and *pin_est* exhibit the likelihood function value and estimated probability of informed trading in the `ll` and `pin` slots, respectively. The calculation of PIN is implemented with the *pin_calc* function in code chunk 5.14.

Code Chunk 5.14 (Source code of function *pin_calc* in the **pinbasic** package):

```
function(param = NULL) {
    param <- param_check(param)
    res <- (param["alpha"] * param["mu"])/(param["alpha"] * param["mu"] + param["epsilon_b"] +
        param["epsilon_s"])
    names(res) <- NULL
    res
}
```

Some information about *nlminb*-specific results can be gathered from `conv`, `message` and `iterations` list elements which deliver the convergence code, additional convergence messages and the number of necessary iterations of the optimizer. The help page of the *nlminb* function is a good starting point for more information about the meaning of the additional messages of the optimizer. However, if one wants to delve deep into the details, then the documentation by Gay (1990) is the recommended lecture. The initial values used for optimization are stored in `init_vals` slot.

If the `confint` flag is set to `TRUE` for calls of either *pin_est* or *pin_est_core*, results are stored in the `confint` slot of the returned list. Confidence intervals for PIN can be computed with the *pin_confint* function. Its source code is shown in code chunk 5.15.

Code Chunk 5.15 (Source code of function *pin_confint* in the **pinbasic** package):

```
function(param = NULL, numbuys = NULL, numsells = NULL, method = "HAC", lower = rep(0,
    5), upper = c(1, 1, rep(Inf, 3)), n = 10000, seed = NULL, level = 0.95,
    ncores = 1) {
    param <- param_check(param)
    if (!is.numeric(ncores) && ncores < 1)
        stop("No valid 'ncores' argument!")
    if (length(numbuys) != length(numsells))
        stop("Unequal lengths for 'numbuys' and 'numsells'")
    meth <- match.arg(method, choices = c("HAC", "HAC_Ref", "Grid"))
    set.seed(seed)
    sim_pin <- numeric(n)
    ndays <- length(numbuys)
    fn <- function(par, buys, sells) {
        pin_ll(param = par, numbuys = buys, numsells = sells, factorization = "Lin_Ke")
    }
    if (is.null(param)) {
        init_vals <- initial_vals(numbuys = numbuys, numsells = numsells, method = "HAC")
        param_dat <- nlminb(start = init_vals[1, ], objective = function(x) -fn(x,
            numbuys, numsells), lower = lower, upper = upper)$par
    } else param_dat <- param
    sim_dat <- replicate(n = n, simulateBS(param = param_dat, ndays = ndays),
        simplify = FALSE)
    initial_mat <- lapply(sim_dat, function(x) {
        initial_vals(numbuys = x[, "Buys"], numsells = x[, "Sells"], method = meth)
    })
    if (ncores == 1) {
        par_est <- Map(function(x, y) nlminb(start = y[1, ], objective = function(par) -fn(par,
            x[, "Buys"], x[, "Sells"]), lower = lower, upper = upper)$par, x = sim_dat,
            y = initial_mat)
    } else {
        cl <- makeCluster(getOption("cl.cores", ncores))
        split_ind <- split(seq_len(n), seq_len(ncores))
        cl_export(cl, sim_dat, initial_mat, split_ind)
```

```
        par_est <- clusterCall(cl, fun = ci_mc_helper, fn = fn, lower = lower,
            upper = upper)
        par_est <- do.call(c, par_est)
        on.exit(stopCluster(cl))
    }
    sim_pin <- sapply(par_est, function(x) pin_calc(x))
    conf <- quantile(sim_pin, probs = c((1 - level)/2, 1 - (1 - level)/2))
    conf
}
```

The number of simulation runs, the seed of the default *Mersenne-Twister* random number generator, the confidence level and the number of utilized CPU cores can be set via arguments `n`, `seed`, `level` and `ncores`, respectively. If confidence intervals should be returned by a call to either *pin_est* or *pin_est_core* the settings can be controlled via the `ci_control` argument of both functions. Names of `ci_control` elements need to match argument names of the *pin_confint* function or are ignored otherwise. The *pin_confint* function harnesses the C++ function *simulateBS* to simulate datasets of daily buys and sells whose source code can be read from code chunk 5.16.

Code Chunk 5.16 (Source code of C++ function *simulateBS* in the **pinbasic** package):

```
NumericMatrix simulateBS(NumericVector param, int ndays) {
  NumericMatrix res(ndays,2);
  IntegerVector states_ind = IntegerVector::create(0,1,2);
  NumericVector state_probs(3);
  state_probs[0] = 1.0 - param[0];
  state_probs[1] = param[0] * (1.0 - param[1]);
  state_probs[2] = param[0] * param[1];

  NumericVector buys(ndays);
  NumericVector sells(ndays);

  IntegerVector states = sample(states_ind, ndays, true, state_probs);

// indices for no-, good- and bad-news days
  LogicalVector ind_no = states == 0;
  LogicalVector ind_good = states == 1;
  LogicalVector ind_bad = states == 2;

  int len_no = sum(ind_no);
  int len_good = sum(ind_good);
  int len_bad = sum(ind_bad);
```

```
// drawing Poisson distributed random numbers for daily buys and sells
// according to the actual buy and sell intensities depending on the
// condition of the trading day
  buys[ind_no] = rpois(len_no, param[2]);
  sells[ind_no] = rpois(len_no, param[3]);

  buys[ind_good] = rpois(len_good, param[2] + param[4]);
  sells[ind_good] = rpois(len_good, param[3]);

  buys[ind_bad] = rpois(len_bad, param[2]);
  sells[ind_bad] = rpois(len_bad, param[3] + param[4]);

  res(_,0) = buys;
  res(_,1) = sells;

  colnames(res) = CharacterVector::create("Buys", "Sells");

  return(res);
}
```

The *cl_export*, *assign_to_global* and *ci_mc_helper* functions are auxiliary functions to enable computations of confidence intervals in parallel. Corresponding source codes are presented in code chunks 5.17, 5.18 and 5.19. While *cl_export* uses *assign_to_global* to export appropriate parts of the complete data to the global environment of each involved CPU core, *ci_mc_helper* ensures that optimizations are performed on each of the workers.

Code Chunk 5.17 (Source code of function *cl_export* in the **pinbasic** package):

```
function(cl = NULL, sim_data = NULL, init_mat = NULL, split_ind = NULL) {
    for (i in seq_along(cl)) {
        clusterCall(cl[i], function(data, init) {
            assign_to_global("data_sub", data)
            assign_to_global("init_sub", init)
            NULL
        }, data = sim_data[split_ind[[i]]], init = init_mat[split_ind[[i]]])
    }
}
```

Code Chunk 5.18 (Source code of function *assign_to_global* in the **pinbasic** package):

```
function(string, object, pos = 1) {
    assign(string, object, envir = as.environment(pos))
}
```

Code Chunk 5.19 (Source code of function *ci_mc_helper* in the **pinbasic** package):

```
function(fn = NULL, lower = NULL, upper = NULL) {
    Map(function(x, y) nlminb(start = y[1, ], objective = function(par) -fn(par,
        x[, "Buys"], x[, "Sells"]), lower = lower, upper = upper)$par, x = get("data_sub",
        envir = .GlobalEnv), y = get("init_sub", envir = .GlobalEnv))
}
```

Posterior probabilities for the conditions of trading days as described in section 3.3 can be computed with the *posterior* function in code chunk 5.20. Similar to confidence intervals, if the `posterior` flag is set to `TRUE` when calling one of *pin_est* or *pin_est_core*, a matrix with probabilities of no-news, good-news and bad-news for each trading day is stored in the `posterior` element of the returned list.

Code Chunk 5.20 (Source code of function *posterior* in the **pinbasic** package):

```
function(param = NULL, numbuys = NULL, numsells = NULL) {
    param <- param_check(param)
    if (is.null(numbuys))
        stop("Missing data for 'numbuys'")
    if (is.null(numsells))
        stop("Missing data for 'numsells'")
    if (length(numbuys) != length(numsells))
        stop("Unequal lengths for 'numbuys' and 'numsells'")
    rat1 <- param["mu"]/param["epsilon_s"]
    rat2 <- param["mu"]/param["epsilon_b"]
    rat1log1p <- log1p(rat1)
    rat2log1p <- log1p(rat2)
    prob_no <- 1 - param["alpha"]
    prob_good <- param["alpha"] * (1 - param["delta"])
    prob_bad <- param["alpha"] * param["delta"]
    e1 <- -param["mu"] + numsells * rat1log1p
    e2 <- -param["mu"] + numbuys * rat2log1p
    e_max <- pmax.int(e1, e2, 0)
    denom_helper <- e_max + log(prob_no * exp(-e_max) + prob_good * exp(e2 -
        e_max) + prob_bad * exp(e1 - e_max))
    no_prob <- log(prob_no) - denom_helper
    good_prob <- log(prob_good) + e2 - denom_helper
    bad_prob <- log(prob_bad) + e1 - denom_helper
    res <- cbind(exp(no_prob), exp(good_prob), exp(bad_prob))
    colnames(res) <- c("no", "good", "bad")
    class(res) <- c("matrix", "posterior")
    res
}
```

We implemented the plotting method *ggplot.posterior* for the results of the *posterior* function in the **pinbasic** package utilizing the **ggplot2** package by Wickham (2009). Objects returned by *posterior* possess the additional class `posterior`. The **ggplot2** package automatically detects the available plotting method for objects with class `posterior`. Therefore calling the *ggplot* function from the **ggplot2** package with the object returned by *posterior* as the only argument is sufficient to generate a plot of daily posterior probabilities.

Code Chunk 5.21 (Source code of function *ggplot.posterior* in the **pinbasic** package):

```
function(x) {
    if (is.null(rownames(x))) {
        df <- data.frame(bs_date = 1:nrow(x), no = x[, "no"], good = x[, "good"],
            bad = x[, "bad"])
    } else {
        check_rows <- check_bs_dates(rownames(x))
        if (!all(is.na(check_rows))) {
            df <- data.frame(bs_date = as.Date(rownames(x)), no = x[, "no"],
                good = x[, "good"], bad = x[, "bad"])
        }
        if (!is.null(rownames(x)) && is.na(check_rows)) {
            df <- data.frame(bs_date = rownames(x), no = x[, "no"], good = x[,
                "good"], bad = x[, "bad"])
        }
    }
    df_melt <- melt(df, id = "bs_date")
    dp_plot <- ggplot(df_melt, aes(x = bs_date, y = value, fill = variable)) +
      geom_bar(stat = "identity", position = "fill", width = 0.5) + ylab("Posterior Probabilities \n of Trad-
ing Days' Conditions") +
        scale_fill_discrete(breaks = c("no", "good", "bad"), labels = c("no-news",
            "good-news", "bad-news"), guide = guide_legend(nrow = 1, keywidth = 0.5,
          keyheight = 0.5, title = NULL, label.position = "right")) + theme(axis.title.x = element_blank(),
        legend.position = "bottom")
    dp_plot
}
```

The previous code chunks show the dependencies of functions in the **pinbasic** package. They also generate insights about what happens internally if *pin_est* or *pin_est_core* is called. In most cases, the user do not want to modify the internal structures of the package and will use the main functions, but we decided to include the complete source code of the **pinbasic** package for the sake of completeness.

In summary, it gets clear that the user is granted more control about the optimization procedure by using *pin_est_core* directly. However, the settings used in *pin_est* will be sufficient for most applications, which uses the fixed combination of Lin-Ke factorization and HAC algorithm.

No information about the time span of the underlying data is needed to perform optimizations with *pin_est* and *pin_est_core*. Since it is common practice in the literature to use a range of

the underlying data of about 60 trading days to estimate the probability of informed trading, we implemented the *qpin* function which delivers quarterly estimates (see code chunk 5.22). The corresponding dates for buys and sells data can be set via the dates argument. The number of available quarters in the data are detected utilizing functions from the **lubridate** package.

Code Chunk 5.22 (Source code of function *qpin* in the **pinbasic** package):

```
function(numbuys = NULL, numsells = NULL, dates = NULL, nlminb_control = list(),
    confint = FALSE, ci_control = list(), posterior = TRUE) {
    if (is.null(numbuys))
        stop("Missing data for 'numbuys'")
    if (is.null(numsells))
        stop("Missing data for 'numsells'")
    if (is.null(dates))
        stop("Missing 'dates'")
    if (length(numbuys) != length(numsells))
        stop("Unequal lengths for 'numbuys' and 'numsells'")
    quarters <- lubridate::quarter(dates, with_year = TRUE)
    quarters_char <- as.character(quarters)
    quarter_num <- length(unique(quarters))
    quarter_list <- vector("list", quarter_num)
    data_years <- unique(lubridate::year(dates))
    data_years_char <- as.character(data_years)
    quarters_per_year <- numeric(length(data_years))
    names(quarters_per_year) <- data_years_char
    quarter_names <- character(0)
    for (i in data_years_char) {
        quarters_per_year[i] <- max(as.numeric(substring(quarters_char[grepl(i,
            quarters_char)], first = 6)))
        quarter_names <- c(quarter_names, paste0(i, ".", 1:quarters_per_year[i]))
    }
    names(quarter_list) <- quarter_names
    BS_data <- cbind(numbuys, numsells)
    for (i in data_years_char) {
        for (j in 1:quarters_per_year[i]) {
            quarter_list[[paste0(i, ".", j)]] <- BS_data[quarters_char == paste0(i,
                ".", j), ]
        }
    }
    res <- lapply(quarter_list, function(x) pin_est(numbuys = x[, 1], numsells = x[,
        2], nlminb_control = nlminb_control, confint = confint, ci_control = ci_control,
        posterior = posterior))
    class(res) <- c("list", "qpin")
    res
}
```

A plotting method for *ggplot* in the **ggplot2** package for the quarterly estimates returned by

qpin is implemented with the *ggplot.qpin* function in code chunk 5.23. Similar to the visualization of posterior probabilities, it is therefore sufficient to call the *ggplot* function which detects the `qpin` class of the object and harnesses the style provided by the **pinbasic** function.

Code Chunk 5.23 (Source code of function *ggplot.qpin* in the **pinbasic** package):

```
function(x) {
    if (!requireNamespace("ggplot2", quietly = TRUE)) {
        stop("ggplot2 is required for this function to work. Please install it.",
            call. = FALSE)
    }
    if (!requireNamespace("reshape2", quietly = TRUE)) {
        stop("reshape2 is required for this function to work. Please install it.",
            call. = FALSE)
    }
    alpha <- delta <- epsilon_b <- epsilon_s <- mu <- pin <- numeric()
    quat <- names(x)
    alpha <- sapply(x, function(x) x$Results["alpha", "Estimate"])
    delta <- sapply(x, function(x) x$Results["delta", "Estimate"])
    epsilon_b <- sapply(x, function(x) x$Results["epsilon_b", "Estimate"])
    epsilon_s <- sapply(x, function(x) x$Results["epsilon_s", "Estimate"])
    mu <- sapply(x, function(x) x$Results["mu", "Estimate"])
    pin <- sapply(x, function(x) x$pin)
    qpin_df <- data.frame(Quarter = quat, alpha = alpha, delta = delta, epsilon_b = epsilon_b,
        epsilon_s = epsilon_s, mu = mu, PIN = pin)
    qpin_df <- melt(qpin_df, id.vars = "Quarter")
    ID <- character(nrow(qpin_df))
    ID[qpin_df[, "variable"] %in% c("alpha", "delta")] <- "Probability Parameters"
    ID[qpin_df[, "variable"] %in% c("epsilon_b", "epsilon_s", "mu")] <- "Intensity Parameters"
    ID[qpin_df[, "variable"] %in% c("PIN")] <- "Prob. of Informed Trading"
    qpin_df <- transform(qpin_df, facet = ID)
    qpin_df$facet_ordered <- factor(qpin_df$facet, levels = c("Probability Parameters",
        "Intensity Parameters", "Prob. of Informed Trading"))
    p <- ggplot(data = qpin_df, aes_string(x = "Quarter", y = "value", group = "variable")) +
        facet_grid(facet_ordered ~ ., scales = "free_y") + geom_line(aes_string(colour = "variable",
       x = "Quarter", y = "value")) + geom_point(shape = 19, size = 1.25, aes_string(colour = "variable")) +
        scale_y_continuous(breaks = pretty_breaks(n = 3)) + theme(legend.position = "right",
        axis.title.y = element_blank(), axis.title.x = element_blank(), legend.title = element_blank())
    p
}
```

Three synthetic datasets are included, namely `BSinfrequent`, `BSfrequent` and `BSheavy` which represent infrequently, frequently and heavily traded equities, respectively.[60]

At the time of writing, version 1.2.2 of **pinbasic** is available on the *Comprehensive R Archive*

[60] The simulated datasets were already discussed in chapter 4.

Network (CRAN)[61] and is listed in the *CRAN Task View: Empirical Finance*[62]. Development version of the package can be found at and installed from `https://github.com/anre005/pinbasic`.

[61] Package sources and binaries for Windows and Mac can be found at `https://cran.r-project.org/web/packages/pinbasic/index.html`.

[62] `https://cran.r-project.org/web/views/Finance.html`

6 Dynamic Models using High-Frequency Data

The dynamic models we will present in this chapter differ from the EHO model in terms of research focus. While the information risk is the key measure of interest in the latter, the former try to capture the variation of the unobservable fraction of market transactions based on private information.

One major point of criticism for static models is that the explanatory power of transaction size is ignored. Hence, transactions with very little volume are treated exactly the same way as trades in which a huge number of shares are involved. No attention is given to additional information which is often available at no cost and is already available in the data. This may lead to PIN not being an appropriate indicator for trading based on private information and being insensitive to market-wide trends as Aktas, de Bodt, Declerck, and van Oppens (2007) argue. In the context of dynamic models for the probability of informed trading, the (signed) volume of transactions is a crucial feature and therefore mitigates this point of criticism.

Posterior probabilities for conditions of trading days are not relevant for the original research question of the authors of the static PIN models (Easley, Kiefer, O'Hara, and Paperman (1996) and Easley, Hvidkjaer, and O'Hara (2002)). However with the expressions in section 3.3 we can shift it in the direction of the models utilizing high-frequency data to estimate the probability of informed trading.

According to static models, the parameters for trading intensities are constant over the whole time range under consideration and informed traders are only active on information events. Therefore the noise trading intensities do not differ on non-information events and trading days with private information. Hence, in the static models the increased market activity on information events induced by private information with either positive or negative direction is completely assigned to informed market participants. However, in the dynamic setting the trading intensity is modeled with an autoregressive approach. Therefore, dynamic approaches are able to distribute the increased trading intensity on information events to both types of traders, informed and uninformed. One can think of a response or feedback of the noise traders to the increased trading activities. Hence, the amount of insider trading in the dynamic models

should be lower than in the static ones.[63]

Our dynamic approach for the probability of informed trading (PIN-HMM) is based on the model introduced by Tay, Ting, Tse, and Warachka (2009) (PIN-ALACD) but extends it in several aspects. The major difference lies in the modelling of conditions of trading days. Instead of an approach similar to logistic regression used by Tay, Ting, Tse, and Warachka (2009), we think of the sequence of trading days' conditions as a Markov chain with latent states. Such a Markov chain is called hidden Markov model (HMM). Therefore dependencies in the series of trading days' states are introduced since the current state depends on its predecessor due to the Markov property. However, we begin with a description of the general setup of dynamic models presented in this work and introduce several essential variables before we delve into the details of each approach.

The relatively restrictive assumption of latent homogeneous Poisson processes with exponentially distributed waiting times for buys and sells in static models is relaxed so that buys and sells are yet latent Poisson processes but with time-varying intensities.[64] Let $\mathscr{F}_{i-1,d}$ denote the information set upon the $(i-1)$-th trade on trading day d which may include prior trade directions, transaction volumes and lagged durations. Specifically, given $\mathscr{F}_{i-1,d}$, upcoming buys and sells are assumed to follow independent latent stochastic point processes whose interarrival times may follow any distribution with positive support and have common starting time t_{i-1}. Since each recorded arrival comes with several characteristics like transaction price or volume etc. , the point processes for buys and sells are so-called *marked point processes* (see Engle and Russell 1998). Furthermore, buys' and sells' processes are assumed to *evolve with after-effects* and to be *conditionally orderly*. Following Snyder and Miller (1991), the first characteristic describes that for any timestamp $t > t_1$ arrivals of both trade directions in the interval $\left[t, t_{\mathscr{I}_d}\right]$, with the total number of transactions on trading day d $\mathscr{I}_d$, do depend on the sequence of arrivals in the range $[t_1, t)$ with $t_1 < t$. With conditionally orderly point processes we ensure that we can determine which process was the first to arrive at any time t of the trading day since for each process the probability of observing two or more arrivals in a sufficiently short time interval conditional on its history is infinitesimal relative to the probability of observing one arrival as described in equation (6.2).[65] According to Engle and Russell (1998), such a stochastic process is completely described by its conditional intensity function (or equivalently conditional hazard function) for timestamp t given the information about the past of the process.

Similarly to Basu and Rigdon (2001), a general form of the (conditional) intensity functions of non-homogeneous Poisson processes is given by

$$\lambda\left(t \mid \mathscr{F}_{i-1,d}\right) = \lim_{\Delta t \longrightarrow 0} \frac{\Pr\left(N(t+\Delta t) - N(t) = 1 \mid \mathscr{F}_{i-1,d}\right)}{\Delta t}. \tag{6.1}$$

[63] See sections 11.1 and 11.2 for a comparison of the fraction of informed buys or sells on corresponding information events in the static EHO model and our dynamic PIN-HMM approach.

[64] A formal definition of non-homogeneous Poisson processes can be found in Ross (1996). It differs from definition 3.1 in chapter 3 by little modifications.

[65] For more details about general characteristics of point processes see Snyder and Miller (1991).

The authors also state that the following relations holds:

$$\lim_{\Delta t \to 0} \frac{\Pr\big(N(t+\Delta t) - N(t) \geq 2 \mid \mathscr{F}_{i-1,d}\big)}{\Delta t} = 0, \tag{6.2}$$

which precludes the occurrence of multiple trades at a single timestamp.

The i-th duration on trading day d is defined as the difference between timestamps of two consecutive transactions,

$$x_{i,d} = t_{i,d} - t_{i-1,d}, \tag{6.3}$$

where $t_{i,d}$ and $t_{i-1,d}$ are the recorded timestamps of the i-th and $(i-1)$-th transactions on trading day d and $i = 1, \ldots, \mathscr{I}_d$.

According to Tay, Ting, Tse, and Warachka (2009) we model trade directions with a two-state ALACD model. Let $y^q_{i,d}$ denote the trade direction of the i-th transaction on trading day d with condition q which takes on values $j = 1$ or -1 representing a buy or sell, respectively. Once an arrival occurs, either from the point process of buys or sells, both waiting times are reset and start anew. We can think of each realized trade direction as outcome of a duel of both trade directions which is the first to arrive.

States of trading days are indicated by q with $q \in Q = \{\mathscr{N}, \mathscr{G}, \mathscr{B}\}$. Since the conditions of trading days are unobservable, we need to introduce variables to capture the different settings for each state. Let $T^q_{1,i,d}$ and $T^q_{-1,i,d}$ denote the random variables for the next potential waiting times up to the next buy or sell on trading day d which resides in state q, respectively. Hence, conditional on $\mathscr{F}_{i-1,d}$, i-th observed duration is the minimum of the corresponding waiting times of buys' and sells' point processes (see Preve and Tse 2013),

$$x_{i,d} = \min\Big(T^q_{1,i,d}, T^q_{-1,i,d}\Big). \tag{6.4}$$

The trade direction of the i-th transaction, $y^q_{i,d}$, is then given by

$$y^q_{i,d} = \underset{j=-1,1}{\arg\min}\Big(T^q_{j,i,d}\Big). \tag{6.5}$$

Latent waiting times of buys and sells are modeled with logarithmic autoregressive conditional duration (LACD) models (e.g., see Bauwens and Giot 2000). However, asymmetry is introduced in the specifications (ALACD) to allow for interactions between the waiting times of buys' and sells' point processes (e.g., see Bauwens and Giot 2003).

The durations of buys and sells on a no-news trading day d are modeled with

$$T^{\mathscr{N}}_{1,i,d} = \exp(\psi^{\mathscr{N}}_{1,i,d})\rho_{1,i,d} \quad \text{and} \tag{6.6}$$

$$T^{\mathscr{N}}_{-1,i,d} = \exp(\psi^{\mathscr{N}}_{-1,i,d})\rho_{-1,i,d}, \tag{6.7}$$

where $\rho_{1,i,d}$ and $\rho_{-1,i,d}$ are iid multiplicative innovations with

$$\mathbb{E}\big(\rho_{1,i,d}\big) = 1 \quad \text{and} \tag{6.8}$$

$$\mathbb{E}\big(\rho_{-1,i,d}\big) = 1, \tag{6.9}$$

such that the conditional expected waiting times for buys and sells are given by

$$\mathbb{E}\Big(T^{\mathcal{N}}_{1,i,d} \,\Big|\, \mathcal{F}_{i-1,d}\Big) = \exp\big(\psi^{\mathcal{N}}_{1,i,d}\big) \quad \text{and} \tag{6.10}$$

$$\mathbb{E}\Big(T^{\mathcal{N}}_{-1,i,d} \,\Big|\, \mathcal{F}_{i-1,d}\Big) = \exp\big(\psi^{\mathcal{N}}_{-1,i,d}\big). \tag{6.11}$$

The idea behind the class of autoregressive models is that the time dependence of durations can be subsumed in their conditional expectations so that standardized durations $\frac{T^{\mathcal{N}}_{1,i,d}}{\exp\left(\psi^{\mathcal{N}}_{1,i,d}\right)}$ and $\frac{T^{\mathcal{N}}_{-1,i,d}}{\exp\left(\psi^{\mathcal{N}}_{-1,i,d}\right)}$ are iid (e. g. see Bauwens and Giot 2000).

ALACD models can be augmented in such a way that additional exogenous variables $X_{m,i}$ are included, dependencies in the processes are introduced by specifying them in an autoregressive manner. Thus, general formulations for baseline $\psi^{\mathcal{N}}_{1,i,d}$ and $\psi^{\mathcal{N}}_{-1,i,d}$ on no-news trading days are given by

$$\psi^{\mathcal{N}}_{1,i,d} = \nu_{1,-1}\,\mathbb{1}_{\{y_{i-1}=-1\}} + \nu_{1,1}\,\mathbb{1}_{\{y_{i-1}=1\}} + \sum_{k=1}^{p} \alpha_{1,k}\,\psi^{\mathcal{N}}_{1,i-k,d} + \sum_{l=1}^{r} \beta_{1,l} \log x_{i-l,d} + \sum_{m=1}^{o} \zeta_{1,m} X_{m,i}, \tag{6.12}$$

$$\psi^{\mathcal{N}}_{-1,i,d} = \nu_{-1,-1}\,\mathbb{1}_{\{y_{i-1}=-1\}} + \nu_{-1,1}\,\mathbb{1}_{\{y_{i-1}=1\}} + \sum_{k=1}^{p} \alpha_{-1,k}\,\psi^{\mathcal{N}}_{-1,i-k,d} + \sum_{l=1}^{r} \beta_{-1,l} \log x_{i-l,d} + \sum_{m=1}^{o} \zeta_{-1,m} X_{m,i}, \tag{6.13}$$

where the indicator function is represented by $\mathbb{1}$.[66] The coefficients $\nu_{1,-1}$, $\nu_{1,1}$ and $\nu_{-1,-1}$, $\nu_{-1,1}$ constitute the intercepts in the equations for buys and sells, respectively. Hence, the waiting times of buys can influence the durations of sells and vice versa. Coefficients $\alpha_{1,k}$ and $\alpha_{-1,k}$ correspond to lag k of the logarithmic conditional expected duration of buys and sells, coefficients $\beta_{1,l}$ and $\beta_{-1,l}$ describe the influence of the logarithm of the observed durations in the buys and sells equations. respectively. Additional exogenous variables enter the equations of waiting times of buys and sells via $\zeta_{1,m}$ and $\zeta_{-1,m}$. Notice that equations (6.12) and (6.13) are also assumed to hold for the activity of noise traders when private information induce additional trading activities by informed market participants.

[66] How the ALACD recursions are initialized is described in section 8.5.

We assume that conditional expected durations of buys on good-news days and sells on bad-news days decrease in comparison with their no-news counterparts due to the occurrence of insiders. The reductions of waiting times are not allowed to vary over time and transactions, they rather represent a decline in the overall level of times between two consecutive buys and sells. Hence, the autoregressive approaches for buys on good-news days and sells on bad-news days differ from their equivalents on no-news days by positive constant parameters τ_1 and τ_{-1}, respectively:

$$\psi^{\mathscr{G}}_{1,i,d} = \psi^{\mathscr{N}}_{1,i,d} - \tau_1, \tag{6.14}$$

$$\psi^{\mathscr{B}}_{-1,i,d} = \psi^{\mathscr{N}}_{-1,i,d} - \tau_{-1}. \tag{6.15}$$

The conditional expected waiting times for buys on good-news days and sells on bad-news days are therefore given by

$$\mathrm{E}\Big(T^{\mathscr{G}}_{1,i,d} \,\Big|\, \mathscr{F}_{i-1,d}\Big) = \frac{\exp\big(\psi^{\mathscr{N}}_{1,i,d}\big)}{\exp(\tau_1)} = \frac{\mathrm{E}\big(T^{\mathscr{N}}_{1,i,d} \,\big|\, \mathscr{F}_{i-1,d}\big)}{\exp(\tau_1)} \quad \text{and} \tag{6.16}$$

$$\mathrm{E}\Big(T^{\mathscr{B}}_{-1,i,d} \,\Big|\, \mathscr{F}_{i-1,d}\Big) = \frac{\exp\big(\psi^{\mathscr{N}}_{-1,i,d}\big)}{\exp(\tau_{-1})} = \frac{\mathrm{E}\big(T^{\mathscr{N}}_{-1,i,d} \,\big|\, \mathscr{F}_{i-1,d}\big)}{\exp(\tau_{-1})}. \tag{6.17}$$

We can see in equations (6.16) and (6.17) that the conditional expected waiting times for buys (sells) on good-news (bad-news) trading days are proportional to their counterparts on no-news days. The baseline waiting times of buys and sells on the corresponding information events are multiplied by the constant factors $\frac{1}{\exp(\tau_1)} \leq 1$ and $\frac{1}{\exp(\tau_{-1})} \leq 1$, respectively. The amount of buys or sells initiated by insiders relative to the total number of buys or sells on information events with positive or negative direction of private information is constant.[67] But in contrast to static models, the trading intensity is allowed to vary over trading days.

Since informed traders only buy (sell) on good-news (bad-news) trading days, logarithms of conditional expected durations of sells (buys) on good-news (bad-news) trading days do not differ from their baseline specification.

$$\psi^{\mathscr{G}}_{-1,i,d} = \psi^{\mathscr{N}}_{-1,i,d}, \tag{6.18}$$

$$\psi^{\mathscr{B}}_{1,i,d} = \psi^{\mathscr{N}}_{1,i,d}. \tag{6.19}$$

Given $\mathscr{F}_{i-1,d}$, we can utilize equations (6.4) and (6.5) to derive the conditional joint density of

[67] We explain this more detailed in section 6.3.

$(y^q_{i,d}, x_{i,d})$:

$$\begin{aligned} & y^q_{i,d} = j \ \wedge \ x_{i,d} = \kappa \\ \iff & \arg\min_{k=-1,1}(T^q_{k,i,d}) = j \ \wedge \ \min_{k=-1,1}(T^q_{k,i,d}) = \kappa \\ \iff & T^q_{j,i,d} = \kappa \ \wedge \ T^q_{-j,i,d} > \kappa \end{aligned}$$

Hence, the conditional joint pdf of $(y^q_{i,d}, x_{i,d})$ for state q on trading day d can be written as

$$p^q_{\left(y^q_{i,d}, x_{i,d}\right)}\left(j, \kappa \mid \mathscr{F}_{i-1,d}\right) = f_{T^q_{j,i,d}}\left(\kappa \mid \mathscr{F}_{i-1,d}\right) \cdot S_{T^q_{-j,i,d}}\left(\kappa \mid \mathscr{F}_{i-1,d}\right) \tag{6.20}$$

with $j \in \{-1, 1\}, \kappa > 0$ and $f_{T^q_{j,i,d}}$ and $S_{T^q_{-j,i,d}}$ denote the pdf and survivor function of $T^q_{j,i,d}$ and $T^q_{-j,i,d}$, respectively.

Let $\mathscr{O}_d$ denote the observation set for trading day d which contains information about waiting times, trade directions, exogenous variables as well as initial values for the autoregressive specifications in the processes of waiting times of buys and sells. Hence, $\boldsymbol{\mathscr{O}} = \{\mathscr{O}_d : d = 1, \dots, D\}$ represents the complete sequence of observation sets for trading days under consideration. Given $\mathscr{O}_d$, the joint density of durations and trade directions for trading day d with state $q \in Q$ is given by

$$\begin{aligned} p^q_d := p^q_d\left(\theta_{\text{dyn}} \mid \mathscr{O}_d\right) &= \prod_{i=1}^{\mathscr{I}_d} p^q_{\left(y^q_{i,d}, x_{i,d}\right)}\left(j_i, \kappa_i \mid \mathscr{F}_{i-1,d}\right) \\ &= \prod_{\{i:\, j_i = 1\}} p^q_{\left(y^q_{i,d}, x_{i,d}\right)}\left(1, \kappa_i \mid \mathscr{F}_{i-1,d}\right) \cdot \prod_{\{i:\, j_i = -1\}} p^q_{\left(y^q_{i,d}, x_{i,d}\right)}\left(-1, \kappa_i \mid \mathscr{F}_{i-1,d}\right), \end{aligned} \tag{6.21}$$

with observed trade direction j_i and observed durations $\kappa_i > 0$.

The vector of model parameters, θ_{dyn}, depends on parameters of the ALACD specifications for buys and sells and possibly additional time-invariant parameters of the waiting times' distributions.

Since point processes for buys and sells after the $\mathscr{I}_d$-th transaction do not arrive before official closing of the market, we cannot determine the initiator of the trade nor can we observe the corresponding duration. Taking this into account, we incorporate the additional factor

$$S_{T^q_{1,i,d}}\left(\text{RT}_{\mathscr{I}_d+1,d} \mid \mathscr{F}_{i-1,d}\right) S_{T^q_{-1,i,d}}\left(\text{RT}_{\mathscr{I}_d+1,d} \mid \mathscr{F}_{i-1,d}\right) \tag{6.22}$$

into the joint density in equation (6.21). The remaining trading time after the $\mathscr{I}_d$-th transaction is represented by $\text{RT}_{\mathscr{I}_d+1,d} = \text{TT}_d - \sum_{i=1}^{\mathscr{I}_d} x_{i,d}$ with total trading time TT_d.

We will conclude this introductory section about dynamic models for the probability of informed trading with a more useful notation of the general conditional intensity function in

equation (6.1). According to Engle and Russell (1998), we can rewrite the conditional intensity function as

$$\lambda_{T^q_{j,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) = \lambda_{\rho_{j,i,d}}\left(\frac{x}{\exp\left(\psi^q_{j,i,d}\right)}\right)\frac{1}{\exp\left(\psi^q_{j,i,d}\right)} \tag{6.23}$$

with duration $x = (t - t_{i-1}) > 0$.

As we will see later, equation (6.23) is essential for calculating the probability of informed trading in the dynamic setup.

6.1 Model by Tay, Ting, Tse and Warachka (PIN-ALACD)

Tay, Ting, Tse, and Warachka (2009) assume that buys and sells follow latent stochastic point processes with time-varying intensities whose waiting times are exponentially distributed. According to equations (6.10) and (6.11), conditional expectations of $T^q_{j,i,d}$ are given by $\exp\left(\psi^q_{j,i,d}\right)$. Since the first moment of the exponential distribution is identical to its scale parameter[68], given the information set $\mathcal{F}_{i-1,d}$, the pdf and survivor function of $T^q_{j,i,d}$ can be written as

$$f_{T^q_{j,i,d}}\left(x; \exp\left(\psi^q_{j,i,d}\right) \mid \mathcal{F}_{i-1,d}\right) = \frac{1}{\exp\left(\psi^q_{j,i,d}\right)}\exp\left(-\frac{x}{\exp\left(\psi^q_{j,i,d}\right)}\right) \quad \text{and} \tag{6.24}$$

$$S_{T^q_{j,i,d}}\left(x; \exp\left(\psi^q_{j,i,d}\right) \mid \mathcal{F}_{i-1,d}\right) = \exp\left(-\frac{x}{\exp\left(\psi^q_{j,i,d}\right)}\right). \tag{6.25}$$

with $x > 0$. Since the mean of an exponential distribution is identical to its scale parameter, ALACD models are applied to the logarithmic scale parameters of buys' and sells' interarrival times distributions.

An ALACD(1,1,1) model is utilized in which the lagged signed logarithm of the volume assigned to each transaction record acts as sole exogenous variable. According to the formulations of the conditional expected durations on no-news days in equations (6.12) and (6.13), lagged logarithmic conditional expected durations $\psi^{\mathcal{N}}_{j,i-1,d}$ and lagged logarithmic observed durations $\log x_{i-1,d}$ are included. Hence, the specifications for the logarithmic conditional expected durations on a no-news trading day d are given by

$$\psi^{\mathcal{N}}_{j,i,d} = \nu_{j,-1}\,\mathbb{1}_{\{y_{i-1}=-1\}} + \nu_{j,1}\,\mathbb{1}_{\{y_{i-1}=1\}} + \alpha_j \psi^{\mathcal{N}}_{j,i-1,d} + \beta_j \log x_{i-1,d} + \zeta_j y_{i-1,d} \log v_{i-1,d}, \tag{6.26}$$

where $y_{i-1,d} \log v_{i-1,d}$ denotes the lagged signed logarithmic volume.

[68] The exponential distribution is defined in many introductory statistical textbooks (e.g., see Johnson, Kotz, and Balakrishnan 1994).

According to equations (6.14) – (6.15), specifications on good-news and bad-news trading days are based on equation (6.26). Influence on durations induced by information-based trading is captured by a single parameter $\tau = \tau_1 = \tau_{-1}$. Hence, we can write the ALACD specification for buys and sells on a good-news trading day d as

$$\begin{aligned}\psi^{\mathcal{G}}_{1,i,d} &= \psi^{\mathcal{N}}_{1,i,d} - \tau \\ &= \nu_{1,-1}\,\mathbb{1}_{\{y_{i-1}=-1\}} + \nu_{1,1}\,\mathbb{1}_{\{y_{i-1}=1\}} - (1-\alpha_1)\tau + \alpha_1\psi^{\mathcal{G}}_{1,i-1,d} + \beta_1 \log x_{i-1,d} + \\ &\quad \zeta_1 y_{i-1,d} \log \nu_{i-1,d} \quad \text{and} \qquad (6.27)\\ \psi^{\mathcal{G}}_{-1,i,d} &= \psi^{\mathcal{N}}_{-1,i,d}. \qquad (6.28)\end{aligned}$$

Likewise, for a bad-news trading day d they are given by

$$\begin{aligned}\psi^{\mathcal{B}}_{1,i,d} &= \psi^{\mathcal{N}}_{1,i,d} \quad \text{and} \qquad (6.29)\\ \psi^{\mathcal{B}}_{-1,i,d} &= \psi^{\mathcal{N}}_{-1,i,d} - \tau \\ &= \nu_{-1,-1}\,\mathbb{1}_{\{y_{i-1}=-1\}} + \nu_{-1,1}\,\mathbb{1}_{\{y_{i-1}=1\}} - (1-\alpha_{-1})\tau + \alpha_{-1}\psi^{\mathcal{B}}_{-1,i-1,d} + \beta_{-1} \log x_{i-1,d} + \\ &\quad \zeta_{-1} y_{i-1,d} \log \nu_{i-1,d}. \qquad (6.30)\end{aligned}$$

Since the waiting times for both trade directions are supposed to follow exponential distributions, we can rewrite equation (6.20) as

$$p^{q}_{\left(y^{q}_{i,d},x_{i,d}\right)}\left(j,\kappa \,\middle|\, \mathcal{F}_{i-1,d}\right) = \frac{1}{\exp\left(\psi^{q}_{j,i,d}\right)} \exp\left(-\frac{\kappa}{\exp\left(\psi^{q}_{j,i,d}\right)}\right) \exp\left(-\frac{\kappa}{\exp\left(\psi^{q}_{-j,i,d}\right)}\right). \qquad (6.31)$$

It can be shown that duration and trade direction are independent, given information until $(i-1)$-th transaction $\mathcal{F}_{i-1,d}$ (see Tay, Ting, Tse, and Warachka 2009).

State probabilities $\pi_{q,d}$ for trading days are modeled using an approach similar to logistic regression. Let V^B_d and V^S_d denote the daily aggregated volume of buyer- and seller-initiated transactions, respectively. Average volumes of buys and sells over all underlying trading days are given by $\bar{V}^B$ and $\bar{V}^S$. State probability $\pi_{\mathcal{N},d}$ that trading day d resides in no-news condition compares daily total aggregated volume, $V^B_d + V^S_d$, with its average value, $\bar{V}^B + \bar{V}^S$,

$$\pi_{\mathcal{N},d} = \frac{1}{1 + \exp\left(\delta_1 + \delta_2\left(\log\left(V^B_d + V^S_d\right) - \log\left(\bar{V}^B + \bar{V}^S\right)\right)\right)}, \qquad (6.32)$$

where the parameter δ_2 is expected to be positive and δ_1 can be interpreted as intercept or scaling parameter and underlies no sign restriction. The conditional probability of good-news trading days given an information event, $\pi_{\mathcal{G},d,\text{cond.}}$, is calculated incorporating the relation between daily aggregated volume of buys and sells and their averages,

$$\pi_{\mathcal{G},d,\text{cond.}} = \frac{1}{1 + \exp\left(\delta_3\left(\log\left(V^S_d\right) - \log\left(\bar{V}^S\right)\right) - \delta_4\left(\log\left(V^B_d\right) - \log\left(\bar{V}^B\right)\right)\right)}, \qquad (6.33)$$

where both coefficients δ_3 and δ_4 are expected to be positive. Thus, unconditional probability of private information with positive direction entering the market on trading day d is given by

$$\pi_{\mathcal{G},d} = \big(1 - \pi_{\mathcal{N},d}\big)\pi_{\mathcal{G},d,\text{cond.}}. \tag{6.34}$$

Finally, the probability of a bad-news trading day d calculates as

$$\pi_{\mathcal{B},d} = \big(1 - \pi_{\mathcal{N},d}\big)\big(1 - \pi_{\mathcal{G},d,\text{cond.}}\big). \tag{6.35}$$

At this point, we need to explain potential shortcomings of the approach for state probabilities by Tay, Ting, Tse, and Warachka (2009). The ad-hoc specifications of probabilities in their model only utilize daily aggregated volumes of buys and sells and averages thereof. These measures are not endogenously modeled and do not have any impact on the autoregressive setting of interarrival times. Trading days with a small number of trades with a large volume can already trigger an increase in equations (6.34) or (6.35). Similarly, this can be induced by trading days with slightly higher trading activities and ordinary volume of trades. While one expects a higher volume of trades or a higher number of transactions on trading days on which insiders enter the market, this can also be caused by noise traders. Therefore the probability modeling by Tay, Ting, Tse, and Warachka (2009) is prone to overreacting to trading activities by non-informed traders.

For the PIN-ALACD model to be confident about labeling a trading day as non-information event the total aggregated volume for this day may vary only little from the corresponding average value over the total time span of the underlying data. Depending on the length of the time range and the marketplace the equities under consideration are traded such constellations are rare. To achieve a (very) high probability for an information event, the aggregated volume of the corresponding trade direction needs be (very) high while the opposite trading direction is close to its overall average. This together leads to the situation that the PIN-ALACD model is unsure about the condition for almost every single trading. Furthermore, the volumes of buys and sells are treated as exogenous variables which implies the knowledge of the winner of the *sprint* between the durations of buys and sells over all trading days. This does not match the dynamic approaches for the probability of informed trading and leads to circular reasoning.

Harnessing equation (6.21) we can write the likelihood function for a total of D trading days as

$$\mathscr{L}\left(\theta_{\text{dyn}} \,\middle|\, \mathcal{O}\right) = \sum_{d=1}^{D} \log\left(\sum_{q\in Q} \pi_{q,d} \cdot p_d^q\right), \tag{6.36}$$

where the length of the parameter vector θ_{dyn} equals 15. It consists of four parameters which are involved in the modelling of state probabilities $\pi_{q,d}$ ($\delta_1, \delta_2, \delta_3, \delta_4$), five parameters of the ALACD specification for each trade direction ($\nu_{j,-1}, \nu_{j,1}, \alpha_j, \beta_j, \nu_j$) and the adjustment parameter τ to control for insider trading.

In the original work by Tay, Ting, Tse, and Warachka (2009), the additional factor in equation (6.22) is not incorporated in the joint density of trade directions and durations on trading day d. This can be interpreted as setting the remaining trading time after the $\mathcal{I}_d$-th transaction to zero and therefore artificially shorten total trading time. To exactly reproduce their approach, we also do not utilize equation (6.22) in optimizations. However, remaining trading times are usually not larger than several seconds and therefore the effects of ignoring this information should be limited.

The calculation of the probability of informed trading in the context of the dynamic models is elaborated in section 6.3. The section is based on the method described in the work by Tay, Ting, Tse, and Warachka (2009) but we generalize it to differentiate between PIN driven by good and bad news. Since the empirical results and graphics in chapter 11 rely on our generalized PIN estimators, we refrain to describe the baseline method by Tay, Ting, Tse, and Warachka (2009) in this section of the thesis.

6.2 PIN-HMM Model

Our model for the probability of informed trading is based on and extends the model introduced by Tay, Ting, Tse, and Warachka (2009) which was discussed in the previous section. However, we choose a completely different approach for modelling state probabilities. In the PIN-ALACD model, states of trading days are assumed to be independent. We relax this assumption by utilizing hidden Markov models for the state probabilities of trading days which yields dependencies in the sequence of trading days' conditions in such a way that the current state depends on its predecessor. This considers that information events may last for several trading periods or that private news are not instantaneously available to the entity of information-based market participants. Private information diffuses among the group of insiders so that they are heterogeneously informed for a certain amount of time.

Furthermore, we gain additional insights by transition probabilities which show the most probable state of the next trading day if the current resides in state q. To the best of our knowledge, we are the first introducing hidden Markov models in the context of dynamic models for estimating the probability of informed trading.[69]

While Tay, Ting, Tse, and Warachka (2009) assume buys and sells to follow latent stochastic point processes whose interarrival times are exponentially distributed, our empirical results show that this distribution is not an appropriate choice for the durations of buys and sells (see chapter 11). To fortify our findings, several publications which reject the assumption of an exponential distribution for ACD models are mentioned in the work by Pacurar (2008). The flat conditional intensity function is often not sufficient, especially in financial applications (see Pacurar 2008).

[69] The work by Yin and Zhao (2015) utilizes hidden Markov chains in the context of static PIN models but instead of modelling the sequence of trading days' conditions, they model the aggregated number of buys and sells per day.

We assume the durations of both trade directions to follow a two-parameter Weibull distribution with pdf $f_{T^q_{j,i,d}}$ and survivor function $S_{T^q_{j,i,d}}$ which are given by[70]

$$f_{T^q_{j,i,d}}(x; k_j, \exp(\phi^q_{j,i,d})) = \frac{k_j}{\exp(\phi^q_{j,i,d})}\left(\frac{x}{\exp(\phi^q_{j,i,d})}\right)^{k_j-1} \exp\left(-\left(\frac{x}{\exp(\phi^q_{j,i,d})}\right)^{k_j}\right), \tag{6.37}$$

$$S_{T^q_{j,i,d}}(x; k_j, \exp(\phi^q_{j,i,d})) = \exp\left(-\left(\frac{x}{\exp(\phi^q_{j,i,d})}\right)^{k_j}\right), \tag{6.38}$$

with $x > 0$, shape parameter $k_j > 0$ and scale parameter $\exp(\phi^q_{j,i,d})$. It can easily be seen that the Weibull distribution encompasses the exponential distribution for the special case that $k_j = 1$. We do not impose a restriction on the shape parameters of buys and sells to be identical.

The expected conditional durations of buys and sells are proportional to the scale parameters of corresponding Weibull distributions:

$$\mathbb{E}\left(T^q_{1,i,d} \mid \mathcal{F}_{i-1}\right) = \exp(\psi^q_{1,i,d}) = \exp(\phi^q_{1,i,d})\Gamma\left(1 + \frac{1}{k_1}\right), \tag{6.39}$$

$$\mathbb{E}\left(T^q_{-1,i,d} \mid \mathcal{F}_{i-1}\right) = \exp(\psi^q_{-1,i,d}) = \exp(\phi^q_{-1,i,d})\Gamma\left(1 + \frac{1}{k_{-1}}\right). \tag{6.40}$$

Both shape parameters, k_1 and k_{-1}, are assumed to be time-invariant.

Baseline specifications for buys and sells on no-news trading days are identical to those in the PIN-ALACD model and can be read from equation (6.26). According to equations (6.14) – (6.15), specifications for the logarithms of the conditional expected durations of buys on good-news days and sells on trading days which reside in bad-news state are given by the baseline specifications reduced by the constant positive parameters τ_1 and τ_{-1}, respectively.

In the Weibull case, we can rewrite equation (6.20) as

$$p^q_{(y^q_{i,d}, x_{i,d})}(j, \kappa \mid \mathcal{F}_{i-1,d}) = \frac{k_j}{\exp(\phi^q_{j,i,d})}\left(\frac{\kappa}{\exp(\phi^q_{j,i,d})}\right)^{k_j-1} \exp\left(-\left(\frac{\kappa}{\exp(\phi^q_{j,i,d})}\right)^{k_j}\right) \cdot \exp\left(-\left(\frac{\kappa}{\exp(\phi^q_{-j,i,d})}\right)^{k_{-j}}\right). \tag{6.41}$$

Conditional on $\mathcal{F}_{i-1,d}$, duration and trade direction are no longer independent in our approach if $k_1 \neq k_{-1}$.

[70] Tay, Ting, Tse, and Warachka (2007) and Tay, Ting, Tse, and Warachka (2009) give some key results for the Weibull distribution used for ACD residuals but report that results are similar to the exponential distribution and therefore use the simpler one. Furthermore, they assume the shape parameters of distributions for buys' and sells' waiting times to be identical.

6.2.1 Hidden Markov Models (HMM)

We model the sequence of trading days' conditions with a Markov chain, which introduces dependencies between the current state and its predecessor. An information event may highly likely be followed by another trading day on which private information enter the market. This may be the case, especially, if we switch from a daily basis to (very) short intraday intervals.

However, states of trading days are not observable, we only can observe observation set $\mathcal{O}_d$ which depends on the underlying condition. Hence, this leads us to hidden Markov models (HMM). This section intends to give a brief introduction to (hidden) Markov chains and to illustrate the general idea of our approach.[71] Furthermore, in the subsequent section, we give a compact description of the forward-backward algorithm which we utilize for estimation purposes.

A stochastic process in discrete time with a finite or countable set of states and which satisfies the Markov property is called a Markov Chain. A definition of the Markov property is given below (similar to Isaacson and Madsen 1976):

Definition 6.1. *A stochastic process* $\{X_d\}$*, with* $d = 1, 2, \dots, D$*, with finite or countable state space* $Q = \{1, 2, \dots\}$ *is said to satisfy the Markov property if for every* $d \geq 2$ *and all states* $q_1, q_2, \dots, q_d$ *it is true that*

$$\Pr\big(X_d = q_d \,\big|\, X_{d-1} = q_{d-1}, X_{d-2} = q_{d-2}, \dots, X_1 = q_1\big) = \Pr\big(X_d = q_d \,\big|\, X_{d-1} = q_{d-1}\big) \tag{6.42}$$

Markov Chains are often supposed to be time-homogeneous which means that the conditional probability of being in state $q_d \in Q = \{\mathcal{N}, \mathcal{G}, \mathcal{B}\}$ at trading day d, given that the condition of its predecessor is known, is time-invariant. Thus, a general definition can be written as (similar to Isaacson and Madsen 1976):

Definition 6.2. *A discrete-time Markov chain is said to be homogeneous in time if the probability of going from one state to another is independent of the time at which the step is being made. That is, for all states* m *and* n *with* $m, n \in Q$*,*

$$\Pr(X_d = m \mid X_{d-1} = n) = \Pr(X_{d+l} = m \mid X_{d+l-1} = n), \tag{6.43}$$

for $l = -(d-2), -(d-3), \dots, -1, 0, 1, 2, \dots, D-d$ *and* $d = 1, 2, \dots, D$*.*

The conditional probabilities in equation (6.43) are typically called *transition probabilities* and stored in a square matrix (*transition matrix*) with the pre-transition state as the row and the post-transition state as the column. This matrix is row-stochastic which means all elements are non-negative and do not exceed a value of 1. In addition, row-sums of the transition matrix are always identical to unity.

[71] For instance, the works by Couvreur (1996) or Zucchini and MacDonald (2009) provide much more detailed descriptions of (hidden) Markov chains and their extensions.

Let $\boldsymbol{\omega}_d = \left(\omega_{\mathcal{N},d}, \omega_{\mathcal{G},d}, \omega_{\mathcal{B},d}\right)$ denote the probability distribution of states for trading day d with $\omega_{q,d} = \Pr\left(X_d = q\right)$ and $\sum_{q\in Q} \omega_{q,d} = 1$. Due to the time-homogeneous property of the Markov Chain, probability distribution of states for trading day d can be expressed by initial state probabilities $\boldsymbol{\omega}_0$ and d-fold multiplication of the transition matrix A:

$$\begin{aligned} \boldsymbol{\omega}_1 &= \boldsymbol{\omega}_0 A \\ \boldsymbol{\omega}_2 &= \boldsymbol{\omega}_0 A^2 \\ &\vdots \\ \boldsymbol{\omega}_d &= \boldsymbol{\omega}_0 A^d \end{aligned} \tag{6.44}$$

We suppose the Markov chain to be stationary which implies that the probability distribution for states is identical on each trading day d. The following proposition together with its proof can be found in Konstantopoulos (2009).

Proposition 6.1. *A discrete-time Markov chain* $(X_d, d = 1, 2, \dots, D)$ *is stationary if and only if it is time-homogeneous and* X_d *has the same distribution as* X_l *for all* d *and* l.

This means that a time-homogeneous Markov chain is stationary if and only if the distribution of states does not change over trading days. According to equation (6.44) $\boldsymbol{\omega}_d$ depends on the initial state probabilities in such a way that the Markov chain is stationary if and only if the following relation holds for $\boldsymbol{\omega}_0$:

$$\boldsymbol{\omega}_0 = \boldsymbol{\omega}_0 A. \tag{6.45}$$

For the remainder of this work $\boldsymbol{\omega} = \left(\omega_{\mathcal{N}}, \omega_{\mathcal{G}}, \omega_{\mathcal{B}}\right)$ denotes the stationary distribution of the Markov chain and is calculated according to equation (6.45). Similarly to previously presented approaches we differentiate between three states: namely no-news, good-news, bad-news. Hence, the finite state space for our Markov Chain of trading days' conditions is given by $Q = \{\mathcal{N}, \mathcal{G}, \mathcal{B}\}$. This yields a transition matrix $A \in \mathbb{R}^{3\times 3}$ with

$$A = \begin{pmatrix} a_{\mathcal{N}\mathcal{N}} & a_{\mathcal{N}\mathcal{G}} & a_{\mathcal{N}\mathcal{B}} \\ a_{\mathcal{G}\mathcal{N}} & a_{\mathcal{G}\mathcal{G}} & a_{\mathcal{G}\mathcal{B}} \\ a_{\mathcal{B}\mathcal{N}} & a_{\mathcal{B}\mathcal{G}} & a_{\mathcal{B}\mathcal{B}} \end{pmatrix}, \tag{6.46}$$

where $a_{mn} = \Pr(X_d = n \mid X_{d-1} = m)$ with $m, n \in Q$. The corresponding state diagram is shown in figure 6.1. A Markov chain is fully described by the transition matrix A and the stationary distribution $\boldsymbol{\omega}$.

The case that trading days' conditions are independent is also nested in the HMM approach. If all rows of the matrix A in equation (6.46) are identical (i.e., $a_{\mathcal{N}\mathcal{N}} = a_{\mathcal{G}\mathcal{N}} = a_{\mathcal{B}\mathcal{N}}, \dots$), irrelevant of the condition of the current trading day the probability that the next day resides in state $q \in Q$ is the same in all three possible scenarios.

HMM differ from standard Markov chains by assuming that states are not observable. Rabiner and Juang (1986) give a compact but suitable description of the class of HMM:

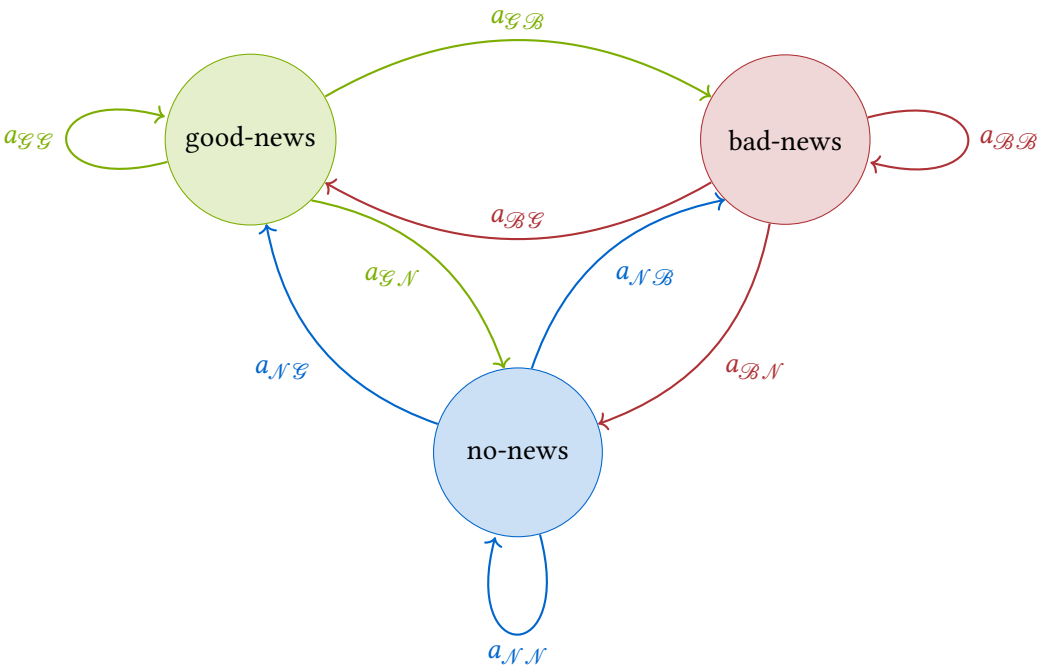

Figure 6.1: Transition probabilities for Markov chain of trading days' conditions.

> An HMM is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols.

Thus, in our setting the underlying non-observable stochastic process is represented by the sequence of trading days' conditions. On each trading day we observe realizations of the latent stochastic point processes for buys and sells.[72] In addition, prices, quotes and volumes are observable. However, we do not make any assumption about the data generating process for these measures and treat them as exogenous variables as they enter the model.

Beside the stationary distribution and transition matrix of the Markov chain, so-called emission densities are required to fully describe the HMM. Given the state of trading day d, durations and trade directions are emitted according to their conditional joint density for which the general structure is given in equation (6.21). Hence, the HMM is fully described by the set $\mathscr{H} = (\omega, A, B)$ with $3 \times D$ matrix of conditional emission densities $B = \{p_d^q\}$.

[72] For the markets under consideration in this work we need to utilize a classification algorithm to assign trade direction to each record in our dataset (see section 8.6).

6.2.2 Forward-Backward Algorithm

Parameters of our HMM are estimated by utilizing the forward-backward algorithm (e. g. see Baum, Petrie, Soules, and Weiss 1970, Couvreur 1996) which intends to maximize the probability of the observation sequence $\mathcal{O}$ given the parameters of our HMM approach,

$$\Pr(\mathcal{O} \mid \mathscr{H}), \tag{6.47}$$

where $\mathscr{H}$ consists of transition probabilities as well as parameters controlling the densities of buys' and sells' interarrival times. The ALACD models for buys and sells each have five parameters for specifying the baseline conditional duration recursion for a no-news trading day. Both trade directions exhibit one parameter for adjustment of conditional durations due to information-based trading. Finally, six entries of the transition matrix are sufficient to determine the switching between states.[73] Therefore, we have a total of 20 parameters which influence the probability in equation (6.47) and need to be estimated.

As the name suggests the estimation algorithm consists of two parts, whereas one directs forward in time and the other backwards. The algorithm delivers likelihood function values and so-called *smoothed probabilities* $\pi_{q,d}$, where $\pi_{q,d}$ is the probability of trading day d residing in state $q \in Q = \{\mathscr{N}, \mathscr{G}, \mathscr{B}\}$ given the complete sequence of observation sets $\mathcal{O}$.

Define the forward term $\alpha_q(d)$ as the joint probability of observing the first d days' data and being in state q on trading day d,

$$\alpha_q(d) = \Pr\big(\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_d; q_d = q \,\big|\, \mathscr{H}\big), \quad q \in Q. \tag{6.48}$$

Equation (6.48) depends on lagged forward probabilities of all states and corresponding elements of the transition matrix. It can be evaluated according to the recursion

$$\alpha_q(d) = p_d^q \cdot \sum_{k \in Q} \alpha_k(d-1) a_{k,q}, \quad \forall q \in Q. \tag{6.49}$$

The recursion is initialized with

$$\alpha_q(1) = \omega_q p_1^q, \quad \forall q \in Q = \{\mathscr{N}, \mathscr{G}, \mathscr{B}\}. \tag{6.50}$$

Let the backward term $\beta_q(d)$ be the joint probability of the partial observation sequence after trading day d given the state of day d is q,

$$\beta_q(d) = \Pr\big(\mathcal{O}_{d+1}, \mathcal{O}_{d+2}, \ldots, \mathcal{O}_D \,\big|\, q_d = q, \mathscr{H}\big), \quad q \in Q. \tag{6.51}$$

Recursion of backward probabilities is initialized with

$$\beta_q(D) = 1, \quad \forall q \in Q. \tag{6.52}$$

[73] We can harness the fact that the transition matrix is row-stochastic. Hence, only two elements of each row need to be estimated.

Backward terms for the remaining trading days $d = D-1, \dots, 1$ can be calculated with the following formula

$$\beta_q(d) = \sum_{k \in Q} a_{q,k} p_{d+1}^k \beta_k(d+1), \quad \forall q \in Q. \tag{6.53}$$

The joint probability of the complete observation sequence $\mathcal{O}$ and state q on trading day d can be written as

$$\begin{aligned} \Pr\big(\mathcal{O}, q_d = q \,\big|\, \mathcal{H}\big) &= \Pr\big(\mathcal{O}_1, \dots, \mathcal{O}_d; q_d = q \,\big|\, \mathcal{H}\big) \cdot \Pr\big(\mathcal{O}_{d+1}, \dots, \mathcal{O}_D \,\big|\, \mathcal{O}_1, \dots, \mathcal{O}_d; q_d = q, \mathcal{H}\big) \\ &= \Pr\big(\mathcal{O}_1, \dots, \mathcal{O}_d; q_d = q \,\big|\, \mathcal{H}\big) \cdot \Pr\big(\mathcal{O}_{d+1}, \dots, \mathcal{O}_D \,\big|\, q_d = q, \mathcal{H}\big) \\ &= \alpha_q(d)\beta_q(d), \end{aligned} \tag{6.54}$$

since the conditional probability of observing $\mathcal{O}_{d+1}, \dots, \mathcal{O}_D$, given that we know the condition of trading day d, is independent of the observation sets of the preceding trading days $\mathcal{O}_1, \dots, \mathcal{O}_d$. Hence the probability of the complete observation sequence is given as the sum of equation (6.54) over all possible states of a trading day, namely no-news, good-news and bad-news,

$$\begin{aligned} \Pr(\mathcal{O} \mid \mathcal{H}) &= \sum_{k \in Q} \Pr\big(\mathcal{O}, q_d = k \,\big|\, \mathcal{H}\big) \\ &= \sum_{k \in Q} \alpha_k(d)\beta_k(d). \end{aligned} \tag{6.55}$$

This relation holds for any trading day d, so by letting $d = D$ and incorporating equation (6.52) the likelihood function can be written as

$$\Pr(\mathcal{O} \mid \mathcal{H}) = \sum_{k \in Q} \alpha_k(D). \tag{6.56}$$

Finally, the (conditional) smoothed probability $\pi_{q,d}$ that trading day d resides in state q can be calculated with

$$\pi_{q,d} = \Pr\big(q_d = q \,\big|\, \mathcal{O}, \mathcal{H}\big) = \frac{\alpha_q(d)\beta_q(d)}{\sum_{k \in Q} \alpha_k(D)}, \quad q \in Q. \tag{6.57}$$

Exemplarily, forward- and backward-probabilities involved in computing the joint probability that the Markov chain spends trading day d in a no-news state and observing $\mathcal{O}$ are displayed in figure 6.2.

Similar to static models, floating point exceptions are a potential problem in optimizations in the context of the dynamic models for the probability of informed trading. Our stable implementations of all involved functions, which minimize the occurrence of them, are described in chapter 7. In addition, to even further stabilize the estimation procedure, we decided to reparameterize the transition probabilities. The corresponding expression can be found in equations (7.1) - (7.6).

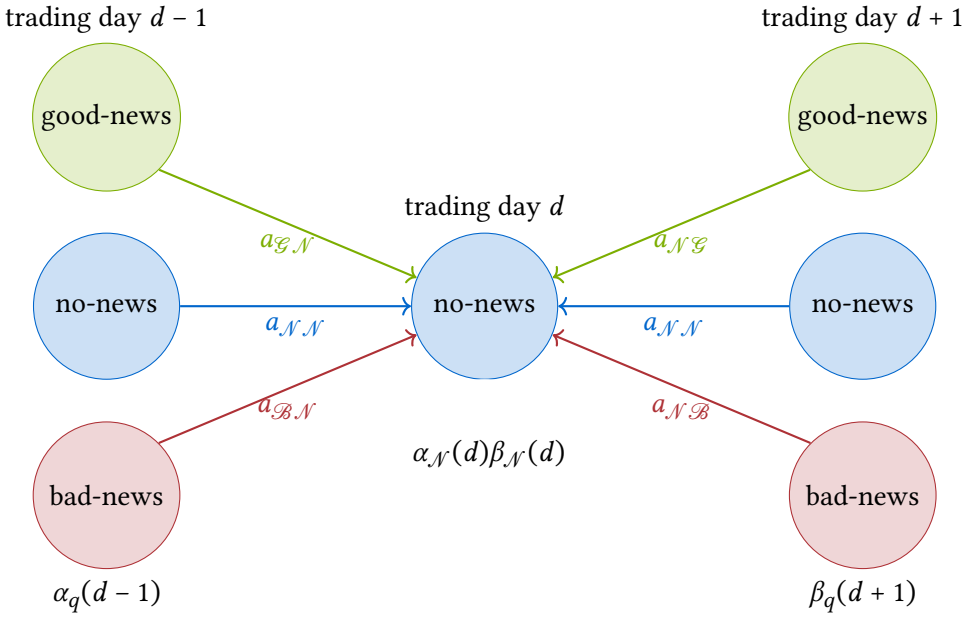

Figure 6.2: Visualization of forward- and backward-probabilities involved in computing the joint probability that the Markov chain spends trading day d in a no-news state and observing $\mathcal{O}$.

At the time of writing, there are no equivalents to the algorithms for generating sets of initial values in static PIN models. Therefore, we follow Tay, Ting, Tse, and Warachka (2009) and generate multiple, randomly drawn, sets of initial values for optimization purposes and utilize the best result to calculate the probability of informed trading and all related measures.

6.3 Computation of the Probability of Informed Trading

The probability of informed trading in the dynamic model setup can be calculated harnessing the conditional intensity function of the point processes for buys and sells in equation (6.23). According to Basu and Rigdon (2001), the non-homogeneous Poisson processes for buys and sells have the property that the conditional expected number of transactions in the interval $(t_{i-1}, t]$, with $t > t_{i-1}$, on trading day d which resides in state q is equal to the conditional cumulative intensity function given by

$$\mathbb{E}\left(N_{(t_{i-1},t]} \mid \mathcal{F}_{i-1,d}\right) = \Lambda_{T^q_{j,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) = \int_{t_{i-1}}^{t} \lambda_{T^q_{j,i,d}}\left(r \mid \mathcal{F}_{i-1,d}\right) \, \mathrm{d}r. \tag{6.58}$$

In the PIN-ALACD model waiting times of buys and sells are supposed to follow exponential distributions. To fulfill the assumption that $\mathbb{E}\left(\rho_{j,i,d}\right) = 1$, we need to set the scale parameter for the distribution of ALACD error terms to unity. According to equation (6.23) and since the hazard function of an exponential distribution is identical to its scale parameter, the hazard function of $\rho_{j,i,d}$ is constant and identical to 1.

The conditional intensity function for trade direction j at timestamp $t > t_{i-1}$ on trading day d which resides in no-news state is then given by

$$\lambda_{T^{\mathcal{N}}_{j,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) = \lambda_{\rho_{j,i,d}}\left(\frac{t - t_{i-1}}{\exp\left(\psi^{\mathcal{N}}_{j,i,d}\right)}\right)\frac{1}{\exp\left(\psi^{\mathcal{N}}_{j,i,d}\right)} = \frac{1}{\exp\left(\psi^{\mathcal{N}}_{j,i,d}\right)}. \tag{6.59}$$

Utilizing the relation in equation (6.58), the expected number of buyer- or seller-initiated transactions in the half-open interval $(t_{i-1}, t_i]$ on a no-news trading day d can be calculated as the corresponding conditional cumulative intensity function,

$$\begin{aligned}\Lambda_{T^{\mathcal{N}}_{j,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) &= \int_{t_{i-1}}^{t} \frac{1}{\exp\left(\psi^{\mathcal{N}}_{j,i,d}\right)}\ \mathrm{d}r \\ &= \frac{t - t_{i-1}}{\exp\left(\psi^{\mathcal{N}}_{j,i,d}\right)}.\end{aligned} \tag{6.60}$$

According to equations (6.27) and (6.29) the expected numbers of buys and sells on good-news and bad-news days can be computed with

$$\Lambda_{T^{\mathcal{G}}_{1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) = \exp(\tau)\Lambda_{T^{\mathcal{N}}_{1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right), \quad \Lambda_{T^{\mathcal{G}}_{-1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) = \Lambda_{T^{\mathcal{N}}_{-1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) \tag{6.61}$$

$$\Lambda_{T^{\mathcal{B}}_{1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) = \Lambda_{T^{\mathcal{N}}_{1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right), \quad \Lambda_{T^{\mathcal{B}}_{-1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) = \exp(\tau)\Lambda_{T^{\mathcal{N}}_{-1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right), \tag{6.62}$$

respectively.

For the Weibull distribution in the PIN-HMM model, the scale parameter $\exp\left(\phi^{q}_{j,i,d}\right)$ is proportional to the conditional expected duration. This procedure yields modified ALACD error terms $\tilde{\rho}_{j,i,d} = \Gamma\left(1 + \frac{1}{k_j}\right)\rho_{j,i,d}$ with $\mathbb{E}\left(\tilde{\rho}_{j,i,d}\right) = \Gamma\left(1 + \frac{1}{k_j}\right)$. The corresponding hazard function is therefore given by $\lambda_{\tilde{\rho}_{j,i,d}}(x) = k_j x^{k_j - 1}$ with shape parameter k_j and unitary scale parameter. Again utilizing equation (6.23), the conditional intensity function of trade direction j on trading day d with no-news state is given by

$$\lambda_{T^{\mathcal{N}}_{j,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) = \lambda_{\rho_{j,i,d}}\left(\frac{t - t_{i-1}}{\exp\left(\phi^{\mathcal{N}}_{j,i,d}\right)}\right)\frac{1}{\exp\left(\phi^{\mathcal{N}}_{j,i,d}\right)} = \frac{k_j(t - t_{i-1})^{k_j - 1}}{\exp\left(\phi^{\mathcal{N}}_{1,i,d}\right)^{k_j}}. \tag{6.63}$$

Therefore the corresponding conditional cumulative intensity functions of buys and sells in the PIN-HMM model can be written as

$$\Lambda_{T^{\mathcal{N}}_{j,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) = \left(\frac{t - t_{i-1}}{\exp\left(\phi^{\mathcal{N}}_{j,i,d}\right)}\right)^{k_j}. \tag{6.64}$$

Similarly to the PIN-ALACD model, we can express the expected number of buys and sells on good-news and bad-news trading days in terms of expected amounts of corresponding transactions on no-news days by utilizing equations (6.14) and (6.15),

$$\Lambda_{T^{\mathcal{G}}_{1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) = \exp(\tau_1 k_1)\Lambda_{T^{\mathcal{N}}_{1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right), \quad \Lambda_{T^{\mathcal{G}}_{-1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) = \Lambda_{T^{\mathcal{N}}_{-1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) \tag{6.65}$$

$$\Lambda_{T^{\mathcal{B}}_{1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) = \Lambda_{T^{\mathcal{N}}_{1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right), \quad \Lambda_{T^{\mathcal{B}}_{-1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right) = \exp(\tau_{-1} k_{-1})\Lambda_{T^{\mathcal{N}}_{-1,i,d}}\left(t \mid \mathcal{F}_{i-1,d}\right). \tag{6.66}$$

The aggregated expected total numbers of buys and sells for trading day d depending on its condition q can be calculated by summing the corresponding conditional cumulative intensity function over all transactions, and the time window starting from the timestamp of the last observable trade until official market closing, which yield

$$\Lambda^{q}_{1,d} = \sum_{i=1}^{\mathcal{I}_d+1} \Lambda_{T^{q}_{1,i,d}}\left(t_i \mid \mathcal{F}_{i-1,d}\right) \quad \text{and} \quad \Lambda^{q}_{-1,d} = \sum_{i=1}^{\mathcal{I}_d+1} \Lambda_{T^{q}_{-1,i,d}}\left(t_i \mid \mathcal{F}_{i-1,d}\right), \tag{6.67}$$

where t_i is the corresponding timestamp of i-th transaction, $t_0 = 0$ and $t_{\mathcal{I}_d+1}$ denotes the time of official market closing.

We know that informed traders only enter the market if they are triggered by private news either of positive or negative direction. Although sells (buys) on an information event d driven by good-news (bad-news) are initiated by noise traders, the expected number of sells (buys) varies from the amount on no-news days. There are typically more trading activities (noise trading plus informed trading) on information events which yield smaller durations and therefore the total number of transactions is higher. Hence, on a trading day d, on which private information enter the market, the expressions in equation (6.67) are updated more often. Assuming shape parameters smaller than unity for the Weibull distributions of both trade directions[74], this induces that the amount of noise trading is higher on information events than it is on trading days without private information. Uninformed traders are infected by the increased trading activities on information events and in contrast to the static models, the additional number of transactions is distributed over both types of market attendees, informed and uninformed. This shows that it is crucial to correctly classify the conditions of trading days.

[74] The results in chapter 11 show that the shape parameters for buys and sells for all equities under consideration in this work are smaller than unity.

The expected number of buys (sells) on good-news (bad-news) trading days consists of transactions by noise traders as well as insiders. We can readily calculate the expected total number of transactions by informed traders on good-news and bad-news trading days as

$$\Lambda_{1,d}^{\mathcal{G},\text{inf}} = \Lambda_{1,d}^{\mathcal{G}} - \Lambda_{1,d}^{\mathcal{N}} \quad \text{and} \quad \Lambda_{-1,d}^{\mathcal{B},\text{inf}} = \Lambda_{-1,d}^{\mathcal{B}} - \Lambda_{-1,d}^{\mathcal{N}}. \tag{6.68}$$

In the PIN-ALACD model we can rewrite the expressions in equation (6.68) as

$$\begin{aligned} \Lambda_{1,d}^{\mathcal{G}} = \exp(\tau)\Lambda_{1,d}^{\mathcal{N}} \Longrightarrow \Lambda_{1,d}^{\mathcal{G},\text{inf}} = \big(\exp(\tau) - 1\big)\Lambda_{1,d}^{\mathcal{N}} \quad \text{and} \\ \Lambda_{-1,d}^{\mathcal{B}} = \exp(\tau)\Lambda_{-1,d}^{\mathcal{N}} \Longrightarrow \Lambda_{-1,d}^{\mathcal{B},\text{inf}} = \big(\exp(\tau) - 1\big)\Lambda_{-1,d}^{\mathcal{N}}. \end{aligned} \tag{6.69}$$

And similar in the context of our PIN-HMM model as

$$\begin{aligned} \Lambda_{1,d}^{\mathcal{G}} = \exp(\tau_1 k_1)\Lambda_{1,d}^{\mathcal{N}} \Longrightarrow \Lambda_{1,d}^{\mathcal{G},\text{inf}} = \big(\exp(\tau_1 k_1) - 1\big)\Lambda_{1,d}^{\mathcal{N}} \quad \text{and} \\ \Lambda_{-1,d}^{\mathcal{B}} = \exp(\tau_{-1} k_{-1})\Lambda_{-1,d}^{\mathcal{N}} \Longrightarrow \Lambda_{-1,d}^{\mathcal{B},\text{inf}} = \big(\exp(\tau_{-1} k_{-1}) - 1\big)\Lambda_{-1,d}^{\mathcal{N}}. \end{aligned} \tag{6.70}$$

As mentioned earlier, we see that the expected number of buys (sells) on good-news (bad-news) trading days are scaled expected numbers of noise trading with scaling factors (weakly) larger than unity. While the PIN-ALACD model assumes the identical scaling factors for buys and sells on the corresponding information events, our new approach allows to assign each scenario a different scaling factor. Hence, with our model we are able to capture different impacts depending on the direction of private information.

The constant fraction of informed buys or sells on good-news or bad-news trading days are given by

$$\frac{\Lambda_{1,d}^{\mathcal{G},\text{inf}}}{\Lambda_{1,d}^{\mathcal{G},\text{inf}} + \Lambda_{1,d}^{\mathcal{N}}} \quad \text{and} \quad \frac{\Lambda_{-1,d}^{\mathcal{B},\text{inf}}}{\Lambda_{-1,d}^{\mathcal{B},\text{inf}} + \Lambda_{-1,d}^{\mathcal{N}}}, \tag{6.71}$$

respectively.

In the PIN-ALACD model the expressions in equation (6.71) translate to

$$1 - \exp(-\tau) \tag{6.72}$$

for the fraction of informed buys as well as for the fraction of informed sells.

In our dynamic approach they can be written as

$$1 - \exp(-\tau_1 k_1) \quad \text{and} \quad 1 - \exp(-\tau_{-1} k_{-1}) \tag{6.73}$$

for informed buys and informed sells, respectively.[75]

[75] The corresponding fractions in the context of the static EHO model are given by $\frac{\mu}{\mu + \epsilon_b}$ and $\frac{\mu}{\mu + \epsilon_s}$.

The expected total number of trades for trading day d consists of three parts, whereas each is weighted with the according smoothed probability: the expected total number of buys and sells for no-news, good-news and bad-news condition. Therefore, using equation (6.68), we can calculate the expected total number of trades as

$$\begin{aligned}
&\pi_{\mathcal{N},d}\left(\Lambda^{\mathcal{N}}_{1,d} + \Lambda^{\mathcal{N}}_{-1,d}\right) + \pi_{\mathcal{G},d}\left(\Lambda^{\mathcal{G}}_{1,d} + \Lambda^{\mathcal{G}}_{-1,d}\right) + \pi_{\mathcal{B},d}\left(\Lambda^{\mathcal{B}}_{1,d} + \Lambda^{\mathcal{B}}_{-1,d}\right) \\
=&\pi_{\mathcal{N},d}\left(\Lambda^{\mathcal{N}}_{1,d} + \Lambda^{\mathcal{N}}_{-1,d}\right) + \pi_{\mathcal{G},d}\left(\Lambda^{\mathcal{G},\text{inf}}_{1,d} + \Lambda^{\mathcal{N}}_{1,d} + \Lambda^{\mathcal{N}}_{-1,d}\right) + \pi_{\mathcal{B},d}\left(\Lambda^{\mathcal{N}}_{1,d} + \Lambda^{\mathcal{B},\text{inf}}_{-1,d} + \Lambda^{\mathcal{N}}_{-1,d}\right) \\
=&\Lambda^{\mathcal{N}}_{1,d}\left(\pi_{\mathcal{N},d} + \pi_{\mathcal{G},d} + \pi_{\mathcal{B},d}\right) + \Lambda^{\mathcal{N}}_{-1,d}\left(\pi_{\mathcal{N},d} + \pi_{\mathcal{G},d} + \pi_{\mathcal{B},d}\right) + \pi_{\mathcal{G},d}\Lambda^{\mathcal{G},\text{inf}}_{1,d} + \pi_{\mathcal{B},d}\Lambda^{\mathcal{B},\text{inf}}_{-1,d} \\
=&\Lambda^{\mathcal{N}}_{1,d} + \Lambda^{\mathcal{N}}_{-1,d} + \pi_{\mathcal{G},d}\Lambda^{\mathcal{G},\text{inf}}_{1,d} + \pi_{\mathcal{B},d}\Lambda^{\mathcal{B},\text{inf}}_{-1,d}.
\end{aligned} \tag{6.74}$$

The expected number of transactions initiated by informed traders consists of informed buys and sells, again each part is weighted with the corresponding smoothed probability. Hence, according to the general definition of the probability of informed trading as relation of expected number of information-based trades to expected total number of transactions, PIN for trading day d calculates as

$$\text{PIN}_d = \frac{\pi_{\mathcal{G},d}\Lambda^{\mathcal{G},\text{inf}}_{1,d} + \pi_{\mathcal{B},d}\Lambda^{\mathcal{B},\text{inf}}_{-1,d}}{\Lambda^{\mathcal{N}}_{1,d} + \Lambda^{\mathcal{N}}_{-1,d} + \pi_{\mathcal{G},d}\Lambda^{\mathcal{G},\text{inf}}_{1,d} + \pi_{\mathcal{B},d}\Lambda^{\mathcal{B},\text{inf}}_{-1,d}}. \tag{6.75}$$

Equation (6.75) is utilized in the model by Tay, Ting, Tse, and Warachka (2009) to compute PIN_d and of course can also be applied in our PIN-HMM setting. However, we decided to split daily PIN_d estimates up into $\text{PIN}_{\mathcal{G},d}$, the probability of informed trading driven by positive private information, and $\text{PIN}_{\mathcal{B},d}$, the probability of informed trading driven by negative private information, which has not been done in literature before. The PIN estimates for good-news and bad-news trading days can be written as

$$\text{PIN}_{\mathcal{G},d} = \frac{\pi_{\mathcal{G},d}\Lambda^{\mathcal{G},\text{inf}}_{1,d}}{\Lambda^{\mathcal{N}}_{1,d} + \Lambda^{\mathcal{N}}_{-1,d} + \pi_{\mathcal{G},d}\Lambda^{\mathcal{G},\text{inf}}_{1,d} + \pi_{\mathcal{B},d}\Lambda^{\mathcal{B},\text{inf}}_{-1,d}}, \tag{6.76}$$

$$\text{PIN}_{\mathcal{B},d} = \frac{\pi_{\mathcal{B},d}\Lambda^{\mathcal{B},\text{inf}}_{-1,d}}{\Lambda^{\mathcal{N}}_{1,d} + \Lambda^{\mathcal{N}}_{-1,d} + \pi_{\mathcal{G},d}\Lambda^{\mathcal{G},\text{inf}}_{1,d} + \pi_{\mathcal{B},d}\Lambda^{\mathcal{B},\text{inf}}_{-1,d}}. \tag{6.77}$$

Hence, we can write the probability of informed trading on trading day d as

$$\text{PIN}_d = \text{PIN}_{\mathcal{G},d} + \text{PIN}_{\mathcal{B},d}. \tag{6.78}$$

We apply the generalized setting of PIN estimators for the empirical results in chapter 11 to receive comparable results for both presented dynamic models in this thesis.

As mentioned before, estimates of the probabilities of informed trading are not fixed to a daily basis. They can also be computed for intraday intervals of arbitrary length as long as sufficient

data exists in every interval and convergence of the optimization routine is ensured. See section 11.5 for intraday estimates of the probabilities of informed trading computed for trading days around the delisting of GM in June 2009 and the period of extremely high prices of VOW in October 2008.

7 Source Code of Implementations for High-Frequency Models

In contrast to the context of static models for PIN, we have not released a package for the dynamic models on CRAN yet. However, in this chapter we will present and shortly discuss the source code of functions we used for computations in which the PIN-ALACD and PIN-HMM approaches are involved.

More precisely, this chapter deals with the source code of our optimization routines and calculations of the probability of informed trading for the dynamic models presented in this work. The core functions are implemented in the C++ programming language to speed up computations.

Since there are no algorithms to find sets of *good* starting values which ensure a fast convergence of the optimizer to an (ideally global) maximum, we are forced to perform a huge amount of maximization runs with changing starting values and finally pick the set of parameter estimates which yield the highest likelihood function value. Hence evaluations of likelihood functions are a potential bottleneck of optimizations, and therefore we spent a lot of time in tuning especially those functions and we decided to not implement them in pure R language. Likelihood functions of the dynamic models are implemented in the C++ language with the help of the excellent **Rcpp** and **RcppArmadillo** packages by Eddelbuettel and François (2011) and Eddelbuettel and Sanderson (2014), respectively.

Therefore, the actual R functions which are called by the user are small in terms of number of lines and act in most cases as wrappers for the underlying C++ implementations. In addition, it is possible to evaluate the likelihood functions in parallel, which is especially useful for big data.

The main function in the context of optimization routines of dynamic models is the *pin_est* function in code chunk 7.1. It can be seen as the equivalent of the *pin_est* function from the **pinbasic** package. In most cases it is sufficient to only call this function because it can handle the creation of sets of initial values and the actual maximization of the chosen model's likelihood function. It returns information about the optimization results, PIN estimates as well as

the corresponding state probabilities for trading days. We tested several different optimizers in advance, but the *nlminb* function from the **stats** packages delivered the best results for our purposes. We already introduced the optimizer in chapter 5 which is dedicated to the **pinbasic** package. Hence, we exclusively use the *nlminb* function for estimations of PIN.

Similar to the **pinbasic** package, *pin_est* shares many arguments with its underlying functions. The `init_mat` argument can be used to specify sets of initial values and `data` specifies the high-frequency data which should be used in optimizations.[76] Maximum number of iterations for the optimizer and how often the intermediate results should be printed to the console can be controlled by `limit` and `print_level`. The `scaling_iter` argument is explained in the context of the *calc_d* function in code chunk 7.15. Whether the results of all maximization runs should be returned can be specified via `all_res`. If the computations should be performed in parallel, the object of the already existing cluster can be passed to `cluster_name`.[77] The logical `close_con` argument decides whether the specified cluster should be closed after computations are finished. Lower and upper bounds for the optimization can be set via the `lower` and `upper` arguments, respectively. If those two arguments remain unchanged, default lower and upper bounds are specified immediately at the beginning of the *pin_est* function with respect to the chosen dynamic approach, either PIN-ALACD or PIN-HMM.[78] The type of dynamic approach can be chosen via the logical flag `HMM`. The number of optimization runs can be set via `nopts`. However, this argument is only active if `init_mat = NULL`, otherwise it is overridden with the number of rows of the matrix passed to `init_mat`.

Code Chunk 7.1 (Source code of function *pin_est*):

```
function(init_mat = NULL, data = NULL, cluster_name = NULL, HMM = TRUE, limit = 500,
    print_level = 25, scaling_iter = 50, lower = NULL, upper = NULL, nopts = 100,
    all_res = FALSE, close_con = FALSE) {
    if (close_con) {
        if (!is.null(cluster_name))
            on.exit(stopCluster(cluster_name))
    }
    if (is.null(lower)) {
        if (!HMM) {
            lower = c(-5, rep(0, 3), rep(-3, 2), rep(0, 2), -0.2, rep(-3, 2),
                rep(0, 2), -0.2, 0)
        } else {
            lower = c(rep(0, 6), rep(-3, 2), rep(0, 2), -0.2, rep(-3, 2), rep(0,
                2), -0.2, rep(0, 2), rep(1e-10, 2))
        }
```

[76] In chapter 9 the functions are presented and explained which are necessary to prepare raw data so that the *pin_est* function is able to handle it.

[77] See chapter 9 for an explanation of the *cluster_prep* function, which can be harnessed to create a cluster which can then be passed to the *pin_est* function.

[78] We set the default values for lower and upper bounds according to experiences we made in optimizations with the datasets we use in this work and which are explained in chapter 8. Therefore, for other datasources or time ranges, the default vectors for `lower` and `upper` may not be appropriate.

```
}
if (is.null(upper)) {
    if (!HMM) {
        upper = c(rep(10, 4), rep(3, 2), 1, 0.3, 0.2, rep(3, 2), 1, 0.3,
            0.2, 2)
    } else {
        upper = c(rep(1, 6), rep(3, 2), 1, 0.3, 0.2, rep(3, 2), 1, 0.3,
            0.2, rep(2, 4))
    }
}
if (is.null(init_mat)) {
    init_mat <- initial_vals(HMM = HMM, n = nopts, lower = lower, upper = upper)
} else {
    nopts <- nrow(init_mat)
}
coef_mat <- matrix(data = NA, ncol = ncol(init_mat), nrow = nopts)
colnames(coef_mat) <- par_names(HMM = HMM, parameterization_prob_names = FALSE)
if (HMM) {
    fun <- function(par_est) loglik(par = trans_prob_stab(par_est), data = data,
        cluster_name = cluster_name, HMM = TRUE)
} else {
    fun <- function(par_est) loglik(par = par_est, data = data, cluster_name = cluster_name,
        HMM = FALSE)
}
loglik_vals <- conv_codes <- numeric(nopts)
stat_dist_mat <- matrix(data = NA, ncol = 3, nrow = nopts)
colnames(stat_dist_mat) <- c("Bad", "Good", "No")
pin_calc_res <- vector("list", nopts)
state_probs_res <- vector("list", nopts)
pit_res <- vector("list", nopts)
for (i in seq_len(nopts)) {
    if (!(i%%(nopts * 0.1)))
        message(paste0("Starting Run ", i))
    feas_init <- FALSE
    while (!feas_init) {
        tmp_par <- init_mat[i, ]
        steps <- 0
        while (steps < limit) {
            d <- calc_d(tmp_par, h = 1e-04, FUN = fun, lower = lower, upper = upper)
            est_res <- nlminb(start = tmp_par * d, objective = function(x) -fun(x/d),
              lower = lower * d, upper = upper * d, control = list(iter.max = scaling_iter,
                trace = print_level))
            if (is.infinite(est_res$objective))
              steps <- limit
            tmp_par <- est_res$par/d
            if (est_res$convergence)
              steps <- steps + scaling_iter else steps <- limit
        }
```

```
            feas_init <- is.finite(est_res$objective)
            if (!feas_init) {
                message("Optimization has not converged. Generating new set of initial values.")
                init_mat[i, ] <- initial_vals(HMM = HMM, lower = lower, upper = upper,
                  n = 1)[1, ]
            }
        }
        if (HMM) {
            tmp_par <- trans_prob_stab(tmp_par)
            coef_mat[i, ] <- tmp_par
            loglik_vals[i] <- -est_res$objective
            conv_codes[i] <- est_res$convergence
            stat_dist_mat[i, ] <- p_stat(tmp_par, warn = TRUE)
        } else {
            coef_mat[i, ] <- tmp_par
            loglik_vals[i] <- -est_res$objective
            conv_codes[i] <- est_res$convergence
        }
        pin_calc_res[[i]] <- pin_calc(par = tmp_par, data = data, HMM = HMM,
            cluster_name = cluster_name, close_con = FALSE)
        state_probs_res[[i]] <- state_probs(par = tmp_par, data = data, HMM = HMM,
            cluster_name = cluster_name, close_con = FALSE)
        pit_res[[i]] <- pit(par = tmp_par, data = data, HMM = HMM, cluster_name = cluster_name,
            close_con = FALSE)
    }
    if (all_res) {
        order_highest_ll <- order(loglik_vals, decreasing = TRUE)
        if (HMM) {
            res <- list(coef = coef_mat[order_highest_ll, ], loglik = loglik_vals[order_highest_ll],
                conv_code = conv_codes[order_highest_ll], stat_dist = stat_dist_mat[order_highest_ll,
                  ], lower = lower, upper = upper, pin = pin_calc_res[order_highest_ll],
                state_probs = state_probs_res[order_highest_ll], pit = pit_res[order_highest_ll])
        } else {
            res <- list(coef = coef_mat[order_highest_ll, ], loglik = loglik_vals[order_highest_ll],
                conv_code = conv_codes[order_highest_ll], lower = lower, upper = upper,
                pin = pin_calc_res[order_highest_ll], state_probs = state_probs_res[order_highest_ll],
                pit = pit_res[order_highest_ll])
        }
    } else {
        max_ll <- which.max(loglik_vals)
        if (HMM) {
            res <- list(coef = coef_mat[max_ll, ], loglik = loglik_vals[max_ll],
                conv_code = conv_codes[max_ll], stat_dist = stat_dist_mat[max_ll,
                  ], lower = lower, upper = upper, pin = pin_calc_res[[max_ll]],
                state_probs = state_probs_res[[max_ll]], pit = pit_res[[max_ll]])
        } else {
            res <- list(coef = coef_mat[max_ll, ], loglik = loglik_vals[max_ll],
                conv_code = conv_codes[max_ll], lower = lower, upper = upper,
```

```
                pin = pin_calc_res[[max_ll]], state_probs = state_probs_res[[max_ll]],
                pit = pit_res[[max_ll]])
        }
    }
    res
}
```

If `init_mat = NULL`, values of `lower` and `upper` are used in the *initial_vals* function in code chunk 7.2, which returns sets of starting values. They are drawn from uniform distributions whose minimum and maximum depend on the corresponding entries in `lower` and `upper`. It is also ensured that the sums of the parameters α_1, β_1 and α_{-1}, β_{-1} are both smaller than unity. With `n` the total number of sets of initial values is specified.

Code Chunk 7.2 (Source code of function *initial_vals*):

```
function(HMM = TRUE, lower = NULL, upper = NULL, n = 100) {
    if (HMM) {
        if (length(lower) != 20 || length(upper) != 20) {
            stop("Length of 'lower' and/or 'upper' does not match HMM Weibull (20)")
        }
        st <- t(replicate(n, runif(20, min = lower, max = upper)))
    }
    if (!HMM) {
        if (length(lower) != 15 || length(upper) != 15) {
            stop("Length of 'lower' and/or 'upper' does not match 3TW (15)")
        }
        st <- t(replicate(n, runif(15, min = lower, max = upper)))
    }
    colnames(st) <- par_names(HMM = HMM)
    non_stat1 <- which(st[, "alpha_B"] + st[, "beta_B"] >= 1)
    if (length(non_stat1) > 0) {
        st[non_stat1, "beta_B"] <- 1 - st[non_stat1, "alpha_B"] - 1e-06
    }
    non_stat2 <- which(st[, "alpha_S"] + st[, "beta_S"] >= 1)
    if (length(non_stat2) > 0) {
        st[non_stat1, "beta_S"] <- 1 - st[non_stat1, "alpha_S"] - 1e-06
    }
    st
}
```

Column names of the matrix of initial values are set with the help of the *par_names* function whose source code is shown in code chunk 7.3. Names of the parameters involved in the ALACD recursion do not differ by model choice. If the logical argument `HMM` is set to `TRUE`, the `parameterization_prob_names` may influence the nomenclature of the parameters which are responsible for conditions of trading days.

Code Chunk 7.3 (Source code of function *par_names*):

```
function(HMM = TRUE, parameterization_prob_names = TRUE) {
    if (HMM) {
        coef_names <- c("nu_BB", "nu_BS", "alpha_B", "beta_B", "zeta_B", "nu_SB",
            "nu_SS", "alpha_S", "beta_S", "zeta_S", "mu_G", "mu_B", "k_B", "k_S")
        if (!parameterization_prob_names) {
            prob_names <- c("p_BG", "p_BN", "p_GB", "p_GN", "p_NB", "p_NG")
        } else {
            prob_names <- c("p_BB", "p_BG|C", "p_GG", "p_GB|C", "p_NN", "p_NB|C")
        }
    } else {
        coef_names <- c("nu_BB", "nu_BS", "alpha_B", "beta_B", "zeta_B", "nu_SB",
            "nu_SS", "alpha_S", "beta_S", "zeta_S", "mu")
        prob_names <- paste("theta", 1:4, sep = "")
    }
    full_names <- c(prob_names, coef_names)
    full_names
}
```

To stabilize optimizations we decided to reparameterize transition probabilities as follows:

$$a_{\mathcal{BG}} = a_{\mathcal{BG}|C} \cdot (1 - a_{\mathcal{BB}}) \tag{7.1}$$

$$a_{\mathcal{BN}} = (1 - a_{\mathcal{BG}|C}) \cdot (1 - a_{\mathcal{BB}}) \tag{7.2}$$

$$a_{\mathcal{GB}} = a_{\mathcal{GB}|C} \cdot (1 - a_{\mathcal{GG}}) \tag{7.3}$$

$$a_{\mathcal{GN}} = (1 - a_{\mathcal{GB}|C}) \cdot (1 - a_{\mathcal{GG}}) \tag{7.4}$$

$$a_{\mathcal{NB}} = a_{\mathcal{NB}|C} \cdot (1 - a_{\mathcal{NN}}) \tag{7.5}$$

$$a_{\mathcal{NG}} = (1 - a_{\mathcal{NB}|C}) \cdot (1 - a_{\mathcal{NN}}), \tag{7.6}$$

where the indices $**|C$ symbolize a switch in states of trading days given that a change happened, e. g. $a_{\mathcal{BG}|C}$ is the probability that the state of the current trading day is bad-news and the state of the next is good-news given that the condition changes from the current to the next day. Hence, the initial values for the transition probabilities in our optimization routine refer to $a_{\mathcal{BG}|C}$, $a_{\mathcal{BB}}$, $a_{\mathcal{GB}|C}$, $a_{\mathcal{GG}}$, $a_{\mathcal{NB}|C}$ and $a_{\mathcal{NN}}$ if the logical argument `parameterization_prob_names` is active.[79]

[79] See the function *trans_prob_stab* in code chunk 7.14.

The *loglik* function is called by *pin_est* to define the function to be maximized, where the parameter values are passed to the par argument. Its source code is presented in code chunk 7.4.

Code Chunk 7.4 (Source code of function *loglik*):

```
function(par = NULL, data = NULL, cluster_name = NULL, HMM = TRUE) {
    if (is.null(par))
        stop("No parameter vector found")
    if (is.null(names(par)))
        stop("'par' has no names! See function 'par_names'")
    if (any(is.nan(par)))
        return(-Inf)
    if ((par["alpha_B"] + par["beta_B"]) ≥ 1)
        return(-Inf)
    if ((par["alpha_S"] + par["beta_S"]) ≥ 1)
        return(-Inf)
    if (!HMM) {
        if (length(par) ≠ 15)
            stop("Length of 'par' does not match 3TW model")
        llexp3TW(par, data = data, cluster_name = cluster_name)
    } else {
        if ((par["p_BG"] + par["p_BN"]) > 1)
            return(-Inf)
        if ((par["p_GB"] + par["p_GN"]) > 1)
            return(-Inf)
        if ((par["p_NB"] + par["p_NG"]) > 1)
            return(-Inf)
        if (length(par) ≠ 20)
            stop("Length of 'par' does not match HMM Weibull model")
        res <- llwei(par, data = data, cluster_name = cluster_name)
        mat_trans <- trans_mat(par[c("p_BG", "p_BN", "p_GB", "p_GN", "p_NB",
            "p_NG")])
        p_init <- p_stat(par[c("p_BG", "p_BN", "p_GB", "p_GN", "p_NB", "p_NG")])
        fb_ll(res, p_init, mat_trans)
    }
}
```

It acts as wrapper for the *llexp3TW* function in case the PIN-ALACD model is chosen, and calls a bundle of functions for the PIN-HMM model, namely *llwei*, *trans_mat*, *p_stat* and *fb_ll*. For the PIN-ALACD model the *llexp3TW* function in code chunk 7.5 is a wrapper for the C++ code in code chunk 7.6 where the actual computation of the likelihood happens. We need the *llexp3TW_helper* function in code chunk 7.7 to enable computations of the likelihood in parallel. If computations should be performed on a cluster, this function is a wrapper for the underlying C++ code which expects that a list called data_sub with slots Intraday and Daily is available on

each node.[80] The naming of this list containing the required data for optimizations is identical on each node, but the content will vary depending on the specified number of CPU cores.

Code Chunk 7.5 (Source code of function *llexp3TW*):

```
function(par, data = NULL, cluster_name = NULL) {
    if (is.null(cluster_name) || length(cl) == 1) {
        ret <- llexp3TW_cpp(par, data[["IntraDay"]], data[["Daily"]][, "initial_buys"],
            data[["Daily"]][, "initial_sells"], data[["Daily"]][, "log_mean_dura"],
            data[["Daily"]][, "log_mean_sa"], data[["Daily"]][, "tn_helper"],
            data[["Daily"]][, "tg_helper1"], data[["Daily"]][, "tg_helper2"])
        ret
    } else {
        ret <- clusterCall(cluster_name, fun = llexp3TW_helper, par = par)
        do.call(sum, ret)
    }
}
```

Code Chunk 7.6 (Source code of C++ function *llexp3TW_cpp*):

```
double llexp3TW_cpp(const NumericVector & param,
                    const List & DATAMAT,
                    const NumericVector & initbuys, const NumericVector & initsells,
                    const NumericVector & meandura, const NumericVector & meansa,
                    const NumericVector & tnhelper,
                    const NumericVector & tghelper1, const NumericVector & tghelper2){
  int  ni = 0, i = 0, j = 0, ni_full = 0;
  const int n = DATAMAT.size();

  double mue = param[14],
         helper3 = exp(mue), helper3m1 = helper3 - 1.0,
         const_buys_sa = param[6] + param[7], const_buys = const_buys_sa - 1.0,
         const_sells_sa = param[11] + param[12], const_sells = const_sells_sa - 1.0,
         meandiff = 0.0;

  double expsumbuys = 0.0, expsumsells = 0.0, ans1 = 0.0, ans2 = 0.0,
         sumcondbuys = 0.0, sumcondsells = 0.0;

  NumericVector ll_vec(3);
  NumericVector ll_vec_stab(3);
  double ll_max = 0.0;
```

[80] See the source code of the function *data_prep* and *cluster_prep* in chapter 9.

```
NumericVector prob_no_helper = no_init(n);
NumericVector prob_good_helper = no_init(n);
NumericVector log_prob_no = no_init(n);
NumericVector log_prob_event = no_init(n);
NumericVector log_prob_good = no_init(n);
NumericVector log_prob_bad = no_init(n);
NumericVector res(n);

prob_no_helper = exp(param[0] + param[1] * tnhelper);
prob_good_helper = exp(param[2] * tghelper1 - param[3] * tghelper2);

log_prob_no = -log1p(prob_no_helper);
log_prob_good = -log1p(prob_good_helper);
log_prob_event = log(prob_no_helper) + log_prob_no;
log_prob_bad = log(prob_good_helper) + log_prob_good;

for(i = 0; i < n; i++){
  NumericMatrix mat_full = DATAMAT[i];
  ni_full = mat_full.nrow();
  ni = ni_full - 1;

  SubMatrix<REALSXP> mat_sub = mat_full(Range(0, ni_full - 2), _);
  NumericMatrix mat = mat_sub;
  NumericVector ans4 = no_init(ni);
  NumericVector ans5 = no_init(ni);
  LogicalVector td_ind = no_init(ni);
  NumericVector ans4_td = no_init(ni);
  NumericVector ans5_ntd = no_init(ni);
  NumericVector psi1n = no_init(ni);
  NumericVector psim1n = no_init(ni);

  psi1n[0] = initbuys[i];
  psim1n[0] = initsells[i];
  meandiff = meandura[i] - meansa[i];

  for(j = 1; j < ni; j++){
    psi1n[j] = const_buys * meandiff + mat(j,3) - (const_buys_sa * mat(j-1,3)) +
      param[4] * mat(j-1,1) + param[5] * mat(j-1,2) +  param[6] * psi1n[j-1] +
      param[7] * mat(j-1,0) + param[8] * mat(j-1,5);

    psim1n[j] = const_sells * meandiff + mat(j,3) - (const_sells_sa * mat(j-1,3)) +
      param[9] * mat(j-1,1) + param[10] * mat(j-1,2) + param[11] * psim1n[j-1] +
      param[12] * mat(j-1,0) + param[13] * mat(j-1,5);
```

```
    }

    ans4 = mat(_,0) - psi1n;
    ans5 = mat(_,0) - psim1n;

    td_ind = mat(_,2) == 1;

    ans4_td = ans4[td_ind];
    ans5_ntd = ans5[!td_ind];

    expsumbuys = sum(exp(ans4));
    expsumsells = sum(exp(ans5));
    sumcondbuys = sum(ans4_td);
    sumcondsells = sum(ans5_ntd);

    ans1 = mue * sum(td_ind) - helper3m1 * expsumbuys;
    ans2 = mue * sum(!td_ind) - helper3m1 * expsumsells;

    ll_vec(2) = sumcondbuys + sumcondsells - sum(mat(_,0)) - expsumsells - expsumbuys; //no-news
    ll_vec(1) = ll_vec(2) + ans1; //good-news
    ll_vec(0) = ll_vec(2) + ans2; //bad-news

    ll_max = max(ll_vec);
    ll_vec_stab = ll_vec - ll_max;

    res(i) = ll_max +
             log(exp(log_prob_no(i) + ll_vec_stab(2)) +
                 exp(log_prob_event(i) + log_prob_good(i) + ll_vec_stab(1)) +
                 exp(log_prob_event(i) + log_prob_bad(i) + ll_vec_stab(0)));

  }
  return sum(res);
}
```

Code Chunk 7.7 (Source code of function *llexp3TW_helper*):

```
function(par) {
    llexp3TW_cpp(par, data_sub[["IntraDay"]], data_sub[["Daily"]][, "initial_buys"],
        data_sub[["Daily"]][, "initial_sells"], data_sub[["Daily"]][, "log_mean_dura"],
        data_sub[["Daily"]][, "log_mean_sa"], data_sub[["Daily"]][, "tn_helper"],
        data_sub[["Daily"]][, "tg_helper1"], data_sub[["Daily"]][, "tg_helper2"])
}
```

The structure for the functions involved in evaluations of the likelihood of the PIN-HMM model is very similar to the PIN-ALACD case by calling the *llwei*, *llwei_cpp* and *llwei_helper* functions. The corresponding source codes can be found in code chunks 7.8, 7.9 and 7.10.

Code Chunk 7.8 (Source code of function *llwei*):

```
function(par = NULL, data = NULL, cluster_name = NULL) {
    if (is.null(cluster_name) || length(cl) == 1) {
        ret <- llwei_cpp(par, data[["IntraDay"]], data[["Daily"]][, "initial_buys"],
            data[["Daily"]][, "initial_sells"], data[["Daily"]][, "log_mean_dura"],
            data[["Daily"]][, "log_mean_sa"])
        ret
    } else {
        ret <- clusterCall(cluster_name, fun = llwei_helper, par = par)
        do.call(rbind, ret)
    }
}
```

Code Chunk 7.9 (Source code of C++ function *llwei_cpp*):

```
NumericMatrix llwei_cpp(const NumericVector & param,
                        const List & DATAMAT,
                        const NumericVector & initbuys, const NumericVector & initsells,
                        const NumericVector & meandura, const NumericVector & meansa) {
  int  ni = 0, i, j = 0, ni_full = 0;
  int n = DATAMAT.size();

  double mue_buys = param[16], mue_sells = param[17], bbuys=param[18], bsells=param[19],
  lbbuys = log(bbuys),lbsells = log(bsells),
  helper1 = bbuys * mue_buys, helper2 = bsells * mue_sells,
  helper3 = exp(helper1), helper4 = exp(helper2),
  helper3m1 = helper3 - 1.0, helper4m1 = helper4 - 1.0,
  const_buys_sa = param[8] + param[9], const_buys = const_buys_sa - 1.0,
  const_sells_sa = param[13] + param[14],
  const_sells = const_sells_sa - 1.0, meandiff = 0.0,
  psi_buys_adjust = lgamma(1.0 + 1.0/bbuys),
  psi_sells_adjust = lgamma(1.0 + 1.0/bsells);

  double expsumbuys = 0.0, expsumsells = 0.0, ans1 = 0.0, ans2 = 0.0,
  sumcondbuys = 0.0, sumcondsells = 0.0,
  last_psi1n = 0.0, last_psim1n = 0.0,
  last_no = 0.0, last_good = 0.0, last_bad = 0.0;
```

```
NumericMatrix ll_vec(n, 3);

for(i = 0; i < n; i++){
  NumericMatrix mat_full = DATAMAT[i];
  ni_full = mat_full.nrow();

  SubMatrix<REALSXP> mat_sub = mat_full(Range(0, ni_full - 2), _);
  NumericMatrix mat = mat_sub;

  ni = ni_full - 1;

  NumericVector ans4 = no_init(ni);
  NumericVector ans5 = no_init(ni);
  LogicalVector td_ind = no_init(ni);
  NumericVector ans4_td = no_init(ni);
  NumericVector ans5_ntd = no_init(ni);
  NumericVector psi1n = no_init(ni);
  NumericVector psim1n = no_init(ni);

  psi1n[0] = initbuys[i];
  psim1n[0] = initsells[i];

  meandiff = meandura[i] - meansa[i];

  for(j = 1; j < ni; j++){
    psi1n[j] = const_buys * meandiff + mat(j,3) - (const_buys_sa * mat(j-1,3)) +
      param[6] * mat(j-1,1) + param[7] * mat(j-1,2) +  param[8] * psi1n[j-1] +
      param[9] * mat(j-1,0) + param[10] * mat(j-1,5) - psi_buys_adjust;

    psim1n[j] = const_sells * meandiff + mat(j,3) - (const_sells_sa * mat(j-1,3)) +
      param[11] * mat(j-1,1) + param[12] * mat(j-1,2) + param[13] * psim1n[j-1] +
      param[14] * mat(j-1,0) + param[15] * mat(j-1,5) - psi_sells_adjust;
  }

  ans4 = mat(_,0) - psi1n;
  ans5 = mat(_,0) - psim1n;

  td_ind = mat(_,1) == 1;

  ans4_td = ans4[td_ind];
  ans5_ntd = ans5[!td_ind];

  expsumbuys = sum(exp(bbuys * ans4));
```

```
    expsumsells = sum(exp(bsells * ans5));

    sumcondbuys = sum(ans4_td);
    sumcondsells = sum(ans5_ntd);

    if(mat_full(ni, 0) == 0) {
      last_psi1n = const_buys * meandiff + mat_full(ni,3) - (const_buys_sa * mat_full(ni-1,3)) +
        param[6] * mat_full(ni-1,1) + param[7] * mat_full(ni-1,2) +  param[8] * psi1n[ni-1] +
        param[9] * mat_full(ni-1,0) + param[10] * mat_full(ni-1,5) - psi_buys_adjust;

      last_psim1n = const_sells * meandiff + mat_full(ni,3) - (const_sells_sa * mat_full(ni-1,3)) +
        param[11] * mat_full(ni-1,1) + param[12] * mat_full(ni-1,2) + param[13] * psim1n[ni-1] +
        param[14] * mat_full(ni-1,0) + param[15] * mat_full(ni-1,5) - psi_sells_adjust;

      last_no = -exp(bbuys * (mat_full(ni,4) - last_psi1n)) -
        exp(bsells * (mat_full(ni,4) - last_psim1n));
      last_good = -exp(bbuys * (mat_full(ni,4) - last_psi1n - mue_buys)) -
        exp(bsells * (mat_full(ni,4) - last_psim1n));
      last_bad = -exp(bbuys * (mat_full(ni,4) - last_psi1n)) -
        exp(bsells * (mat_full(ni,4) - last_psim1n - mue_sells));
    }

    ans1 = helper1 * sum(td_ind) - helper3m1 * expsumbuys + last_good - last_no;
    ans2 = helper2 * sum(!td_ind) - helper4m1 * expsumsells + last_bad - last_no;

    ll_vec(i, 2) = lbbuys * sum(td_ind) + lbsells * sum(!td_ind) + bbuys * sumcondbuys +
      bsells * sumcondsells - sum(mat(_,0)) - expsumsells -
      expsumbuys + last_no; //no-news
    ll_vec(i, 1) = ll_vec(i, 2) + ans1; //good-news
    ll_vec(i, 0) = ll_vec(i, 2) + ans2; //bad-news
  }
  colnames(ll_vec) = CharacterVector::create("Bad", "Good", "No");
  return ll_vec;
}
```

Code Chunk 7.10 (Source code of function *llwei_helper*):

```
function(par) {
    llwei_cpp(par, data_sub[["IntraDay"]], data_sub[["Daily"]][, "initial_buys"],
        data_sub[["Daily"]][, "initial_sells"], data_sub[["Daily"]][, "log_mean_dura"],
        data_sub[["Daily"]][, "log_mean_sa"])
}
```

The *llwei* function returns values of emission densities for each unit time interval and all potential conditions of trading days.[81] These results are then processed by the forward-backward algorithm implemented with the C++ function *fb_ll* in code chunk 7.11.[82]

Code Chunk 7.11 (Source code of C++ function *fb_ll*):

```
double fb_ll(arma::mat daily_ll, arma::rowvec pInit, arma::mat transmat) {
  int nrow = daily_ll.n_rows, ncol = daily_ll.n_cols, i = 0;

  arma::colvec ll_max = arma::max(daily_ll, 1);
  arma::mat ll_max_mat(nrow, ncol);
  arma::mat ll_stab(nrow, ncol);
  arma::mat logalpha(nrow, ncol);
  arma::rowvec alpha(3);

  ll_max_mat.col(0) = ll_max;
  ll_max_mat.col(1) = ll_max;
  ll_max_mat.col(2) = ll_max;

  ll_stab = daily_ll - ll_max_mat;

  logalpha.row(0) = log(pInit) + ll_stab.row(0);

  for(i = 1; i < nrow; i++) {
      alpha = exp(logalpha.row(i-1));
      logalpha.row(i) = ll_stab.row(i) + log(alpha * transmat);
  }
  double K = max(logalpha.row(nrow-1));
  return sum(ll_max) + K + log(sum(exp(logalpha.row(nrow-1) - K)));
}
```

To apply the forward-backward algorithm to the results from the *llwei* function, results for the transition matrix as well as the stationary distribution of conditions of trading days must be available which are then passed to the `pInit` and `transmat` arguments of *fb_ll*. Transition matrices and stationary distributions can be computed with the `trans_mat` and `p_stat` functions, respectively. The `trans_mat` function has the additional logical argument `hessian` which initiates further checks and computations if one intends to compute Hessian matrices. The `p_stat` functions prints a warning to the console if `warn` is set to true and the computation of the stationary distribution of states is not reasonable. Moreover, each element in the vector of the stationary

[81] For the definition of emission densities see equation (6.21).

[82] While most other C++ functions rely only on the **Rcpp** package, *fb_ll* makes use of the **RcppArmadillo** package. An introduction to the **RcppArmadillo** package and its syntax is given in Eddelbuettel and Sanderson (2014).

distribution of states is then set to $\frac{1}{3}$. The source code for both functions is given in code chunks 7.12 and 7.13, respectively.

Code Chunk 7.12 (Source code of function *trans_mat*):

```
function(par, hessian = FALSE) {
    mat_trans <- matrix(c(1 - (par["p_BG"] + par["p_BN"]), par["p_BG"], par["p_BN"],
        par["p_GB"], 1 - (par["p_GB"] + par["p_GN"]), par["p_GN"], par["p_NB"], par["p_NG"],
        1 - (par["p_NB"] + par["p_NG"])), nrow = 3, byrow = TRUE)
    if (hessian) {
        neg_entries <- apply(mat_trans, MARGIN = 1, FUN = function(x) which(x < 0))
        for (i in seq_along(neg_entries)) {
            if (length(neg_entries[[i]]) > 0) {
                mat_trans[i, neg_entries[[i]]] <- 0
                mat_trans[i, -neg_entries[[i]]] <- mat_trans[i, -neg_entries[[i]]]/sum(mat_trans[i,
                  -neg_entries[[i]]])
            }
        }
    }
    colnames(mat_trans) <- rownames(mat_trans) <- c("Bad", "Good", "No")
    mat_trans
}
```

Code Chunk 7.13 (Source code of function *p_stat*):

```
function(par, warn = FALSE) {
    if (any(is.nan(par))) {
        erg <- rep(1/3, 3)
        names(erg) <- c("Bad", "Good", "No")
        return(erg)
    }
    erg <- numeric(3)
    names(erg) <- c("Bad", "Good", "No")
    erg["Bad"] <- par["p_NB"] * (par["p_GB"] + par["p_GN"]) + par["p_NG"] *
        par["p_GB"]
    erg["Good"] <- par["p_BG"] * (par["p_NB"] + par["p_NG"]) + par["p_BN"] *
        par["p_NG"]
    erg["No"] <- par["p_GN"] * (par["p_BG"] + par["p_BN"]) + par["p_GB"] * par["p_BN"]
    summe <- sum(erg)
    if (summe > 0) {
        erg/summe
    } else {
        if (warn) {
            warning("No unique stationary distribution!")
        }
```

```
        erg <- rep(1/3, 3)
        erg
    }
}
```

For the PIN-HMM model one further function is involved in the specification of the function to be maximized, namely *trans_prob_stab* in code chunk 7.14, which transforms reparameterized transition probabilities back into the form shown by the results of the empirical applications in section 11.1 and 11.2.

Code Chunk 7.14 (Source code of function *trans_prob_stab*):

```
function(par) {
    if (all(names(par)[1:6] %in% c("p_BG", "p_BN", "p_GB", "p_GN", "p_NB", "p_NG"))) {
        return(par)
    } else {
        par_trans <- numeric(6)
        names(par_trans) <- c("p_BG", "p_BN", "p_GB", "p_GN", "p_NB", "p_NG")
        par_trans["p_BG"] <- par["p_BG|C"] * (1 - par["p_BB"])
        par_trans["p_BN"] <- (1 - par["p_BG|C"]) * (1 - par["p_BB"])
        par_trans["p_GB"] <- par["p_GB|C"] * (1 - par["p_GG"])
        par_trans["p_GN"] <- (1 - par["p_GB|C"]) * (1 - par["p_GG"])
        par_trans["p_NB"] <- par["p_NB|C"] * (1 - par["p_NN"])
        par_trans["p_NG"] <- (1 - par["p_NB|C"]) * (1 - par["p_NN"])
        return(c(par_trans, par[!(names(par) %in% c("p_BB", "p_BG|C", "p_GG",
            "p_GB|C", "p_NN", "p_NB|C"))]))
    }
}
```

For optimizations we use the *nlmimb* function from the **stats** package, as we already did in the context of static models. To increase the stability of maximizations and improve convergence, we scale parameters with the help of the *calc_d* function presented in code chunk 7.15.

Code Chunk 7.15 (Source code of function *calc_d*):

```
function(par = NULL, h = 1e-04, FUN = NULL, lower = rep(-Inf, length(par)),
    upper = rep(Inf, length(par))) {
    fun <- FUN
    npar <- length(par)
    low.hit <- (lower > (par - h))
    upper.hit <- (upper < (par + h))
    bound.hit <- low.hit | upper.hit
    gr <- matrix(0, nrow = npar, ncol = 3)
    loglik <- fun(par)
```

```
    for (i in seq_len(npar)) {
        if (!bound.hit[i]) {
            gr[i, 1] <- fun(par - h * (i == 1:npar))
            gr[i, 2] <- fun(par + h * (i == 1:npar))
        }
    }
    gr[, 1] <- (loglik - gr[, 1])/h
    gr[, 2] <- (gr[, 2] - loglik)/h
    res <- sqrt(abs(gr[, 2] - gr[, 1])/h)
    res[res == 0] <- 1
    res_na <- is.na(res)
    res[res_na] <- 1
    return(res)
}
```

This function returns approximations of the square root of diagonal elements of the Hessian matrix of a function `FUN` at each parameter value `par` by using second symmetric derivatives[83]. The value of argument `h` is used to check which estimates hit their lower or upper bound or are too close to them and as step size in approximations. These approximations are then used to scale the parameters similar to the sensitivity-based scaling method proposed by Yang and Lee (2010). Parameters for which the *calc_d* function would have returned a scaling factor of either 0 or `NA` are assigned a scaling factor of 1 so that actually those parameters are excluded from scaling. The intention to use this scaling method is to reduce the condition number of the Hessian matrix. The condition number of symmetric positive definite matrices is given be the ratio of the largest to the smallest eigenvalue (see Quarteroni, Sacco, and Saleri 2007, p. 61). Ill-conditioned matrices with high condition numbers are almost singular and therefore calculations of their inverses, if they are invertible at all, lead to large numerical errors which also influences the convergence rate of the optimizer. Ideally, diagonal elements of the Hessian matrix equal unity while off-diagonal elements are smaller than unity. According to Yang and Lee (2010), condition numbers of such scaled matrices are typically lower than that of the original ones. How often the scaling is updated can be controlled with the `scaling_iter` argument of the *pin_est* function.[84] We set this argument to 50 to receive the parameter estimates presented in the sections 11.1 and 11.2.

After each optimization run we compute estimates of the probability of informed trading as well as the state probabilities of trading days according to the chosen dynamic model. The latter are calculated with the help of the *state_probs* function whose source code can be found in code chunk 7.16.

Code Chunk 7.16 (Source code of function *state_probs*):

```
function(par = NULL, data = NULL, cluster_name = NULL, HMM = TRUE, close_con = FALSE) {
```

[83] Explanations about the second symmetric derivative can be found in Zygmund and Fefferman (2003, p. 22).
[84] See code chunk 7.1.

```
    if (close_con) {
        if (!is.null(cluster_name))
            on.exit(stopCluster(cluster_name))
    }
    if (is.null(par))
        stop("No parameter vector found")
    if (!HMM) {
        if (length(par) != 15)
            stop("Length of 'par' does not match 3TW model")
        probs3TW(par, data = data, cluster_name = cluster_name)
    } else {
        if (length(par) != 20)
            stop("Length of 'par' does not match HMM Weibull model")
        res <- llwei(par, data = data, cluster_name = cluster_name)
        mat_trans <- trans_mat(par[c("p_BG", "p_BN", "p_GB", "p_GN", "p_NB",
            "p_NG")])
        p_init <- p_stat(par[c("p_BG", "p_BN", "p_GB", "p_GN", "p_NB", "p_NG")])
        probs <- fb_smooth(res, p_init, mat_trans)
        colnames(probs) <- c("Bad", "Good", "No")
        probs
    }
}
```

To calculate the state probabilities of trading days in the PIN-ALACD model we use the *probs3TW* function from code chunk 7.17.

Code Chunk 7.17 (Source code of function *probs3TW*):

```
function(par, data = NULL, cluster_name = NULL) {
    if (is.null(cluster_name) || length(cl) == 1) {
        ret <- state_probs_3TW(par, data[["Daily"]][, "tn_helper"], data[["Daily"]][,
            "tg_helper1"], data[["Daily"]][, "tg_helper2"])
        ret
    } else {
        ret <- clusterCall(cluster_name, fun = probs3TW_helper, para = par)
        do.call(rbind, ret)
    }
}
```

This function depends on the C++ function *state_probs_3TW* and *probs3TW_helper* which is utilized to execute function calls in parallel where each worker expects that a sublist `data_sub` of the whole dataset is available. The source code of both functions can be read from code chunks 7.18 and 7.19.

Code Chunk 7.18 (Source code of C++ function *state_probs_3TW*):

```
NumericMatrix state_probs_3TW(const NumericVector & param,
                              const NumericVector & tnhelper,
                              const NumericVector & tghelper1, const NumericVector & tghelper2) {
  int i = 0, n = tnhelper.size();

  NumericVector prob_no_helper = no_init(n);
  NumericVector prob_good_helper = no_init(n);
  NumericVector log_prob_no = no_init(n);
  NumericVector log_prob_event = no_init(n);
  NumericVector log_prob_good = no_init(n);
  NumericVector log_prob_bad = no_init(n);
  NumericMatrix res(n,3);

  prob_no_helper = exp(param[0] + param[1] * tnhelper);
  prob_good_helper = exp(param[2] * tghelper1 - param[3] * tghelper2);

  log_prob_no = -log1p(prob_no_helper);
  log_prob_good = -log1p(prob_good_helper);
  log_prob_event = log(prob_no_helper) + log_prob_no;
  log_prob_bad = log(prob_good_helper) + log_prob_good;

  for(i = 0; i < n; i++) {
    res(i,2) = exp(log_prob_no(i));
    res(i,1) = exp(log_prob_event(i) + log_prob_good(i));
    res(i,0) = exp(log_prob_event(i) + log_prob_bad(i));
  }
  colnames(res) = CharacterVector::create("Bad", "Good", "No");
  return res;
}
```

Code Chunk 7.19 (Source code of function *probs3TW_helper*):

```
function(para) {
    state_probs_3TW(para, data_sub[["Daily"]][, "tn_helper"], data_sub[["Daily"]][,
        "tg_helper1"], data_sub[["Daily"]][, "tg_helper2"])
}
```

In the PIN-HMM case, most involved function were already mentioned before, except the *fb_smooth*[85] function in code chunk 7.20 which calculates smoothed probabilities according

[85] Similar to the *fb_ll* function, *fb_smooth* is using functionality from the **RcppArmadillo** package.

to equation (6.57) in section 6.2.2. This function expects an object returned by the *llwei* function as its first argument.

Code Chunk 7.20 (Source code of C++ function *fb_smooth*):

```
arma::mat fb_smooth(arma::mat daily_ll, arma::rowvec pInit, arma::mat transmat) {
  int nrow = daily_ll.n_rows, ncol = daily_ll.n_cols, i = 0;

  arma::colvec ll_max = arma::max(daily_ll, 1);
  arma::rowvec tmp(3);
  arma::mat ll_max_mat(nrow, ncol);
  arma::mat ll_stab(nrow, ncol);
  arma::mat logalpha(nrow, ncol);
  arma::mat logbeta(nrow, ncol, arma::fill::zeros);

  arma::rowvec alpha(3);
  arma::mat res(nrow, ncol);

  ll_max_mat.col(0) = ll_max;
  ll_max_mat.col(1) = ll_max;
  ll_max_mat.col(2) = ll_max;

  ll_stab = daily_ll - ll_max_mat;

  logalpha.row(0) = log(pInit) + ll_stab.row(0);

  for(i = 1; i < nrow; i++) {
    alpha = exp(logalpha.row(i-1));
    tmp = exp(ll_stab.row(nrow-i) + logbeta.row(nrow-i));
    logalpha.row(i) = ll_stab.row(i) + log(alpha * transmat);
    logbeta.row(nrow-i-1) = log(transmat * tmp.t()).t();
  }
  double K = max(logalpha.row(nrow-1));
  double loglik =  K + log(sum(exp(logalpha.row(nrow-1) - K)));
  res = logalpha + logbeta - loglik;
  return exp(res);
}
```

The probability of informed trading for every unit time interval is calculated with the *pin_calc* function in code chunk 7.21. It depends on the just explained *state_probs* function as well as the *cum_intens* function to calculate cumulative intensities as described in section 6.3. The source code of the *cum_intens* function is shown in code chunk 7.22. The *cum_intens* function relies on the C++ function *cum_intens_cpp* and the *cum_intens_helper* function. The latter

exhibits the already known structure for helper function for function evaluations in parallel. Corresponding source codes are given in code chunks 7.23 and 7.24, respectively.

Code Chunk 7.21 (Source code of function *pin_calc*):

```
function(par = NULL, data = NULL, HMM = TRUE, cluster_name = NULL, close_con = FALSE) {
    if (close_con) {
        if (!is.null(cluster_name))
            on.exit(stopCluster(cluster_name))
    }
    if (is.null(par))
        stop("'par' is NULL")
    state_probs_mat <- state_probs(par = par, data = data, cluster_name = cluster_name,
        HMM = HMM)
    cum_intens_mat <- cum_intens(par = par, data = data, cluster_name = cl,
        HMM = HMM)
    informed_trading_good <- state_probs_mat[, "Good"] * (cum_intens_mat[, "cond_intens_goodnews_buys"] -
        cum_intens_mat[, "cond_intens_nonews_buys"])
    informed_trading_bad <- state_probs_mat[, "Bad"] * (cum_intens_mat[, "cond_intens_badnews_sells"] -
        cum_intens_mat[, "cond_intens_nonews_sells"])
    informed_trading <- informed_trading_good + informed_trading_bad
    noise_trading <- cum_intens_mat[, "cond_intens_nonews_buys"] + cum_intens_mat[,
        "cond_intens_nonews_sells"]
    total_trading <- noise_trading + informed_trading
    pin <- informed_trading/total_trading
    pin_good <- informed_trading_good/total_trading
    pin_bad <- informed_trading_bad/total_trading
    res <- cbind(pin, pin_good, pin_bad)
    colnames(res) <- c("pin", "pin_good", "pin_bad")
    res
}
```

Code Chunk 7.22 (Source code of function *cum_intens*):

```
function(par = NULL, data = NULL, cluster_name = NULL, HMM = TRUE, close_con = FALSE) {
    if (close_con) {
        if (!is.null(cluster_name))
            on.exit(stopCluster(cluster_name))
    }
    if (is.null(cluster_name) || length(cl) == 1) {
        ret <- cum_intens_cpp(par, data[["IntraDay"]], data[["Daily"]][, "initial_buys"],
            data[["Daily"]][, "initial_sells"], data[["Daily"]][, "log_mean_dura"],
            data[["Daily"]][, "log_mean_sa"], HMM)
        ret
    } else {
        ret <- clusterCall(cluster_name, fun = cum_intens_helper, para = par,
```

```
            HMM = HMM)
        do.call(rbind, ret)
    }
}
```

Code Chunk 7.23 (Source code of C++ function *cum_intens_cpp*):

```
NumericMatrix cum_intens_cpp(const NumericVector & param,
                    const List & DATAMAT,
                    const NumericVector & initbuys, const NumericVector & initsells,
                    const NumericVector & meandura, const NumericVector & meansa,
                    bool HMM) {
  int  ni = 0, i, j = 0, ni_full = 0;
  int n = DATAMAT.size();

  double const_buys_sa = param["alpha_B"] + param["beta_B"],
         const_buys = const_buys_sa - 1.0,
         const_sells_sa = param["alpha_S"] + param["beta_S"],
         const_sells = const_sells_sa - 1.0, meandiff = 0.0,
         last_non_observed_dura = 0.0, inf_corr = 0.0;

  NumericMatrix res(n,4);

  for(i = 0; i < n; i++){
    NumericMatrix mat_full = DATAMAT[i];
    ni = mat_full.nrow();

    SubMatrix<REALSXP> mat_sub = mat_full(Range(0, ni - 1), _);
    NumericMatrix mat = mat_sub;

    NumericVector dura = mat(_,0);
    NumericVector psi1n = no_init(ni);
    NumericVector psim1n = no_init(ni);
    NumericVector psi1g = no_init(ni);
    NumericVector psim1b = no_init(ni);

    NumericVector psi1n_helper = no_init(ni);
    NumericVector psim1n_helper = no_init(ni);
    NumericVector psi1g_helper = no_init(ni);
    NumericVector psim1b_helper = no_init(ni);

    NumericVector psi1n_intens = no_init(ni);
```

```
NumericVector psim1n_intens = no_init(ni);
NumericVector psi1g_intens = no_init(ni);
NumericVector psim1b_intens = no_init(ni);

psi1n[0] = initbuys[i];
psim1n[0] = initsells[i];

meandiff = meandura[i] - meansa[i];

if(HMM) {
  double bbuys=param[18], bsells=param[19],
  psi_buys_adjust = lgamma(1.0 + 1.0/bbuys),
  psi_sells_adjust = lgamma(1.0 + 1.0/bsells);

  for(j = 1; j < ni; j++){
    psi1n[j] = const_buys * meandiff + mat(j,3) - (const_buys_sa * mat(j-1,3)) +
      param[6] * mat(j-1,1) + param[7] * mat(j-1,2) +  param[8] * psi1n[j-1] +
      param[9] * mat(j-1,0) + param[10] * mat(j-1,5) - psi_buys_adjust;

    psim1n[j] = const_sells * meandiff + mat(j,3) - (const_sells_sa * mat(j-1,3)) +
      param[11] * mat(j-1,1) + param[12] * mat(j-1,2) + param[13] * psim1n[j-1] +
      param[14] * mat(j-1,0) + param[15] * mat(j-1,5) - psi_sells_adjust;
  }
} else {
  for(j = 1; j < ni; j++){
    psi1n[j] = const_buys * meandiff + mat(j,3) - (const_buys_sa * mat(j-1,3)) +
      param[4] * mat(j-1,1) + param[5] * mat(j-1,2) +  param[6] * psi1n[j-1] +
      param[7] * mat(j-1,0) + param[8] * mat(j-1,5);

    psim1n[j] = const_sells * meandiff + mat(j,3) - (const_sells_sa * mat(j-1,3)) +
      param[9] * mat(j-1,1) + param[10] * mat(j-1,2) + param[11] * psim1n[j-1] +
      param[12] * mat(j-1,0) + param[13] * mat(j-1,5);
  }
}

if(HMM) {
  psi1n_intens = exp(param[18] * (dura - psi1n));
  psim1n_intens = exp(param[19] * (dura - psim1n));

  res(i,0) = sum(psi1n_intens);
  res(i,1) = sum(psim1n_intens);
  res(i,2) = exp(param[16] * param[18]) * res(i,0);
  res(i,3) = exp(param[17] * param[19]) * res(i,1);
```

```
    } else {
      psi1n_intens = exp(dura - psi1n);
      psim1n_intens = exp(dura - psim1n);

      inf_corr = exp(param[14]);

      res(i,0) = sum(psi1n_intens);
      res(i,1) = sum(psim1n_intens);
      res(i,2) = inf_corr * res(i,0);
      res(i,3) = inf_corr * res(i,1);
    }
  }
  colnames(res) = CharacterVector::create("cond_intens_nonews_buys", "cond_intens_nonews_sells",
                                          "cond_intens_goodnews_buys", "cond_intens_badnews_sells");
  return res;
}
```

Code Chunk 7.24 (Source code of function *cum_intens_helper*):

```
function(para, HMM = TRUE) {
    cum_intens_cpp(para, data_sub[["IntraDay"]], data_sub[["Daily"]][, "initial_buys"],
        data_sub[["Daily"]][, "initial_sells"], data_sub[["Daily"]][, "log_mean_dura"],
        data_sub[["Daily"]][, "log_mean_sa"], HMM)
}
```

Probability integral transforms, as described in section 11.4, are returned by the *pit* function with the source code of the function shown in code chunk 7.25. The cdf of durations, shown in equation (11.6), is implemented with the *dura_cdf* function which calls the C++ function *dura_cdf_cpp* and the helper function *dura_cdf_helper*. Source codes of the three functions are presented in the code chunks 7.26, 7.27 and 7.28.

Code Chunk 7.25 (Source code of function *pit*):

```
function(par = NULL, data = NULL, HMM = TRUE, cluster_name = NULL, close_con = FALSE) {
    if (close_con) {
        if (!is.null(cluster_name))
            on.exit(stopCluster(cluster_name))
    }
    if (is.null(par))
        stop("'par' is NULL")
    state_probs_mat <- state_probs(par = par, data = data, cluster_name = cluster_name,
        HMM = HMM)
```

```
    dura_cdf_list <- dura_cdf(par = par, cluster_name = cl, data = data, HMM = HMM)
    res <- vector("list", length(dura_cdf_list))
    for (i in seq_along(dura_cdf_list)) {
        res[[i]] <- state_probs_mat[i, "No"] * dura_cdf_list[[i]][, "No"] +
            state_probs_mat[i, "Good"] * dura_cdf_list[[i]][, "Good"] + state_probs_mat[i,
            "Bad"] * dura_cdf_list[[i]][, "Bad"]
    }
    res <- do.call(c, res)
    res
}
```

Code Chunk 7.26 (Source code of function *dura_cdf*):

```
function(par = NULL, data = NULL, cluster_name = NULL, HMM = TRUE) {
    if (is.null(cluster_name) || length(cl) == 1) {
        ret <- dura_cdf_cpp(par, data[["IntraDay"]], data[["Daily"]][, "initial_buys"],
            data[["Daily"]][, "initial_sells"], data[["Daily"]][, "log_mean_dura"],
            data[["Daily"]][, "log_mean_sa"], HMM)
        ret
    } else {
        ret <- clusterCall(cluster_name, fun = dura_cdf_helper, para = par,
            HMM = HMM)
        do.call(c, ret)
    }
}
```

The two functions, *cum_intens* and *dura_cdf*, share its structure. The actual core function is implemented in the C++ language due to performance reasons and then called by a wrapper written in R. Moreover, in both cases helper functions are utilized to allow for executions in parallel.

Code Chunk 7.27 (Source code of C++ function *dura_cdf_cpp*):

```
List dura_cdf_cpp(const NumericVector & param,
                  const List & DATAMAT,
                  const NumericVector & initbuys, const NumericVector & initsells,
                  const NumericVector & meandura, const NumericVector & meansa,
                  bool HMM) {
  int  ni = 0, i, j = 0, ni_full = 0;
  int n = DATAMAT.size();

  double const_buys_sa = param["alpha_B"] + param["beta_B"],
  const_buys = const_buys_sa - 1.0,
```

```
const_sells_sa = param["alpha_S"] + param["beta_S"],
const_sells = const_sells_sa - 1.0, meandiff = 0.0;

List res(n);

for(i = 0; i < n; i++){
  NumericMatrix mat_full = DATAMAT[i];
  ni_full = mat_full.nrow();

  SubMatrix<REALSXP> mat_sub = mat_full(Range(0, ni_full - 2), _);
  NumericMatrix mat = mat_sub;

  ni = ni_full - 1;

  NumericMatrix tmp_res(ni,3);

  NumericVector dura = mat(_,0);
  NumericVector rt = mat(_,4);
  NumericVector psi1n = no_init(ni);
  NumericVector psim1n = no_init(ni);
  NumericVector psi1g = no_init(ni);
  NumericVector psim1b = no_init(ni);

  NumericVector psi1n_helper = no_init(ni);
  NumericVector psim1n_helper = no_init(ni);
  NumericVector psi1g_helper = no_init(ni);
  NumericVector psim1b_helper = no_init(ni);

  psi1n[0] = initbuys[i];
  psim1n[0] = initsells[i];

  meandiff = meandura[i] - meansa[i];

  if(HMM) {
    double bbuys=param[18], bsells=param[19],
    psi_buys_adjust = lgamma(1.0 + 1.0/bbuys),
    psi_sells_adjust = lgamma(1.0 + 1.0/bsells);

    for(j = 1; j < ni; j++){
      psi1n[j] = const_buys * meandiff + mat(j,3) - (const_buys_sa * mat(j-1,3)) +
        param[6] * mat(j-1,1) + param[7] * mat(j-1,2) +  param[8] * psi1n[j-1] +
        param[9] * mat(j-1,0) + param[10] * mat(j-1,5) - psi_buys_adjust;
```

```
      psim1n[j] = const_sells * meandiff + mat(j,3) - (const_sells_sa * mat(j-1,3)) +
        param[11] * mat(j-1,1) + param[12] * mat(j-1,2) + param[13] * psim1n[j-1] +
        param[14] * mat(j-1,0) + param[15] * mat(j-1,5) - psi_sells_adjust;
    }
  } else {
    for(j = 1; j < ni; j++){
      psi1n[j] = const_buys * meandiff + mat(j,3) - (const_buys_sa * mat(j-1,3)) +
        param[4] * mat(j-1,1) + param[5] * mat(j-1,2) +  param[6] * psi1n[j-1] +
        param[7] * mat(j-1,0) + param[8] * mat(j-1,5);

      psim1n[j] = const_sells * meandiff + mat(j,3) - (const_sells_sa * mat(j-1,3)) +
        param[9] * mat(j-1,1) + param[10] * mat(j-1,2) + param[11] * psim1n[j-1] +
        param[12] * mat(j-1,0) + param[13] * mat(j-1,5);
    }
  }

  if(HMM) {
    NumericVector rt_buys_no_helper = no_init(ni);
    NumericVector rt_sells_no_helper = no_init(ni);
    NumericVector rt_buys_good_helper = no_init(ni);
    NumericVector rt_sells_bad_helper = no_init(ni);

    psi1g = psi1n - param["mu_G"];
    psim1b = psim1n - param["mu_B"];

    psi1n_helper = exp(param["k_B"] * (dura - psi1n));
    psim1n_helper = exp(param["k_S"] * (dura - psim1n));
    psi1g_helper = exp(param["k_B"] * (dura - psi1g));
    psim1b_helper = exp(param["k_S"] * (dura - psim1b));

    rt_buys_no_helper = exp(param["k_B"] * (rt - psi1n));
    rt_sells_no_helper = exp(param["k_S"] * (rt - psim1n));
    rt_buys_good_helper = exp(param["k_B"] * (rt - psi1g));
    rt_sells_bad_helper = exp(param["k_S"] * (rt - psim1b));

    tmp_res(_,2) = exp(log1p(-exp(-psi1n_helper - psim1n_helper)) -
                   log1p(-exp(-rt_buys_no_helper - rt_sells_no_helper)));
    tmp_res(_,1) = exp(log1p(-exp(-psi1g_helper - psim1n_helper)) -
                   log1p(-exp(-rt_buys_good_helper - rt_sells_no_helper)));
    tmp_res(_,0) = exp(log1p(-exp(-psi1n_helper - psim1b_helper)) -
                   log1p(-exp(-rt_buys_no_helper - rt_sells_bad_helper)));
  } else {
    NumericVector surv_buys_no = no_init(ni);
```

```
      NumericVector surv_sells_no = no_init(ni);
      NumericVector surv_buys_good = no_init(ni);
      NumericVector surv_sells_bad = no_init(ni);

      psi1g = psi1n - param["mu"];
      psim1b = psim1n - param["mu"];

      psi1n_helper = exp(dura - psi1n);
      psim1n_helper = exp(dura - psim1n);
      psi1g_helper = exp(dura - psi1g);
      psim1b_helper = exp(dura - psim1b);

      surv_buys_no = exp(-psi1n_helper);
      surv_sells_no = exp(-psim1n_helper);
      surv_buys_good = exp(-psi1g_helper);
      surv_sells_bad = exp(-psim1b_helper);

      tmp_res(_,2) = 1.0 - surv_buys_no * surv_sells_no;
      tmp_res(_,1) = 1.0 - surv_buys_good * surv_sells_no;
      tmp_res(_,0) = 1.0 - surv_buys_no * surv_sells_bad;
    }
    colnames(tmp_res) = CharacterVector::create("Bad", "Good", "No");

    res[i] = tmp_res;
  }
  return res;
}
```

Code Chunk 7.28 (Source code of function *dura_cdf_helper*):

```
function(para, HMM = TRUE) {
    dura_cdf_cpp(para, data_sub[["IntraDay"]], data_sub[["Daily"]][, "initial_buys"],
        data_sub[["Daily"]][, "initial_sells"], data_sub[["Daily"]][, "log_mean_dura"],
        data_sub[["Daily"]][, "log_mean_sa"], HMM)
}
```

The *summary_opt* function return summaries in a similar style of the *summary* function from the **stats** package for regression models estimated with the built-in *lm* function.[86] Hence, we get standard deviations, the test statistic of a t-test for significance and the corresponding p-value for each parameter. Standard deviations are calculated as square roots of the diagonal

[86] The *lm* function is also shipped with the **stats** package.

elements of the Hessian matrix. The *summary_opt* function shares the `control` argument with the *optimHess* function in the **stats** package which is used for computations of Hessian matrices.[87]

Code Chunk 7.29 (Source code of function *summary_opt*):

```
function(par, data = NULL, cluster_name = NULL, HMM = TRUE, lower = NULL, upper = NULL,
    control = list(), close_con = FALSE, return_cov_mat = FALSE) {
    if (close_con) {
        if (!is.null(cluster_name))
            on.exit(stopCluster(cluster_name))
    }
    if (is.null(lower)) {
        if (!HMM) {
            lower = c(-10, rep(0, 3), rep(-3, 2), rep(0, 2), -0.2, rep(-3, 2),
                rep(0, 2), -0.2, 0)
        } else {
            lower = c(rep(0, 6), rep(-3, 2), rep(0, 2), -0.2, rep(-3, 2), rep(0,
                2), -0.2, rep(0, 2), rep(1e-10, 2))
        }
    }
    if (is.null(upper)) {
        if (!HMM) {
            upper = c(rep(100, 4), rep(3, 2), 1, 0.3, 0.2, rep(3, 2), 1, 0.3,
                0.2, 2)
        } else {
            upper = c(rep(1, 6), rep(3, 2), 1, 0.3, 0.2, rep(3, 2), 1, 0.3,
                0.2, rep(2, 4))
        }
    }
    if (HMM) {
        fun <- function(par_est) loglik_hessian(par = trans_prob_stab(par_est),
            data = data, cluster_name = cluster_name, HMM = TRUE)
    } else {
        fun <- function(par_est) loglik_hessian(par = par_est, data = data,
            cluster_name = cluster_name, HMM = FALSE)
    }
    low_hit <- abs(lower - par) < 1e-08
    upper_hit <- abs(upper - par) < 1e-08
    bound_hit <- (low_hit | upper_hit)
    par_bound <- par[bound_hit]
    join_par <- function(x) c(par_bound, x)[names(par)]
    d <- calc_d(par, h = 1e-04, FUN = fun, lower = lower, upper = upper)
    con <- list(fnscale = 1, parscale = rep.int(1, length(par[!bound_hit])),
        ndeps = rep.int(0.001, length(par[!bound_hit])))
```

[87] A detailed explanation of the various options of the `control` argument is given on the help page of the *optimHess* function (call `?optimHess` from the R console).

```
    if (!(is.null(control$parscale))) {
        control$parscale <- control$parscale[!bound_hit]
    }
    if (!(is.null(control$ndeps))) {
        control$ndeps <- control$ndeps[!bound_hit]
    }
    con[(names(control))] <- control
    comp_hess <- FALSE
    comp_hess <- is.matrix(try(hessian1 <- optimHess(par = (par * d)[!bound_hit],
        fn = function(x) -fun(join_par(x/d[!bound_hit])), control = con), silent = FALSE))
    if (!comp_hess) {
        d <- rep(1, length(par))
        comp_hess <- is.matrix(try(hessian1 <- optimHess(par = (par * d)[!bound_hit],
            fn = function(x) -fun(join_par(x/d[!bound_hit])), control = con),
            silent = FALSE))
        if (!comp_hess)
          stop(paste0("Error in 'optimHess'.", "Hessian can not be computed for given parameter and/or con-
trol settings."))
    }
    hessian <- diag(d[!bound_hit]) %*% hessian1 %*% diag(d[!bound_hit])
    eigen_vals <- eigen(hessian, symmetric = TRUE, only.values = TRUE)$values
    if (any(eigen_vals < 0))
        warning("Hessian Matrix is not positive definite")
    var_cov <- try(solve(hessian), silent = FALSE)
    inv_hess <- is.matrix(var_cov)
    if (!inv_hess) {
        stop("Can't invert negative hessian matrix!")
    }
    if (return_cov_mat) {
        return(var_cov)
    }
    st.dev <- sqrt(diag(var_cov))
    t.val <- par[!bound_hit]/st.dev
    p.val <- 2 * (1 - pnorm(abs(t.val)))
    res <- data.frame(cbind(Estimate = par[!bound_hit], Std.Error = st.dev,
        t.value = t.val, p.value = p.val), row.names = names(par[!bound_hit]))
    newdat <- matrix(nrow = length(par), ncol = ncol(res))
    rownames(newdat) <- names(par)
    colnames(newdat) <- colnames(res)
    newdat[, "Estimate"] <- par
    newdat[rownames(res), ] <- as.matrix(res)
    res <- newdat
    res
}
```

The source code of *summary_opt* introduces the yet unknown *loglik_hessian* function, which is a modification of the already explained *loglik* function by removing the checks if the sums of the parameters α_1, β_1 and $\alpha{-}1, \beta_{-1}$ are larger than unity.

Code Chunk 7.30 (Source code of function *loglik_hessian*):

```
function(par = NULL, data = NULL, cluster_name = NULL, HMM = TRUE) {
    if (is.null(par))
        stop("No parameter vector found")
    if (is.null(names(par)))
        stop("'par' has no names! See function 'par_names'")
    if (any(is.nan(par)))
        return(-Inf)
    if (!HMM) {
        if (length(par) != 15)
            stop("Length of 'par' does not match 3TW model")
        llexp3TW(par, data = data, cluster_name = cluster_name)
    } else {
        if (length(par) != 20)
            stop("Length of 'par' does not match HMM Weibull model")
        res <- llwei(par, data = data, cluster_name = cluster_name)
        mat_trans <- trans_mat(par[c("p_BG", "p_BN", "p_GB", "p_GN", "p_NB",
            "p_NG")], hessian = TRUE)
        p_init <- p_stat(par[c("p_BG", "p_BN", "p_GB", "p_GN", "p_NB", "p_NG")])
        fb_ll(res, p_init, mat_trans)
    }
}
```

It is obvious that the majority of presented functions in this chapter do not need to be called by the user directly. In most cases it will be sufficient to call the *pin_est* function and eventually post-process the returned results or utilize them in further analyses. However, for the sake of completeness, this chapter contains the complete source code of each function used to receive the results for the dynamic approaches to estimate the probability of informed trading displayed in chapter 11.

8 Data

8.1 High-Frequency Data

Before we begin to present and explain our data source, matching of trades and quotes, data preparation and rules we apply for detecting outliers in the data, we will shortly describe some general properties of high-frequency data.

The literature is not consistent in terminology for this type of datasets. Often used synonyms are *(intraday) transaction data*, *ultra-high-frequency data* or *tick data* (e.g see Hautsch 2011, p. 27). Every trade or quote occurring in the market is typically recorded. According to the level of details a high-frequency dataset exhibits, we can distinguish between five major levels (see Hautsch 2011, p. 28):

Trade data The transaction level is associated with information on individual trades consisting of

- the time stamp of trades,
- the price at which a trade was executed and the corresponding
- trade volume.

Trade and quote data Information on trades and quotes provides the most common form of transaction data containing

- the time stamp of trades and best ask/bid quote updates,
- the underlying best ask/bid quotes,
- the price at which a trade was executed,
- the traded volume,
- the trade direction (can be reconstructed with trade classification algorithms which we will describe later)
- the indicative depth associated with best ask and bid quotes

Fixed level order book data If the underlying trading system is a fully computerized system, often also (at least partial) information on the depth behind the market is available. This type of data contains the same information as above but provides also information on limit order activities behind the market. Based on such data it is possible to reconstruct the limit order book up to a fixed level.

Messages on all limit order activities Such data provide full information on any limit order activities, including time stamps, (limit) prices, sizes and specific attributes of limit order submissions, executions, cancellations and amendments. It allows to fully reproduce the trading flow and to reconstruct the limit order book at any point in time during continuous trading and allows for an exact identification of buyer-initiated or seller-initiated trades.

Data on order book snapshots Some datasets provide snapshots of the limit order book at equidistant time intervals avoiding the need for order book reconstructions. However, as they are recorded on an equidistant grid, the matching with the corresponding underlying trading process is difficult. Therefore, this data is only useful to study limit order book dynamics but is of limited use to analyze interactions between the order book and the trading process.

The last three mentioned types of high-frequency datasets allow to reconstruct the limit order book. Despite the availability of data containing full information, it is still a difficult undertaking. Transactions happening outside the regular trading hours including opening auctions, closing auctions and overnight market activities have to be taken into account, as Hautsch (2011) states. Since our primary goal is not the accurate rebuilding of order books, we will not delve any deeper in this specific field and proceed with the description of our data.

8.2 Data Source

We use trade and quote data over four years for all applications of models for the probability of informed trading. Data is available from January 1, 2007 to December 31, 2010 and covers two marketplaces.[88] All nine equities under consideration belong to the automobile industry.[89]

New York Stock Exchange (NYSE): Ford Motor Company (F), General Motors Company (GM), Honda Motor Co. Ltd (HMC), Johnson Controls International (JCI) and Toyota Motor Corp. (TM)

Xetra: Bayerische Motorenwerke AG (BMW), Continental AG (CON), Daimler AG (DAI) and Volkswagen AG (VOW)

[88] GM was delisted after June 1, 2009.

[89] We receive our raw data from Tick Data, LLC. (2019).

Official market opening for NYSE and Xetra are at 9:30 am and 9 am, while regular trading ends at 4 pm and 5:30 pm, respectively. Trade and quote databases are separated from each other and contain different additional market-specific information.

Trade files for the US equities offer the following information: *Date*, *Time*, *Price*, *Volume*, *Exchange Code*, *Sales Condition*, *Correction Indicator*, *Sequence Number*, *Trade Stop Indicator*, *Source of Trade*, *MDS 127/TRF*[90], *Exclude Record Flag* and *Filtered Price*.[91] Exemplary records from the raw trade data for NYSE are shown in table 8.1.

01/03/2007	08:33:11.093	39.81	100	D	T	0	2936	N	C	—	—	—
01/03/2007	09:30:31.040	39.81	100	D	—	0	12204	N	C	—	—	—
01/03/2007	09:30:32.593	39.82	200	D	—	0	12430	N	C	—	—	—
01/03/2007	09:30:47.381	39.54	20200	N	—	0	14756	N	C	—	—	—
01/03/2007	09:30:47.840	39.60	800	N	E	0	14903	N	C	—	—	—
01/03/2007	09:30:48.491	39.60	400	N	E	0	15035	N	C	—	—	—
01/03/2007	09:30:48.491	39.54	100	M	—	0	15036	N	C	—	—	—
01/03/2007	09:30:48.491	39.54	200	M	—	0	15037	N	C	—	—	—
01/03/2007	09:30:48.491	39.54	200	M	—	0	15038	N	C	—	—	—
01/03/2007	09:30:48.491	39.60	500	N	E	0	15039	N	C	—	—	—

Table 8.1: Exemplary raw trade data for HMC on January 3rd, 2007. Blank fields are marked with "—".

The first transaction on NYSE (*Exchange Code* = N) in table 8.1 occurs at 09:30:47.381 at a price of 39.54 and consists of 20200 shares. According to the manual by Nexa Technologies Inc. (2011), a value of 0 for the *Correction Indicator* states that it is a "regular trade that was not corrected, changed, or signified as cancel or error". The trade was not indicated as stop stock (*Stop Indicator* = 0) and stems from the *Consolidated Tape System (CTS)*.

Information delivered by quote files are: *Date*, *Time*, *Exchange*, *Bid Price*, *Ask Price*, *Bid Size*, *Ask Size*, *Quote Condition*, *Market Maker ID*, *Sequence Number*, *Bid Exchange*, *Ask Exchange*, *National BBO Indicator*[92], *NASDAQ BBO Indicator*, *Quote Cancel/Correction*, *Quote Source*[93]. Exemplary records from the raw quote data for NYSE are shown in table 8.2.

The first record of quotes file from table 8.2 has timestamp 04:15:01.554 and bid and ask prices of 35.64 and 0, respectively. It belongs to the *NYSE Arca* marketplace (*Exchange* = P, *Bid Exchange* = P, *Ask Exchange* = P) which is an electronic trading platform, and is labelled as regular (*Quote Condition = R*). Bid and ask sizes are given in number of round lots (100 share units) and equal 2 and 0, respectively. The source of the quote is *Consolidated Quote System (CQS)*. Although

[90] TRF = Trade Reporting Facility

[91] The fields for *MDS 127/TRF*, *Exclude Record Flag* and *Filtered Price* are very often empty like in the excerpt of raw trade data shown in table 8.1.

[92] NBBO = National Best Bid and Offer

[93] For detailed explanation of trade and quote file fields see the manual for US equities by Nexa Technologies Inc. (2011).

01/03/2007	04:15:01.554	P	35.64	0.00	2	0	R	—	558	P	P	1	2	—	C
01/03/2007	04:15:01.554	P	37.52	39.81	1	1	R	—	560	P	P	1	2	—	C
01/03/2007	04:33:45.492	P	39.25	39.81	1	1	R	—	2401	P	P	1	2	—	C
01/03/2007	07:20:58.677	P	39.25	0.00	1	0	R	—	17154	P	P	1	2	—	C
01/03/2007	07:47:57.105	P	39.25	41.44	1	1	R	—	21441	P	P	1	2	—	C
01/03/2007	08:14:00.817	P	39.25	39.54	1	4	R	—	27333	P	P	1	2	—	C
01/03/2007	08:30:06.967	D	0.00	0.00	0	0	R	AUTO	43874	D	D	0	2	A	C
01/03/2007	08:30:06.967	D	0.01	2000.00	2	2	R	NAQS	43876	D	D	0	2	A	C
01/03/2007	08:30:06.967	D	0.00	0.00	0	0	R	BTRD	43879	D	D	0	2	A	C
01/03/2007	08:30:06.967	D	0.00	0.00	0	0	R	CDRG	43881	D	D	0	2	A	C

Table 8.2: Exemplary raw quote data for HMC on January 3rd, 2007. Blank fields are marked with "—".

the two fields about BBO indicators are included for the sake of completeness, it is not recommended to use them and therefore we will not give any explanation (see Nexa Technologies Inc. 2011).

Trade data as well as quote data for the German stocks are less informative. Trade files contain the following fields: *Trade Date*, *Trade Time*, *Trade Price*, *Trade Volume*, *Trade Exchange*, *Price Adjustment*, *Filtered Price*. Exemplary records from the raw trade data for Xetra are shown in table 8.3.

01/02/2007	09:02:31.400	47.15000	6231	FRA	72.46719	—
01/02/2007	09:02:44.000	47.20000	77829	ETR	72.54404	—
01/02/2007	09:02:44.790	47.20000	3000	ETR	72.54404	—
01/02/2007	09:02:47.480	47.20000	4000	ETR	72.54404	—
01/02/2007	09:02:49.000	47.20000	2783	ETR	72.54404	—
01/02/2007	09:02:50.400	47.20000	2500	ETR	72.54404	—
01/02/2007	09:02:57.440	47.20000	1000	ETR	72.54404	—
01/02/2007	09:03:02.340	47.20000	5000	ETR	72.54404	—
01/02/2007	09:03:02.660	47.20000	375	ETR	72.54404	—
01/02/2007	09:03:02.660	47.24000	454	ETR	72.60552	—

Table 8.3: Exemplary raw trade data for DAI on January 2nd, 2007. Blank fields are marked with "—".

The first trade on Xetra (*Exchange* = ETR) in table 8.3 happens at 09:02:44.000 with a price of 47.20 and 77829 traded shares. According to the manual for German equities data, *Price Adjustment* comes with no supporting documentation by Xetra. We ignore this field, and additionally, filtered prices due to the many blanks for the preparation or our datasets.

Following information are available in the quote data files: *Date*, *Time Stamp*, *Exchange*, *Quote*,

Quote Size, *Bid Ask Flag*[94]. Exemplary records from the raw quote data for Xetra are shown in table 8.4.

01/02/2007	08:50:04.290	ETR	47.73000	45546	I
01/02/2007	08:50:07.910	ETR	47.73000	45886	I
01/02/2007	08:51:00.110	ETR	47.73000	50446	I
01/02/2007	08:51:47.630	ETR	47.73000	50526	I
01/02/2007	08:51:58.650	ETR	47.80000	54029	I
01/02/2007	08:52:05.670	ETR	47.73000	50526	I
01/02/2007	08:52:35.440	ETR	47.73000	51526	I
01/02/2007	08:53:11.110	ETR	47.73000	52526	I
01/02/2007	08:53:22.840	ETR	47.73000	53526	I
01/02/2007	08:53:34.360	ETR	47.73000	54526	I

Table 8.4: Exemplary raw quote data for DAI on January 2nd, 2007. Blank fields are marked with "—".

The first record of quotes file from table 8.4 has timestamp 08:50:04.290 and a quote of 47.73. Quote size equals 45546, while the last field labels the quote as *indicative price*.[95]

Before the actual data preparation we exclude transactions occurring outside regular market trading hours. These orders are typically ignored in models for the probability of informed trading and would substantially increase file sizes of trade and quote data.[96] We make use of the **RSQLite** R package, which offers an easy-to-use SQLite interface for R, to store complete trade and quote data (see Wickham, James, and Falcon 2014).

8.3 Data Preparation

Since available raw data exhibit varying levels of information about the trading activities depending on the marketplace, we need to apply different data preparation strategies for NYSE and Xetra. Raw files for US equities contain information about several marketplaces (e.g., NASDAQ and AMEX[97]), whereas the German raw files only distinguish between floor and electronic trading. For US stocks we will concentrate on transactions occurring at NYSE, which have to fulfill several constraints, for German symbols we will discard floor trading.

[94] For detailed explanation of trade and quote file fields see the manual for German equities by Nexa Technologies Inc. (2007).

[95] At the given time Xetra marketplace resides in pre-trading phase.

[96] We utilize the *TickWrite* program for pre-preparation of data files. The user guide by Tick Data, LLC. (2011) gives detailed explanations.

[97] NASDAQ = National Association of Securities Dealers Automated Quotations and AMEX = American Stock Exchange

8.3.1 New York Stock Exchange (NYSE)

We extract date, timestamp, price and volume information from the trade database incorporating the following conditions (see Vergote 2005)):

- trades have to be executed at NYSE
  `Exchange Code = N`
- trades need to be regular way or NYSE Direct+ trades; [98]
  `Sales Condition = "E"` or `Sales Condition = ""` or `Sales Condition = "@"` or `Sales Condition = "*"`
- source of trades must be *Consolidated Tape System* (CTS)
  `Source of Trade = "C"` or `Source of Trade = ""`
- trades must not be indicated as Stop Stock
  `Trade Stop Indicator = "N"`
- *Tick Data LLC* offers a recommendation that trades should be excluded due to condition code(s)
  `Exclude Record Flag` $\neq$ `"X"`

We gather date, timestamp, bid price, ask price, bids size and ask size fields for transactions from quote database if they meet the following constraints (see Vergote 2005)):

- exchange that issued the quote is NYSE
  `Exchange = "N"`
- exchange where the bid and ask prices originated from is NYSE
  `Bid Exchange = "N"` and `Ask Exchange = "N"`
- we consider only quotes which have its origin in normal trading (i.e., regular quotes, opening quotes and closing quotes)
  `Quote Condition = "R"` or `Quote Condition = "O"` or `Quote Condition = "C"`
- quote source needs to be *Consolidated Quote System* (CQS)
  `Quote Source = "C"`
- only non-corrected quotes are kept in database
  `Quote Cancel Correction = ""` or `Quote Cancel Correction = "A"`

[98] The high-speed electronic connection for immediate automatic execution of limit orders up to 1,099 shares (see Nexa Technologies Inc. (2011)).

8.3.2 Xetra Germany

Raw files for the German equities display information about orders originating from floor (Frankfurt) and electronic trading system (Xetra). We select the same fields in raw data as for NYSE but narrow our analysis to data from the Xetra system. Hence, we only need to check the *Exchange* field entries for keeping transactions in database:
`Exchange = "ETR"` or `Exchange = "ETE"`.

8.4 Matching of Trades and Quotes

Both exchanges, NYSE and Xetra, deliver data for trades and quotes in separate files which raises the problem of appropriate matching. Lee and Ready (1991) established the prominent "5-Second-Rule" which intends to reduce the mismatching of trade and quote data. This rule takes into account that quotes are posted more quickly and trades are recorded with some delay. Hence, the "5-Second-Rule" gives the recommendation to match trades with quote data which was posted at least five seconds earlier. However, it is not appropriate for any recent data since it was developed by analyzing trade and quote data from the late eighties. Henker and Wang (2006) show that the delay for NYSE stocks is rather one second than five seconds. Hautsch (2011) state that this is in line with most recent studies linking each trade arrival to the most recent quote.

For matching our trade and quote datasets we do not use any data-driven algorithm (e.g., see Hautsch 2011, p. 30), but use the most recent posted quote as the relevant one for each order. We compute year-wise average differences in timestamps of transactions and corresponding quote records for all stocks in our study. The results displayed in table 8.5 show that for the majority of US stocks yearly average delays are substantially lower than one second. Only HMC and TM exhibits delay times which are considerable higher than one second in 2008 and 2010.

German stocks present substantially higher delays than equities traded on NYSE. Only average differences in trade and corresponding quote timestamps for BMW and DAI are lower than one second in 2009 and 2010, CON equity suffers from the highest delay times which are more than three seconds in 2008 and nearly 10 seconds in 2009.

Table 8.6 offers summary statistics for matched datasets including information about average duration, average trade size, average number of transactions per day and the total number of records for all stocks over the range from 2007 to 2010. HMC and TM can be labelled as less-frequently traded stocks with average waiting times of almost 30 seconds, whereas the average durations for the remaining equities are at most 10 seconds. The average number of orders per day in our matched datasets is higher than 5000 for almost all stocks. Hence, we see a tremendously high number of observations for the nine equities under consideration. Even for HMC and TM the total amount of matched transactions is close to one million, maximum value is shown for DAI which equals almost 10 million.

	2007	2008	2009	2010
F	0.32917	0.52323	0.14389	0.32195
GM	0.27756	0.59960	0.23142	
HMC	0.57407	1.89123	0.37644	1.57884
JCI	0.38104	0.75234	0.19316	0.32347
TM	0.52561	1.34746	0.69363	1.47848
BMW	2.06983	1.60170	1.22111	0.61925
CON	2.08280	3.00380	9.72654	1.39777
DAI	1.36974	1.23455	0.88366	0.49265
VOW	1.81722	1.49404	1.84066	1.58838

Table 8.5: Average delay times for stocks traded on NYSE and Xetra (measured in seconds).

	F	GM	HMC	JCI	TM	BMW	CON	DAI	VOW
∅ Dura.	4.11	3.26	29.55	6.81	24.73	5.97	10.32	3.19	5.35
∅ Volume	1683.01	616.50	243.46	237.62	196.05	614.31	367.17	759.26	233.84
∅ Daily Trades	5675.98	7147.82	788.01	3424.25	941.87	5117.23	2958.61	9572.50	5707.86
# Obs.	5721386	4338728	794314	3451643	949401	5199102	3005951	9725655	5799188

Table 8.6: Summary statistics for matched datasets in the range from 2007 to 2010 including average duration, average trade size, average number of trades per day and the number of observation.

8.5 Data Cleaning

After matching trade and quote files we can begin to prepare the newly created datasets for estimation. To suppress effects by morning auctions which differ from regular trading we delete the first 20 minutes after the official market opening for each trading day which is common in recent PIN literature (e.g., see Tay, Ting, Tse, and Warachka 2009, Preve and Tse 2013). Therefore, timestamps in the matched data for NYSE-listed stocks starts at 9:50 am and for German stocks at 9:20 am. Buys' and Sells' ALACD processes in chapter 6 are initialized using the next ten minutes of data. Trading days which exhibit less than three records in this time span are removed. For the less-frequently traded US stocks HMC and TM 44 and 23 trading days offer insufficient number of trade records for initialization, CON and VOW are the only German equities for which trading days must be removed due to this condition. After deletion of affected days from the database, logarithmic mean duration of buys and sells in this ten minute time-window is set as start value for the corresponding ALACD recursions.[99] Data entries with execution times lying in this ten minute window are only utilized for initialization of ALACD processes and are deleted from datasets intended for estimation of PIN. Hence, trading begins at 10 am for US stocks and at 9:30 am for German equities in our cleaned data.

Trading days with special closings are excluded. For instance, on days around holidays market

[99] Classification of transactions is described in section 8.6.

attendees can place orders until 1 pm (NYSE)[100] or 2 pm (Xetra)[101] with a subsequent closing auction. Furthermore, if no market activities are recorded for the last 15 minutes of a trading day it is removed. This condition, however, is only active for one trading day over all nine equities and four years and deletes a trading day from the database for HMC. An overview about the number of removed transactions is given in table 8.7.

	F	GM	HMC	JCI	TM	BMW	CON	DAI	VOW
Morning Data	563298	402106	59228	209555	74190	281128	141472	506221	283689
Special Closing	10	6	10	10	10	4	4	4	4
Early Closing	0	0	1	0	0	0	0	0	0
#Obs. ALACD Init.	0	0	44	0	23	0	13	0	4

Table 8.7: Number of transactions removed to avoid morning effects (Morning Data), number of trading days removed due to special closing (Special Closing), number of trading days removed due to no market activities in the last 15 minutes of regular trading hours (Early Closing), number of trading days removed due to insufficient observations for initialization of ALACD processes (# Obs. ALACD Init.). Whole time range of the underlying data is taken into account.

Further filtering of matched data is performed utilizing rules similar to those presented in Hautsch (2011, p. 33) and Barndorff-Nielsen, Hansen, Lunde, and Shephard (2009, p. C7):

1. Delete transactions with a quote or transaction price equal to zero or being negative.
2. Delete transactions with a quote or transaction volume equal to zero or being negative.
3. Delete transactions with crossed prices for which bid price is higher than ask price and therefore exhibit negative spreads.
4. Delete entries whenever the price lies outside the interval
$$\left[\text{bid} - 2 \cdot \text{spread}; \text{ask} + 2 \cdot \text{spread}\right].$$
5. Delete all entries with the spread being weakly greater than 50 times the median spread of that trading day.
6. Delete all entries with the price being weakly higher than five times the median mid-quote of that trading day.
7. Delete entries for which the mid-quote deviate by more than 10 mean absolute deviations from a rolling centered median (excluding the observation under consideration)
8. Delete entries for which the price deviate by more than 10 mean absolute deviations from a rolling centered median (excluding the observation under consideration)

[100] NYSE offers a very detailed list about all special closings since 1885 which can is available online at `http://s3.amazonaws.com/armstrongeconomics-wp/2013/07/NYSE-Closings.pdf`

[101] Special closings for Xetra are all occurring on trading days before New Year's Eve.

We use 25 transaction before and after the current order to compute the local median mid-quote for rules 7 and 8. In general, the interval length in rule 4, thresholds in rules 5 and 6 as well as neighborhood size in the last two rules can be altered and adopted to specific data, whereat the chosen values are similar or equal to those in other publications. Barndorff-Nielsen, Hansen, Lunde, and Shephard (2009) and Hautsch, Kyj, and Oomen (2012) set an amount of 50 transactions as neighborhood size, Brownlees and Gallo (2006) report that choosing 30 orders before and after the current trade yields the best results applying their rule of outlier detection.

	F	GM	HMC	JCI	TM	BMW	CON	DAI	VOW
Rule 1	0	0	0	0	0	0	0	0	0
Rule 2	0	0	0	0	0	0	0	0	0
Rule 3	0	0	0	0	0	589	493	713	606
Rule 4	0	2	2	2	0	0	1	1	6
Rule 5	0	0	0	0	0	0	0	3	0
Rule 6	0	0	0	0	0	0	0	0	0
Rule 7	0	0	0	0	0	0	0	0	0
Rule 8	0	0	0	0	0	0	0	0	0

Table 8.8: Number of transaction records deleted due to violating Rule 1 to 8 after deletion of morning data, trading days with special closing and more than 15 minutes without any market activities as well as trading days with insufficient number of records for initialization of ALACD processes. Whole time range of the underlying data is taken into account.

Summarizing table 8.8, outliers are not a big problem our raw data files suffer from. For US stocks solely the fourth rule claims that records in the matched datasets should be excluded. Comparing the tremendous small number of "dangerous" entries to the remaining several millions of "clean" transactions in the range from 2007 to 2010, estimation results would probably not change if we were neglecting the outlier aspect for NYSE. For German equities, beside prices lying far away either from the bid or ask price and very large spreads, every stock suffers from crossed prices. Each of the German shares display transactions for which the bid price exceeds the ask price. However, this number of outliers detected by the third rule is still very small in relation to the total number of transaction records.

8.6 Filtering of Zero-Durations and Trade Classification

Zero durations occur if two or more consecutive transaction records have identical timestamps and are a typical characteristic of electronic trading systems. Estimation of the probability of informed trading is not possible for data including such special durations, since they constitute a severe problem in the dynamic models incorporating ALACD specifications to capture the dynamics of buys' and sells' waiting times. Simultaneous trades can be interpreted as one

large order broken into smaller pieces (split-transactions), for example, due to removing one or more levels of depth. The most common method for handling zero durations is an aggregation of the corresponding data entries (see Pacurar 2008).[102] We group sequences of matched transaction records with zero durations and compute volume-weighted prices as well as quotes. Order sizes and bid and ask volumes are aggregated. These newly calculated values replace the corresponding fields of the first element of the sequence, whereas remaining transactions are discarded. Figure 8.1 summarizes the frequencies of zero durations for all nine equities.

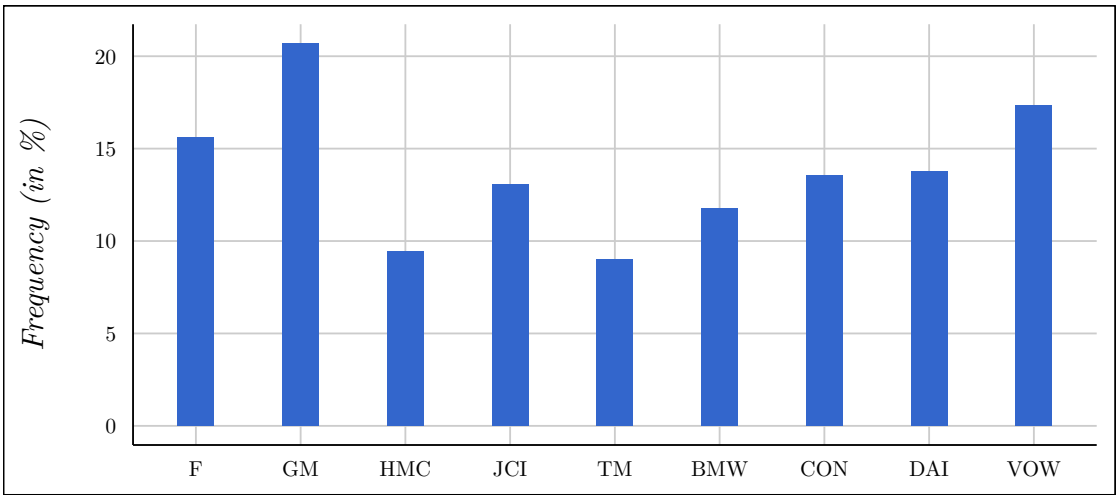

Figure 8.1: Frequency of zero durations for all equities after data cleaning. Timestamps and therefore durations are measured up to milliseconds.

For classification of transactions into buys and sells various algorithms can be utilized. The LR-method by Lee and Ready (1991) as well as the *quote test* requires trade and quote data, whereas the *tick test* (e.g., see Holthausen, Leftwich, and Mayers 1987) can be applied if only trade data is available. Likewise to LR, the EMO algorithm by Ellis, Michaely, and O'Hara (2000) requires trade and quote data.

Tick test marks transactions as buyer- (seller-)initiated if the current price is higher (lower) than the preceding. If both prices are identical, the most recent price change is relevant. Quote rule classifies transactions according to the distance of the current transaction price to the corresponding bid and ask price. For prices lying above (below) the mid-quote trades are identified as buyer- (seller-)initiated. Trades with prices equaling the mean of quotes cannot be classified with quote rule.

In literature, the LR-algorithm is most commonly used and states a combination of quote rule and tick test. For transactions with a price identical to the mid-quote the tick test is utilized. All other trades are classified according to the quote rule. Orders with values identical to ask

[102] Sometimes not only simultaneous orders are considered as split-transactions. For instance, the threshold for the difference in timestamps of trades which are potential parts of a split-transaction is set to one second in Hautsch (2004).

(bid) prices are identified as buys (sells) by EMO method (direct classification), for remaining transactions the tick test is utilized.

Several studies inspect the accuracy of the different classification approaches on varying marketplaces. Theissen (2000) investigates the performance on German stock markets, Odders-White (2000) studies the accuracy for equities traded on NYSE and Chakrabarty, Li, Nguyen, and Ness (2007) test algorithms for identification of the originator of transactions in electronic trading systems. Furthermore, Boehmer, Grammig, and Theissen (2007) analyze how misclassification influences the estimation of the probability of informed trading in the EKOP setting and come to the result that PIN is downward-biased if not all transactions are correctly labelled as either buys or sells. The magnitude of bias depends on model parameter constellation and the rate of false assigning.

We employ an approach similar to the EMO strategy to classify transactions in our data. Direct classification is active for trades with prices identical to either bid or ask, orders for which their transaction price differs from the quotes and their mean are processed by LR-algorithm and tick test decides about trade direction for orders happening at the mid-quote.[103]

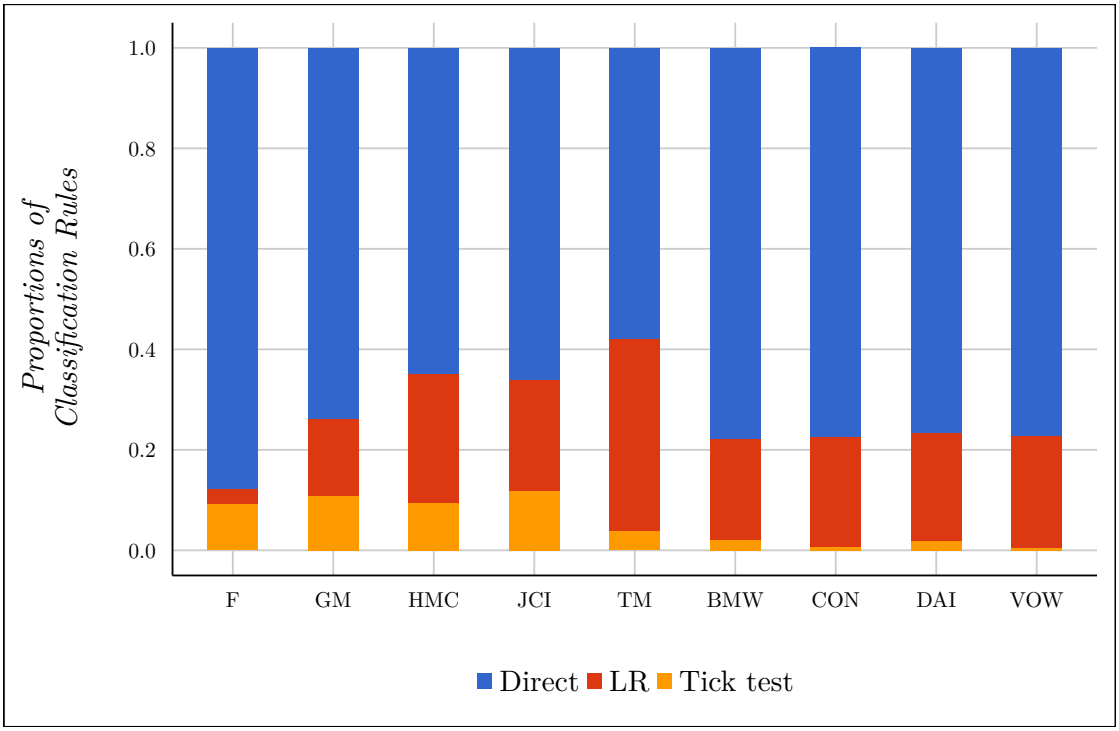

Figure 8.2: Summary of proportions which different classification rules occupy. By direct classification we check if transaction prices are identical either to bid or ask prices, remaining transactions are either classified by LR-algorithm or tick test.

[103] More sophisticated methods for modelling trade directions are available (e.g., see Rosenthal 2012). However, the improvement in accuracy is in the range from 1% to 2%.

The proportion of orders which we are able to directly classify varies across US stocks, as can be seen from figure 8.2. Almost 90% of trade prices of F equal bid or ask prices, whereas only little more than the half of all transactions of TM can immediately be identified as buyer- or seller-initiated. All German shares display nearly identical proportions for classification rules, roughly 77% of all orders allow a direct classification, 22% can be identified with LR-method and only 1% of all transactions display prices which are equal to the corresponding mid-quote.

The accuracy of LR-algorithm and tick test reported in the literature lies around 75% or above (see Chakrabarty, Li, Nguyen, and Ness (2007), Theissen (2000), Odders-White (2000)). Lee and Radhakrishna (2000) attest the LR classification approach even a rate of 93% for correctly identifying trade direction for NYSE stocks. Due to the huge number of records in our datasets which allow us to receive the trade initiator by comparisons of transaction prices and quotes we achieve a very high accuracy for all stocks. Hence, we are able to minimize the bias in our studies and estimation results induced by misclassification of trade direction.

Some summary statistics for cleaned datasets for which zero durations are filtered and trade directions are assigned for each record are presented in table 8.9. We see slightly higher average durations compared to table 8.6 and waiting times between two consecutive buyer- or seller-initiated orders are nearly identical for all stocks. Proportions of buys and sells are very similar. For the majority of US stocks the number of buys is slightly higher, whereas more sells are identified for all German equities. The *TD Switches* row offers information about the frequency of trade direction switches, either from buy to sell or vice versa. We see very similar rates for almost all equities and across markets. The lowest rate is shown by F with 26.31% which indicates longer series of consecutive buys and sells. Furthermore, there is no big difference in trade sizes of buyer- or seller-initiated transactions. The number of observations is still very high with a minimum of more than 600,000 transactions for HMC and a maximum of nearly 8,000,000 entries for DAI.

	F	GM	HMC	JCI	TM	BMW	CON	DAI	VOW
∅ Dura.	5.20	4.34	32.90	7.94	27.77	6.89	11.94	3.76	6.53
∅ Dura. (Buys)	10.25	8.53	65.70	15.87	55.02	13.79	24.05	7.52	13.31
∅ Dura. (Sells)	10.55	8.83	65.57	15.88	55.80	13.76	23.63	7.51	12.80
Freq. Buys (%)	50.72	50.88	49.95	50.01	50.34	49.93	49.56	49.98	49.03
Freq. Sells (%)	49.28	49.12	50.05	49.99	49.66	50.07	50.44	50.02	50.97
∅ Volume	1935.04	710.44	256.41	265.82	207.34	685.06	418.03	865.38	280.66
∅ Volume (Buys)	1961.14	718.42	259.66	265.38	209.65	684.01	417.68	859.86	268.56
∅ Volume (Sells)	1908.17	702.17	253.17	266.25	205.01	686.11	418.37	870.90	292.29
∅ Daily Trades	4154.36	4977.86	656.09	2720.52	777.33	4177.32	2409.01	7658.85	4405.47
∅ Daily Buys	2107.23	2532.71	327.73	1360.59	391.34	2085.75	1194.01	3827.68	2159.85
∅ Daily Sells	2047.13	2445.15	328.36	1359.92	385.99	2091.57	1215.00	3831.17	2245.62
TD Switches (%)	26.31	38.18	38.99	38.26	37.76	38.07	37.46	38.13	37.98
# Obs.	4146047	2991691	625253	2715074	757115	4227446	2406598	7750755	4440716

Table 8.9: Summary statistics for matched datasets in the range from 2007 to 2010 after cleaning rules are applied, zero durations are filtered and trade directions are identified. Table contains information about average durations, average trade sizes, average number of trades per day, frequency of switching trade direction and the number of observation.

After assigning trade direction to each record in our datasets we are able to calculate the aggregated number of daily buys and sells for every stock. These datasets are then utilized in estimations of the EHO model in chapter 10. One could argue that trading days with insufficient amount of data entries for initialization of buys' and sells' ALACD processes should be included in underlying datasets for optimizations in the static PIN model. This would be appropriate if solely likelihood function maximizations are conducted which require identical data structure. However, we estimate several models to receive the probability of informed trading. Static EHO model only needs the sequences of daily buys and sells but therefore drops a lot of information from the datasource. Models using high-frequency data incorporate much more information but the complexity of the data is therefore substantially higher. Since we want estimation results for the different approaches to be comparable in terms of the underlying data, we decide not to include the trading days removed due to ALACD initialization restriction in the datasets incorporated in EHO model optimizations.

8.7 Diurnally Adjustment

One of the empirical characteristics of high-frequency transaction data is the existence of intraday patterns of durations (see Tsay 2010, p. 238). Heavier trading activities and therefore lower durations can be observed in the hours shortly after market opening and before closing, whereas the number of orders reduces around lunch time. This leads to a reverted U- or bathtub-shape of seasonal figures.

According to Engle and Russell (1998) duration series are split into two parts, a deterministic (seasonal) component and a stochastic part. Furthermore, the seasonal component is assumed to act multiplicatively which allows to write the i-th observed duration on trading day d as product of corresponding diurnally-adjusted duration $\breve{x}_{i,d}$ and diurnal factor $\phi_i(t_{i-1})$ which depends on the timestamp of the preceding trade t_{i-1},

$$x_{i,d} = \breve{x}_{i,d} \cdot \phi_i(t_{i-1}). \tag{8.1}$$

Thus, the sequence of diurnally adjusted durations is given by the ratio of observed durations and diurnal factors,

$$\breve{x}_{i,d} = \frac{x_{i,d}}{\phi_i(t_{i-1})}. \tag{8.2}$$

As it is common practice in literature, after determination of seasonal figures we scale the adjusted duration series such that its sample mean is identical to the sample mean of the non-adjusted series for each trading day in our datasource (see Tay, Ting, Tse, and Warachka (2009) and Kwok, Li, and Yu (2009)),

$$\breve{x}_{i,d}{}^{\text{scaled}} = \breve{x}_{i,d} \frac{\bar{x}_d}{\bar{\breve{x}}_d}, \tag{8.3}$$

with mean of adjusted durations $\bar{\tilde{x}}_d$ and mean of non-adjusted durations $\bar{x}_d$ on trading day d.

There are various methods available in literature to compute the deterministic components $\phi_i(t_{i-1})$ (e.g., see Hautsch 2011, Tsay 2010, Bauwens and Giot 2000). We use a common approach which employs a cubic spline function to imitate the run of the intra-day pattern curve.[104] Knots of the spline function are set at each full hour during regular trading as well as at the first and last possible timestamp of a trading day in cleaned datasets.[105] Using higher rates for the placement of knots is possible but should be done carefully to avoid overfitting.[106] We assume the seasonal pattern to be constant over trading days and the time range under consideration for each of the stocks in our database. Other approaches like splitting trading days in morning and afternoon session (e.g., see Kwok, Li, and Yu 2009) or calculation of weekday-specific seasonal figures (e.g., see Meitz and Teräsvirta 2006) are also utilized in literature.

Values of fixed points of the cubic spline functions are calculated by averaging durations happening in a certain time span around each knot's timestamp. Furthermore, spline functions can be interpreted as expected durations depending on the time of day. Therefore the mean of the non-scaled adjusted durations from equation (8.2) should be close to 1. We investigate the performance of five different time ranges, utilizing intervals covering duration data from one to five minutes around each fixed timestamp.[107] Full hours are set as midpoint of the corresponding intervals, first and last possible timestamp of a trading day represent the left and right bound, respectively.

	1 min.	2 min.	3 min.	4 min.	5 min.
2007	1.06366	1.04375	1.03273	1.02651	1.01167
2008	1.08297	1.06273	1.04437	1.03562	1.01629
2009	1.11154	1.07619	1.05851	1.04245	1.02583
2010	1.10181	1.06742	1.05199	1.03703	1.02208

Table 8.10: Mean of diurnally adjusted durations from equation (8.2) averaged over all stocks. Interval lengths from one to five minutes are used to compute average durations for knots of cubic spline functions. GM is not included in computations for 2010.

Table 8.10 displays the means of non-scaled adjusted durations averaged over all stocks. An interval length of five minutes is a good choice over the range of years from 2007 to 2010 and achieves values which are closer to 1 in comparison with shorter time spans.

[104] We use the *splinefun* function from **stats** package. For more details see Forsythe, Malcolm, and Moler (1979).

[105] As mentioned before, trading days start at 10 am and end at 4pm for NYSE-listed stocks in our cleaned datasets, whereas trading for German shares starts at 9:30 am and ends at 5:30 pm.

[106] Knots are also often set every 30 minutes of a trading day (e.g., see Bauwens and Giot 2000).

[107] We think that the chosen interval lengths on the one hand are reasonably short to concentrate on the duration dynamics around fixed points of the spline functions and on the other hand cover enough durations for meaningful calculation of the average value.

Since we split the datasource for estimation by years[108] in the models utilizing high-frequency data (similar to Tay, Ting, Tse, and Warachka 2009), we need to compute diurnal patterns for each of the four years under consideration which are displayed in figures 8.3 and 8.4.

The seasonal figures for all stocks show the typical picture with lower durations in the first and last hours of a trading day with a period of higher durations and therefore less trading activities around lunch time. However, the reverted bathtub shape can much clearer be recognized for NYSE-listed than for Xetra-listed stocks. The peak for the German equities, which lies for all years and stocks around 1 pm, is much more pronounced.

The level of seasonal figures does not vary much in the range from 2007 to 2010 for most of the stocks in our study. GM shows a considerably increased magnitude of durations in the diurnal pattern of 2009 which may be induced by the delisting from NYSE in June. Two German equities, CON and VOW, also display increased levels of durations in 2009 and 2010, respectively.

[108] See the beginning of chapter 6.

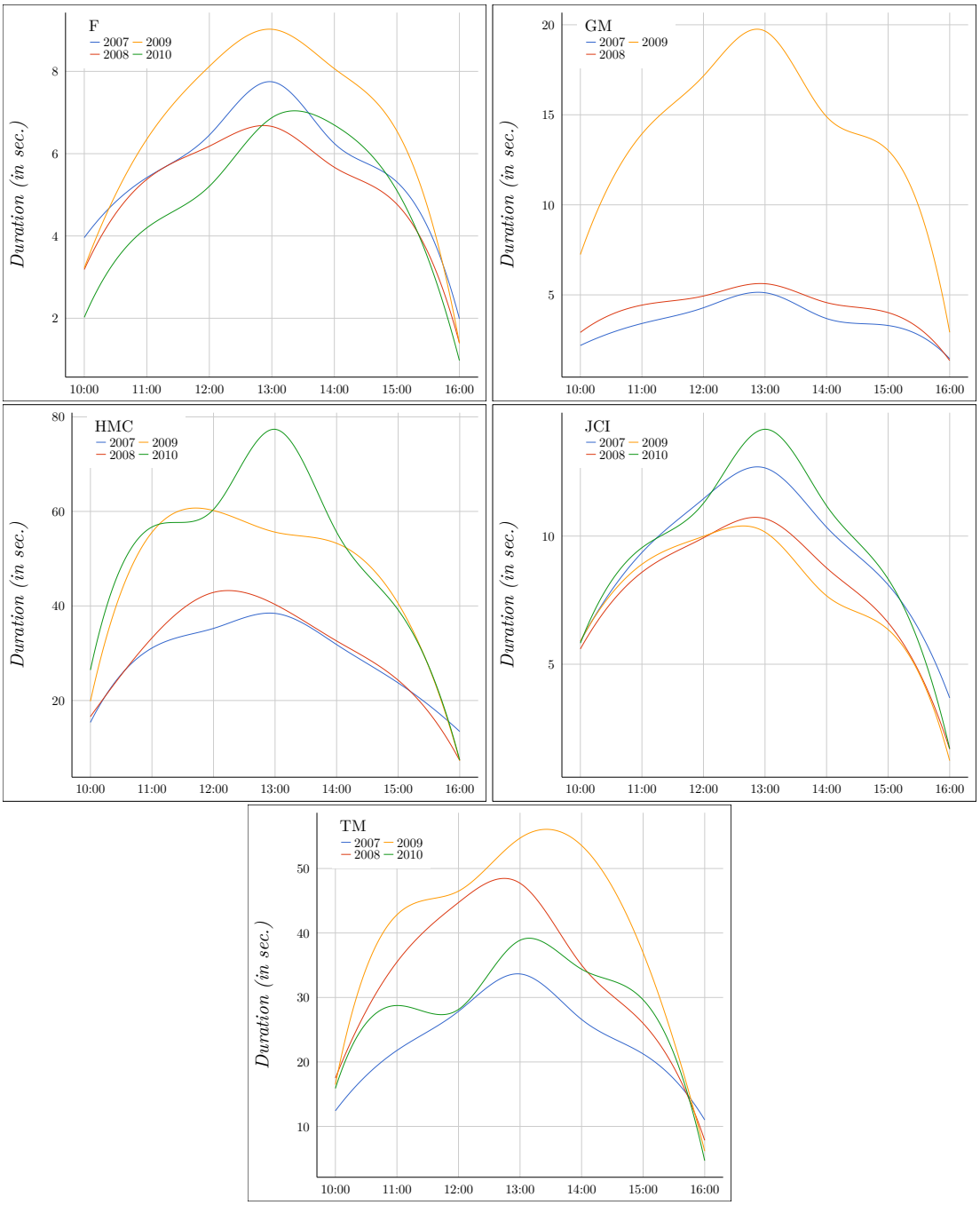

Figure 8.3: Seasonal figures for all NYSE equities from 2007 to 2010. Knots are placed at each full hour as well as at the first and last possible timestamp of a trading day.

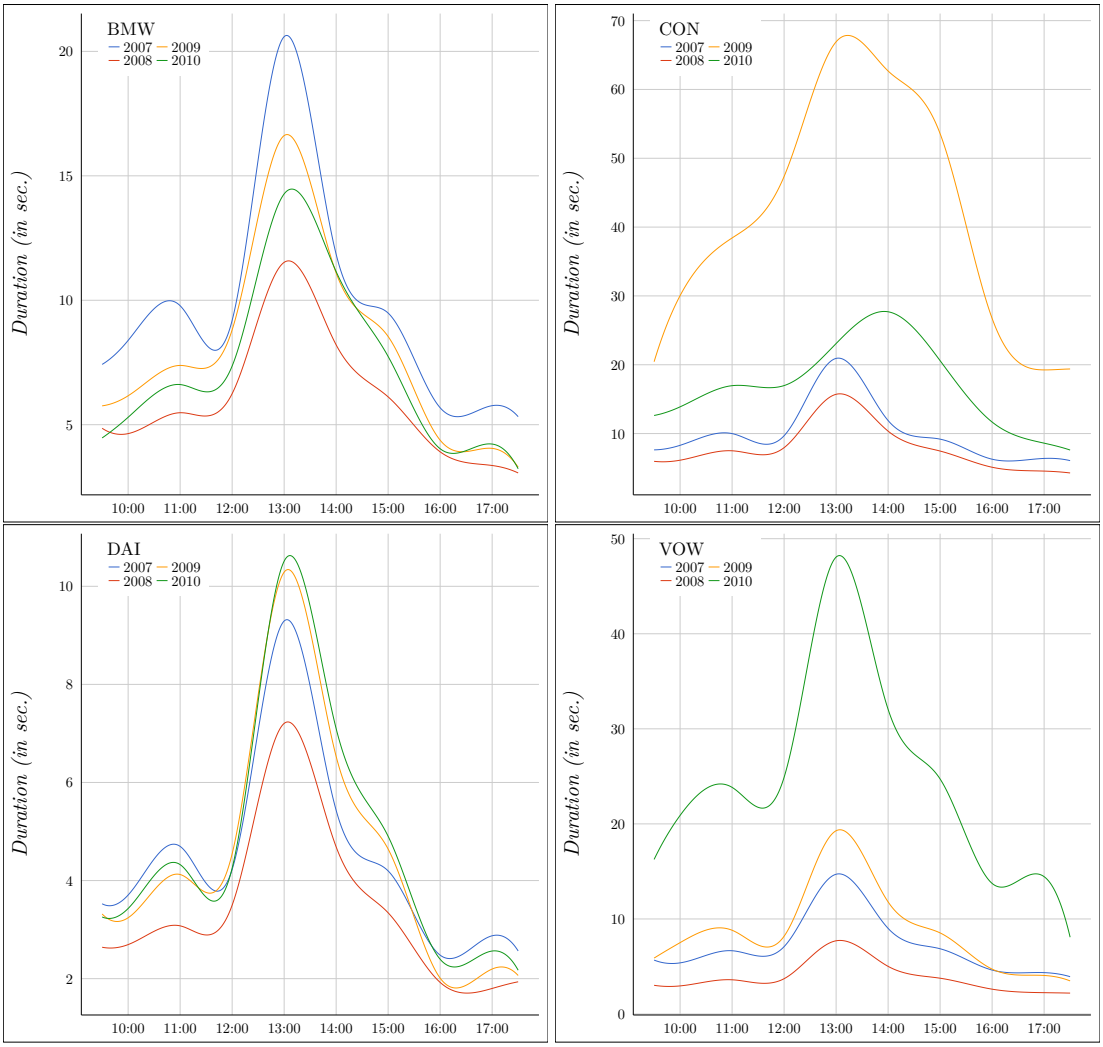

Figure 8.4: Seasonal figures for all Xetra equities from 2007 to 2010. Knots are placed at each full hour as well as at the beginning and end of a trading day.

8.8 Data for Static Models

When the raw data has undergone cleansing and most of the preparation steps, it is trivial to receive data in the structure needed for estimations of parameters in the static models. Since only the aggregated number of buys and sells per unit time interval is needed, classified datasets are sufficient. Hence, the operations described in sections 8.3 – 8.6 are mandatory, while the process of diurnally adjustment of durations does not influence the amount of buys and sells on trading days. Once zero durations are filtered and trade directions (buy or sell) are assigned to transactions, one can simply count the number of buys and sells for each trading day.

9 Source Code for Data Preparation

In order to get data which we can use in optimizations routines and subsequent computations presented in the previous chapter, raw data which is typically purchased by commercial providers needs to be prepared and cleaned in advance.[109] These preliminary steps are essential for the quality of results of maximization runs, since messy datasets can influence them to a huge degree. To speed up the data preparation we use the C++ language for time-consuming parts of the procedure. This chapter displays the source code of all functions which are involved in data preparation of the raw data. Furthermore, we briefly discuss their usage.

As already explained in chapter 8, we get trade and quote data in two different datasources and need to match them according to section 8.4. We utilize the *match_trades_quotes* function for this task which is presented in code chunk 9.1.[110] Lists of daily trade data and quote data can be passed to the `trades_day` and `quotes_day` arguments, respectively. The delay for matching can be specified with `delay`. This argument is set to zero by default, as explained in section 8.4. The marketplace can be chosen via `market`. Since we only consider equities from the NYSE and Xetra in this work, the only options for this function are `"US"` and `"DE"`.

Code Chunk 9.1 (Source code of function *data_prep*):

```
match_trades_quotes <- function(trades, quotes, delay = 0, market = c("DE",
    "US")) {
    if (length(trades) != length(quotes)) {
        stop("Different length of trades and quotes data")
    }

    start <- switch(market, DE = 32400, US = 34200)

    tmp <- vector("list", length(trades))

    for (i in 1:length(tmp)) {
```

[109] We received our high-frequency transaction data from *Tick Data, Inc.*.

[110] This function is customized to fit the characteristics of our raw data. If one purchased high-frequency data from a different vendor or different market or time horizon, this function must be adopted.

```
        tmp[[i]] <- trades[[i]]

        tmp[[i]] <- cbind(tmp, Bid = NA, `B-Vol` = NA, Ask = NA, `A-Vol` = NA,
            Dura = NA)

        time <- findInterval(trades[[i]][, "Timestamp"] - delay, quotes[[i]][,
            "Timestamp"], all.inside = TRUE)

        if (market == "DE") {
            tmp[[i]][, "Bid"] <- quotes[time, "Bid"]
            tmp[[i]][, "B-Vol"] <- quotes[time, "Bid.Vol"]
            tmp[[i]][, "Ask"] <- quotes[time, "Ask"]
            tmp[[i]][, "A-Vol"] <- quotes[time, "Ask.Vol"]
        }

        if (market == "US") {
            tmp[[i]][, "Bid"] <- quotes[time, "Bid"]
            tmp[[i]][, "B-Vol"] <- quotes[time, "Bid.Vol"] * 100
            tmp[[i]][, "Ask"] <- quotes[time, "Ask"]
            tmp[[i]][, "A-Vol"] <- quotes[time, "Ask.Vol"] * 100
        }

        tmp[[i]][, "Dura"] <- as.integer(diff(as.integer(c(start, tmp[[i]][,
            "Timestamp"]) * 1000)))
        tmp[[i]][, "Dura"] <- tmp[[i]][, "Dura"]/1000L
    }
    return(tmp)
}
```

To bring the matched raw data in a format that is expected by our estimation routines, we use the user-friendly wrapper function `data_prep` in code chunk 9.2. One could tag it with a label like *one-for-all*. Its main purposes are the filtering of the original high-frequency transactions, the classification of trades either as buys or sells[111], the calculation of initial values for the autoregressive specifications of the conditional expected durations[112] and finally the computation of diurnal factors for the series of observed durations.[113] Moreover, it computes the aggregated volumes which are needed to receive the probabilities of conditions of trading days in the PIN-ALACD model.

This very handy function expects the `data` argument to be a list of matched trades and quotes data. Each list element needs to exhibit observations for one trading day. Ideally, names of list elements are set to the date of the trading day so that these dates can be used to visualize optimization results afterwards. Observations for each trading day can be stored in data frames or matrices which need to have column names `Timestamp`, `Price`, `Volume`, `Bid`, `B-Vol`, `Ask` and `A-Vol`

[111] Both approaches, the filtering and the classification of trades, are explained in section 8.6.
[112] See section 8.5 for more details about how the initial values are calculated.
[113] The calculation of diurnal factors is described in section 8.7.

(in this order). The columns should contain data for the timestamps, the actual prices and the sizes of the transactions as well as for the bid and ask prices and their corresponding volumes. The `start` and `end` arguments can be used to define the time window of a trading day for which transactions are included in the computation of the probability of informed trading. Typically, `start` is set to to the official opening with a delay of 30 minutes and `end` is identical to the official closing of the underlying market. Both arguments expect integer values representing the chosen time of day in seconds after midnight. The `range` argument specifies the time range (in seconds) which is used in the computation of diurnal factors and defaults to five minutes. Hence, if the value of the `range` argument remains unchanged, averages of durations in five-minute intervals around each full hour as well as the `start` and `end` timestamp are assigned to the knots of the cubic spline function. If it is intended to set the unit time interval to one trading day for PIN estimations, the `intraday_interval` argument must be set to `NULL` which is also its default value. To prepare datasets for PIN estimations for intraday time intervals, `intraday_interval` can be specified as the length of the intraday unit time interval in seconds. The last argument of the function, `ncores`, takes a positive integer according to which the results are split for subsequent optimizations to run in parallel. By default, the value for this argument equals the number of available cores.

Code Chunk 9.2 (Source code of function *data_prep*):

```
function(data, start = NULL, end = NULL, range = 300, intraday_interval = NULL,
    ncores = detectCores()) {
    if (!is.list(data))
        stop("'data' needs to be a list!")
    check_colnames <- sapply(data, function(x) !colnames(x) %in% c("Timestamp",
        "Price", "Volume", "Bid", "B-Vol", "Ask", "A-Vol"))
    if (sum(check_colnames) ≥ 1)
        stop(paste0("Please check colnames of elements of \"data\" argument.",
            "They need to be: Timestamp, Price, Volume, Bid, B-Vol, Ask, A-Vol."))
    data <- clean_raw_data(data)
    data <- lapply(data, function(x) {
        x <- cbind(x, NA)
        colnames(x)[ncol(x)] <- "Dura"
        x
    })
    data <- lapply(data, function(x) {
        x[-1, "Dura"] <- diff(x[, "Timestamp"])
        x
    })
    data_filtered <- filtering(data)
    data_td <- trade_class(data_filtered)
    data_init <- lapply(data_td, function(x) subset(x, subset = x[, "Timestamp"] ≤
        start))
    acd_obs_min <- sapply(data_init, function(x) nrow(x) ≥ 3)
    data_init <- data_init[acd_obs_min]
    inits <- initial_values_alacd(data_init, start = start - 10 * 60, end = start)
```

```
data_reg <- lapply(data_td[acd_obs_min], function(x) subset(x, subset = x[,
    "Timestamp"] > start))
data_reg <- lapply(data_reg, function(x) {
    x[1, "Dura"] <- x[1, "Timestamp"] - start
    x
})
diurnal_factors <- diurnal_factors_spline(data_reg, start = start, end = end,
    range = range)
trading_time <- end - start
rawdura <- lapply(data_reg, function(x) x[, "Dura"])
cumsum.rawdura <- lapply(rawdura, function(x) c(0, cumsum(x)))
remain_time <- lapply(cumsum.rawdura, function(x) log(trading_time - x))
if (!(is.null(intraday_interval))) {
    n_days <- length(data_reg)
    intervals <- seq(from = start, to = end, by = intraday_interval)
    n_iv <- length(intervals) - 1
    interval_ind <- lapply(data_reg, function(x) findInterval(x[, "Timestamp"],
        intervals))
    data_reg_intraday <- Map(function(x, y) split(as.data.frame(x), y),
        data_reg, interval_ind)
    data_reg <- unlist(data_reg_intraday, recursive = FALSE)
    if (length(data_reg) < (n_days * n_iv)) {
        stop(paste0("Length of ", intraday_interval, " seconds is not sufficient. ",
            "Could not find data for every intraday interval."))
    }
    diurnal_factors_split <- unlist(Map(function(x, y) split(x[-length(x)],
        y), diurnal_factors, interval_ind), recursive = FALSE)
    remain_time_split <- unlist(Map(function(x, y) split(x[-length(x)],
        y), remain_time, interval_ind), recursive = FALSE)
    k <- 1
    for (i in seq_along(diurnal_factors_split)) {
        if (!(i%%n_iv)) {
            diurnal_factors_split[[i]] <- c(diurnal_factors_split[[i]],
              diurnal_factors[[k]][length(diurnal_factors[[k]])])
       remain_time_split[[i]] <- c(remain_time_split[[i]], remain_time[[k]][length(remain_time[[k]])])
            k <- k + 1
        }
    }
    diurnal_factors <- diurnal_factors_split
    remain_time <- remain_time_split
}
cumsum_trades <- cumsum_tr(data_reg)
split_ind <- data_split(cumsum_trades, ncores = ncores)
BSda <- BSday(data_reg)
mean.buy <- mean(BSda[, 1])
mean.sell <- mean(BSda[, 2])
tn_helper <- log(BSda[, 1] + BSda[, 2]) - log(mean.buy + mean.sell)
tg_helper1 <- log(BSda[, 2]) - log(mean.sell)
```

```
    tg_helper2 <- log(BSda[, 1]) - log(mean.buy)
    log_mean_sa <- numeric(length(data_reg))
    if (is.null(intraday_interval)) {
        for (j in 1:length(data_reg)) {
            rm <- length(diurnal_factors[[j]])
            log_mean_sa[j] <- log(mean(data_reg[[j]][, "Dura"]/diurnal_factors[[j]][-rm]))
        }
    } else {
        for (j in seq_along(data_reg)) {
            if (!(j%%n_iv)) {
                rm <- length(diurnal_factors[[j]])
                log_mean_sa[j] <- log(mean(data_reg[[j]][, "Dura"]/diurnal_factors[[j]][-rm]))
            } else {
                log_mean_sa[j] <- log(mean(data_reg[[j]][, "Dura"]/diurnal_factors[[j]]))
            }
        }
    }
    log_diurnal_factors <- lapply(diurnal_factors, function(x) log(x))
    log_dura <- lapply(data_reg, function(x) log(x[, "Dura"]))
    log_mean_dura <- sapply(log_dura, function(x) mean(x))
    volumen <- lapply(data_reg, function(x) log(x[, "Volume"]))
    td <- lapply(data_reg, function(x) as.integer(x[, "Td"]))
    ntd <- lapply(data_reg, function(x) as.integer(!(x[, "Td"])))
    signed_vol <- Map(function(vol, td) ifelse(td, vol, -vol), volumen, td)
    if (is.null(intraday_interval)) {
        log_dura <- lapply(log_dura, function(x) c(x, 0))
        signed_vol <- lapply(signed_vol, function(x) c(x, 0))
        td <- lapply(td, function(x) c(x, 0))
        ntd <- lapply(ntd, function(x) c(x, 0))
    } else {
        for (i in seq_along(log_dura)) {
            if (!(i%%n_iv)) {
                log_dura[[i]] <- c(log_dura[[i]], 0)
                signed_vol[[i]] <- c(signed_vol[[i]], 0)
                td[[i]] <- c(td[[i]], 0)
                ntd[[i]] <- c(ntd[[i]], 0)
            }
        }
    }
    tmp_data <- Map(cbind, log_dura, td, ntd, log_diurnal_factors, remain_time,
        signed_vol)
    tmp_data <- lapply(tmp_data, function(x) {
        colnames(x) <- c("log_dura", "td", "ntd", "log_diurnal_factor", "log_remain_time",
            "log_signed_vol")
        x
    })
    if (is.null(intraday_interval)) {
        daily_vals <- cbind(log_mean_dura, log_mean_sa, inits, tn_helper, tg_helper1,
```

```
            tg_helper2)
    } else {
        inits_intraday_buys <- inits_intraday_sells <- numeric(length(data_reg))
        for (i in seq_len(n_days)) {
            inits_intraday_buys[1 + (i - 1) * n_iv] <- inits[i, "initial_buys"]
            inits_intraday_sells[1 + (i - 1) * n_iv] <- inits[i, "initial_sells"]
            inits_intraday_buys[(2 + (i - 1) * n_iv):(i * n_iv)] <- sapply(tmp_data[(1 +
                (i - 1) * n_iv):(i * n_iv - 1)], function(x) {
                td_ind <- which(x[, "td"] == 1)
                subset <- x[td_ind, ]
                if (is.null(nrow(subset))) {
                  subset["log_dura"]
                } else {
                  subset[nrow(subset), "log_dura"]
                }
            })
            inits_intraday_sells[(2 + (i - 1) * n_iv):(i * n_iv)] <- sapply(tmp_data[(1 +
                (i - 1) * n_iv):(i * n_iv - 1)], function(x) {
                ntd_ind <- which(x[, "ntd"] == 1)
                subset <- x[ntd_ind, ]
                if (is.null(nrow(subset))) {
                  subset["log_dura"]
                } else {
                  subset[nrow(subset), "log_dura"]
                }
            })
        }
        daily_vals <- cbind(log_mean_dura, log_mean_sa, unlist(inits_intraday_buys),
            unlist(inits_intraday_sells), tn_helper, tg_helper1, tg_helper2)
        colnames(daily_vals)[3:4] <- c("initial_buys", "initial_sells")
    }
    if (length(split_ind) == 1) {
        res <- list(IntraDay = tmp_data, Daily = daily_vals)
        return(res)
    } else {
        res <- list(length(split_ind))
        for (i in seq_along(split_ind)) {
            res[[i]] <- vector("list", 2)
            names(res[[i]]) <- c("IntraDay", "Daily")
            res[[i]][["IntraDay"]] <- tmp_data[split_ind[[i]]]
            res[[i]][["Daily"]] <- daily_vals[split_ind[[i]], ]
        }
        return(res)
    }
}
```

By calling the *data_prep* function several helper functions are invoked. The first step is to clean the matched data according to the rules in section 8.5 with the *clean_raw_data* function whose

source code is given in code chunk 9.3.

Code Chunk 9.3 (Source code of function *clean_raw_data*):

```
function(matchedtq = NULL) {
    res <- vector("list", length(matchedtq))
    for (i in seq_along(matchedtq)) {
        bid <- matchedtq[[i]][, "Bid"]
        ask <- matchedtq[[i]][, "Ask"]
        bid_size <- matchedtq[[i]][, "B-Vol"]
        ask_size <- matchedtq[[i]][, "A-Vol"]
        price <- matchedtq[[i]][, "Price"]
        vol <- matchedtq[[i]][, "Volume"]
        spread <- ask - bid
        spread_median <- median(spread)
        mid_quote <- (bid + ask)/2
        mid_quote_median <- median(mid_quote)
        run_med_mid_quote <- runmed(mid_quote, k = 51)
        mean_abs_dev_mid <- mean(abs(mid_quote - mean(mid_quote)))
        run_med_prices <- runmed(price, k = 51)
        mean_abs_dev_prc <- mean(abs(price - mean(price)))
        zero_neg_bid_ind <- which(bid ≤ 0)
        zero_neg_ask_ind <- which(ask ≤ 0)
        zero_neg_bid_size_ind <- which(bid_size ≤ 0)
        zero_neg_ask_size_ind <- which(ask_size ≤ 0)
        zero_price_ind <- which(price ≤ 0)
        zero_vol_ind <- which(vol ≤ 0)
        crossed_prices_ind <- which(bid ≥ ask)
        large_spread_ind <- which(spread > (50 * spread_median))
        large_diff_prc_mid_ind <- which(price ≥ 5 * mid_quote_median)
        prc_outside_spread_ind <- which(price < (bid - 2 * spread) || price >
            (ask + 2 * spread))
        run_med_mid_ind <- which((mid_quote - run_med_mid_quote) ≥ 10 * mean_abs_dev_mid)
        run_med_prc_ind <- which((price - run_med_prices) ≥ 10 * mean_abs_dev_prc)
        rm_total <- c(zero_neg_bid_ind, zero_neg_ask_ind, zero_neg_bid_size_ind,
            zero_neg_ask_size_ind, zero_price_ind, zero_vol_ind, crossed_prices_ind,
            large_spread_ind, large_diff_prc_mid_ind, prc_outside_spread_ind,
            run_med_mid_ind, run_med_prc_ind)
        if (length(rm_total) > 0) {
            res[[i]] <- matchedtq[[i]][-unique(rm_total), ]
        } else {
            res[[i]] <- matchedtq[[i]]
        }
    }
    res
}
```

In the next step the raw data is filtered to eliminate zero durations which is performed by

the *filtering* function with the corresponding source code given in code chunk 9.4. To speed up filtering, especially for big datasets, the core part is implemented with the C++ function `filtering_cpp` in code chunk 9.5 while pre-processing is done in R.

Code Chunk 9.4 (Source code of function *filtering*):

```
function(data) {
    for (i in seq_along(data)) {
        tmp <- data[[i]]
        remove <- numeric()
        zero_duras <- which(tmp[, "Dura"] == 0)
        if (length(zero_duras) > 0) {
            start_zero_seq <- zero_duras[diff(c(-Inf, zero_duras)) != 1] - 1
            end_zero_seq <- zero_duras[diff(c(zero_duras, Inf)) != 1]
            zero_seqs <- zero_seqs_cpp <- vector("list", length(start_zero_seq))
            for (j in seq_along(start_zero_seq)) {
                zero_seqs[[j]] <- start_zero_seq[j]:end_zero_seq[j]
                zero_seqs_cpp[[j]] <- (start_zero_seq[j] - 1):(end_zero_seq[j] -
                  1)
                if (-1 %in% zero_seqs_cpp[[j]]) {
                  zero_seqs_cpp[[j]] <- NULL
                }
                remove <- c(remove, zero_seqs[[j]][-1])
            }
            zero_seqs_cpp <- zero_seqs_cpp[!sapply(zero_seqs_cpp, is.null)]
            tmp <- filtering_cpp(as.matrix(tmp), zero_seqs_cpp)
        }
        if (length(remove) > 0)
            data[[i]] <- tmp[-remove, ]
    }
    return(data)
}
```

Code Chunk 9.5 (Source code of C++ function *filtering_cpp*):

```
NumericMatrix filtering_cpp(NumericMatrix & data, List zero_seq_list) {
  int i = 0, n = zero_seq_list.size(), first = 0;

  NumericVector price = data(_,1);
  NumericVector vol = data(_,2);
  NumericVector bid = data(_,3);
  NumericVector bid_size = data(_,4);
  NumericVector ask = data(_,5);
  NumericVector ask_size = data(_,6);
  IntegerVector zero_seq;
```

```
  NumericVector vol_tmp;
  NumericVector bidsize_tmp;
  NumericVector bid_tmp;
  NumericVector asksize_tmp;
  NumericVector ask_tmp;
  NumericVector price_tmp;

  int agg_vol = 0, agg_b_vol = 0, agg_a_vol = 0;

  for(i = 0; i < n; i++) {
    zero_seq = zero_seq_list[i];
    first = zero_seq[0];
    vol_tmp = vol[zero_seq];
    bidsize_tmp = bid_size[zero_seq];
    bid_tmp = bid[zero_seq];
    asksize_tmp = ask_size[zero_seq];
    ask_tmp = ask[zero_seq];
    price_tmp = price[zero_seq];

    agg_vol = sum(vol_tmp);
    agg_b_vol = sum(bidsize_tmp);
    agg_a_vol = sum(asksize_tmp);
    data(first,1) = sum(vol_tmp * price_tmp)/(agg_vol);
    data(first,4) = sum(vol_tmp * bid_tmp)/(agg_vol);
    data(first,5) = sum(vol_tmp * ask_tmp)/(agg_vol);
    data(first,2) = agg_vol;
    data(first,4) = agg_b_vol;
    data(first,6) = agg_a_vol;
  }
  return data;
}
```

After filtering, the direction of transactions need to be identified which is done by the *trade_class* function in code chunk 9.6. Like for the *filtering* function, only the `data` argument can be specified for *trade_class*, which typically should be a result returned by the former function. Again, we implemented the actual classification algorithm in C++ with the `trade_class_cpp` function in code chunk 9.7 and do some post-processing of the results in R.

Code Chunk 9.6 (Source code of function *trade_class*):

```
function(data = NULL) {
    td_list <- lapply(data, function(x) trade_class_cpp(as.matrix(x)))
```

```
    res <- Map(function(x, y) {
        cbind(x, Td = y)
    }, data, td_list)
    res <- lapply(res, function(x) {
        rem <- which(is.na(x[, "Td"]))
        if (length(rem > 0))
            x[-rem, ] else x
    })
    res
}
```

Code Chunk 9.7 (Source code of C++ function *trade_class_cpp*):

```
IntegerVector trade_class_cpp(NumericMatrix & data) {
  int i = 0, j = 0, n = data.nrow();

  NumericVector price = round(data(_,1), 10);
  NumericVector bid = round(data(_,3), 10);
  NumericVector ask = round(data(_,5), 10);
  NumericVector midpoints = 0.5 * (bid + ask);
  IntegerVector td(n, IntegerVector::get_na());

  for(i = 0; i < n; i++) {
    // direct classification
    if(price(i) == ask(i)) {
      td(i) = 1;
    }
    if(price(i) == bid(i)) {
      td(i) = 0;
    }

    // LR algorithm
    if((price(i) != ask(i)) && (price(i) != bid(i)) && (price(i) > midpoints(i))) {
      td(i) = 1;
    }
    if((price(i) != ask(i)) && (price(i) != bid(i)) && (price(i) < midpoints(i))) {
      td(i) = 0;
    }

    // tick test
    if(price(i) == midpoints(i)){
      if(i == 0) continue;
```

```
      j = 1;
      while(IntegerVector::is_na(td(i))) {
        if((i-j) < 0) break;
        if(price(i) > price(i - j)) {
          td(i) = 1; //uptick
        }
        if(price(i) < price(i - j)) {
          td(i) = 0; //downtick
        }
        if(price(i) == price(i - j)) {
          j++; //zerotick
        }
      }
    }
  }
  return td;
}
```

The initial values for the autoregressive specifications of the conditional expected durations can be calculated with *initial_values_alacd*. In addition to the `data` argument, the beginning and the end of the used time interval for computations can be specified with arguments `start` and `end`, respectively. The source code of the function is displayed in code chunk 9.8.

Code Chunk 9.8 (Source code of function *initial_values_alacd*):

```
function(data = NULL, start = NULL, end = NULL) {
    init.buys <- numeric(length(data))
    init.sells <- numeric(length(data))
    morning.buys <- lapply(data, function(x) x[which(x[, "Td"] == TRUE), "Timestamp"])
    morning.sells <- lapply(data, function(x) x[which(x[, "Td"] == FALSE), "Timestamp"])
    morning.buys <- lapply(morning.buys, function(x) c(start, x, end))
    morning.sells <- lapply(morning.sells, function(x) c(start, x, end))
    for (i in 1:length(data)) {
        init.buys[i] <- log(mean(diff(morning.buys[[i]])))
        init.sells[i] <- log(mean(diff(morning.sells[[i]])))
        if (is.na(init.buys[i]))
            init.buys[i] <- init.sells[i]
        if (is.na(init.sells[i]))
            init.sells[i] <- init.buys[i]
    }
    res <- cbind(init.buys, init.sells)
    colnames(res) <- c("initial_buys", "initial_sells")
    return(res)
}
```

The *diurnal_factors_spline* function in code chunk 9.9 which computes diurnal factors for durations shares its arguments with *initial_values_alacd* but has one additional `range` argument. The extra argument specifies the range for timestamps around each knot of the spline function. Each full hour acts as center of the symmetric interval with length `range` which covers durations for the corresponding knot. For `start` and `end`, durations which occur `range` minutes after and before the specified timestamps are considered, respectively. Values of the knots are calculated as average duration in the corresponding interval and are returned by the `aver_dura` function which is internally called by *diurnal_factors_spline*. The source code of the `aver_dura` function is given in code chunk 9.10.

Code Chunk 9.9 (Source code of function *diurnal_factors_spline*):

```
function(data = NULL, start = NULL, end = NULL, range = NULL) {
    if (!(start%%3600)) {
        full_hour_start <- start + 3600
    } else {
        full_hour_start <- start + start%%3600
    }
    num_x_points <- (end - full_hour_start)/3600
    if (num_x_points%%1) {
        x_points <- numeric(ceiling(num_x_points))
    } else {
        x_points <- numeric(num_x_points)
    }
    for (i in 1:length(x_points)) {
        x_points[i] <- full_hour_start + 3600 * (i - 1)
    }
    x_points_total <- c(start, x_points, end)
    fixed_points <- aver_dura(data = data, start = start, end = end, range = range)
    cubic <- splinefun(x = x_points_total, y = fixed_points, method = "fmm")
    tmp.sa <- eval(expression(z), envir = environment(cubic))
    diurnal_factors <- lapply(data, function(x) cubic(c(start, x[, "Timestamp"])))
    return(diurnal_factors)
}
```

Code Chunk 9.10 (Source code of function *aver_dura*):

```
function(data = NULL, start = NULL, end = NULL, range = NULL) {
    helper0 <- lapply(data, function(x) subset(x, x[, "Timestamp"] ≥ start &
        x[, "Timestamp"] ≤ start + range)[, "Dura"])
    p0 <- mean(unlist(helper0))
    if (start%%3600) {
        full_hour_start <- start + start%%3600
    } else {
        full_hour_start <- start + 3600
```

```
    }
    num_spline_points <- (end - full_hour_start)/3600
    if (num_spline_points%%1) {
        spline_points <- numeric(ceiling(num_spline_points))
    } else {
        spline_points <- numeric(num_spline_points)
    }
    for (i in 1:length(spline_points)) {
        helper <- lapply(data, function(x) {
            subset(x, x[, "Timestamp"] ≥ full_hour_start + (i - 1) * 3600 -
                range/2 & x[, "Timestamp"] ≤ full_hour_start + (i - 1) * 3600 +
                range/2)[, "Dura"]
        })
        spline_points[i] <- mean(unlist(helper))
    }
    helper.end <- lapply(data, function(x) subset(x, x[, "Timestamp"] ≥ end -
        range & x[, "Timestamp"] ≤ end)[, "Dura"])
    pend <- mean(unlist(helper.end))
    res <- c(p0, spline_points, pend)
    return(res)
}
```

The *data_split* function in code chunk 9.11 is crucial for optimization routines to run in parallel. It ensures that the dataset is split so that each involved CPU core receives a very similar amount of transactions. By default, the dataset is split according to the maximum number of available CPU cores. The *cumsum_tr* function in code chunk 9.12 is a helper function which returns the cumulative sum of the number of total trades per day for the dataset. Those results should then be used to populate the `cumsumtrades` argument of the *data_split* function. The auxiliary argument `cur_max`, which saves the current maximum of entries of objects returned by the *cumsum_tr* function which belong to a specific CPU core, is needed to recursively call the *data_split* function for each worker. We utilize the built-in *Recall* function for recursive function calls.

Code Chunk 9.11 (Source code of function *data_split*):

```
function(cumsumtrades, ncores = detectCores(), cur_max = 0) {
    if (ncores == 1)
        return(list(1:length(cumsumtrades)))
    res <- list(ncores)
    anz <- cumsumtrades[length(cumsumtrades)]
    left <- max(which(cumsumtrades ≤ anz/ncores))
    if (cumsumtrades[left + 1] < max((anz - cumsumtrades[left])/(ncores - 1),
        cur_max)) {
        res[[1]] <- 1:(left + 1)
        cur_max <- max(cur_max, cumsumtrades[left + 1])
        res[2:ncores] <- lapply(Recall(cumsumtrades[-res[[1]]] - cumsumtrades[left +
```

```
            1], ncores = ncores - 1, cur_max = cur_max), function(x) x + left +
            1)
    } else {
        res[[1]] <- 1:left
        cur_max <- max(cur_max, (anz - cumsumtrades[left])/(ncores - 1))
        res[2:ncores] <- lapply(Recall(cumsumtrades[-res[[1]]] - cumsumtrades[left],
            ncores = ncores - 1, cur_max = cur_max), function(x) x + left)
    }
    res
}
```

Code Chunk 9.12 (Source code of function *cumsum_tr*):

```
function(data = NULL) {
    res <- sapply(data, function(x) nrow(x))
    res <- cumsum(res)
    res
}
```

The daily aggregated volumes of buys and sells which are involved in the computation of the probabilities of trading days' conditions in the PIN-ALACD model are returned by the *BSday* function in code chunk 9.13.

Code Chunk 9.13 (Source code of function *BSday*):

```
function(data) {
    ldata <- length(data)
    av.vol.b <- numeric(ldata)
    av.vol.s <- numeric(ldata)
    y <- lapply(data, function(z) as.integer(z[, "Td"]))
    for (i in 1:ldata) {
        av.vol.b[i] <- sum(data[[i]][, "Volume"][y[[i]] == 1])
        av.vol.s[i] <- sum(data[[i]][, "Volume"][!(y[[i]])])
    }
    result <- cbind(av.vol.b, av.vol.s)
    return(result)
}
```

If one intends to call the *initial_values_alacd* and *BSday* functions outside the scope of *data_prep*, it must be ensured that each list element has a column named `Td`, hence the dataframe should be classified in advance with *trade_class*.

Finally, there is the *cluster_prep* function whose `data` argument expects a list returned by *data_prep*. By the structure of the list, *cluster_prep* detects how many CPU cores should be utilized for the

optimization routines and sends portions of the list to each node of the newly created cluster. Its source code is given in code chunk 9.14.

Code Chunk 9.14 (Source code of function *cluster_prep*):

```
function(data = NULL, type = "PSOCK") {
    if (length(data) == 2 && all(names(data) %in% c("IntraDay", "Daily"))) {
        ncores <- 1
    } else {
        ncores <- length(data)
    }
    cl <- makeCluster(getOption("cl.cores", ncores), type = type)
    for (i in seq_along(cl)) {
        clusterCall(cl[i], function(d) {
            assign("data_sub", d, pos = .GlobalEnv)
            NULL
        }, data[[i]])
    }
    cl
}
```

Although we show every auxiliary function which is involved by a call to the main function *data_prep*, usually one does not have to call them directly. For the very most use cases it is sufficient to match raw trades and quotes data, call the *data_prep* function and then distribute the prepared data over a cluster with *cluster_prep*. After the cluster is created and each node exhibits its corresponding sub-dataset we are ready to use our functions for optimizations of both high-frequency models which we already explained in section chapter 7.

As already described in section 8.8, once the data is prepared, it is easy to obtain datasets which can be used for optimizations in the context of the static EHO model with functions provided by our **pinbasic** package presented in chapter 5. Hence, after the matched raw data has been successfully modified by our *data_prep* function, the returned object can be passed to the `prep_data` argument of the *agg_buys_sells* function in code chunk 9.15 to receive a matrix of daily aggregated buys and sells.

Code Chunk 9.15 (Source code of function *agg_buys_sells*):

```
function(prep_data = NULL) {
    if (is.null(data_prep))
        stop("No dataset provided. 'prep_data' is NULL.")
    if (is.null(names(prep_data))) {
        prep_data_intern <- lapply(prep_data, function(x) x[["IntraDay"]])
        prep_data_intern <- do.call("c", prep_data_intern)
    } else {
        prep_data_intern <- prep_data[["IntraDay"]]
```

```
    }
    mat <- matrix(data = NA, ncol = 2, nrow = length(prep_data_intern))
    colnames(mat) <- c("Buys", "Sells")
    for (i in seq_along(prep_data_intern)) {
        mat[i, "Buys"] <- sum(prep_data_intern[[i]][, "td"])
        mat[i, "Sells"] <- sum(prep_data_intern[[i]][, "ntd"])
    }
    mat
}
```

10 Empirical Applications (Static Models)

We apply the static model for the probability of informed trading by Easley, Hvidkjaer, and O'Hara (2002) on four years of data for nine equities from two different marketplaces. Timestamps of our data range from January 1, 2007 to December 31, 2010. We estimate model parameters and the probability of informed trading on a quarterly basis for equities which all belong to the automobile industry.

Ford Motor Company (F), General Motors Company (GM), Honda Motor Co. Ltd (HMC), Johnson Controls International (JCI) and Toyota Motor Corp. (TM) are traded on NYSE, while Bayerische Motorenwerke AG (BMW), Continental AG (CON), Daimler AG (DAI) and Volkswagen AG (VOW) are listed on the German electronic trading system Xetra. In section 10.1 we discuss the estimation results of the EHO model for the five securities listed on the NYSE, the parameter estimates for the German equities are presented in section 10.2.

In these two sections the focus lies on the EHO model which belongs to the static models for the probability of informed trading. As mentioned earlier, the PIN measure in this context can be interpreted as the a priori risk that a market maker has to face a trader which has access to private information. In the following section 10.3 we analyze outcomes for the posterior probabilities established in section 3.3 which shift the scope of the EHO model towards that of the dynamic models and the latent fraction of insider trading.

To improve the readability of sections 10.1 and 10.2, we place corresponding tables of estimation results as well as visualizations thereof at the end of each section.

10.1 Estimation Results (NYSE)

The MLE for US stocks yield estimates of α that have their maximum around the 50% level. The lowest and highest value for the probability of an information event are displayed for F in the third quarter of 2008 and JCI in the fourth quarter of 2007, respectively. No clear trend can be identified in the graphs for any stock. Each one contains several, more or less pronounced, ups and downs. For instance, F sees a strong decrease during 2007 from 47.5% to around 8%, whereas the estimate for JCI increases at the end of 2007 by more than 30% from about 17.5%

to 50%. In 2010, an increasing curve for all manufacturers can be observed, with F constituting an exception due to a huge decline of the probability of an information event from the first to the second quarter. The estimates for the supplier, JCI, do not behave in a similar way in 2010. The values of $\widehat{\alpha}$ reside in a quite small interval which ranges from 31.7% to 36.1%. Furthermore, we do not obtain any corner solutions for $\widehat{\alpha}$ for any US equity.

The range of the estimates of the conditional probability of bad-news trading days, $\hat{\delta}$, spreads wider than it does for $\widehat{\alpha}$. Every manufacturer has at least one boundary solution (e.g., $\hat{\delta}$ hits the upper bound of 1 for GM in the second quarter of 2009 and for TM in the third quarter of 2007). Hence, according to the static EHO model, these corner cases represent trading periods in which, if private information enters the market, solely informed sellers will be triggered to participate in the trading activities. For the supplier, JCI, the values for $\hat{\delta}$ neither reach the lower bound of 0 nor are they equal to unity over the complete time span of four years. The estimates of δ for JCI lie in the range from about 13% to 80%. Again, we see several (extreme) jumps in the plots for $\hat{\delta}$ for every equity.

We observe an overall negative trend for noise trading for almost all stocks from 2007 to 2010. Only F exhibits a slight increase in the market participation of uninformed traders. Exemplary, the estimated noise trading intensities, $\widehat{\epsilon_b}$ and $\widehat{\epsilon_s}$ steadily decrease since 2008 for GM. The delisting of GM in 2009 may be an explanation for this behavior. According to the estimation results, we also see some large positive jumps in the amount of transactions by uninformed traders for the equities under consideration in this work. However, these positive jumps are almost always directly followed by a movement in the opposite direction. A good example is the TM stock in 2010, which has its overall highest level of noise trading at the beginning of 2010 but suffers from a huge decline ending in the overall lowest level at the end of the year. The expected amount of transactions initiated by uninformed buyers sinks from 743 in the first quarter to 135 per day in the fourth quarter. The shrinkage of the intensity of uninformed sells is also remarkable. In the fourth quarter only less than one third of the amount of sells initiated by noise traders are expected compared to the beginning of the year.

The estimates of the intensity of informed trading, $\hat{\mu}$, show that for the stocks under consideration periods with large amount of transactions due to private information correspond with those of high noise trading activities. However, GM states an exception. While the noise trading is at its lowest level in the second quarter of 2009, the number of transactions initiated by insiders almost reaches its maximum. Hence, in the last quarter before the delisting of GM, according to the estimates of the EHO model, insiders clearly dominate the market activities. Likewise to the graphs for the noise trading intensities, we see large jumps in both directions for every stock. In the third quarter of 2008, the number of information-based transactions for F is substantially higher than in the surrounding quarters. A similar behavior can be found around the second quarter of 2010. Another example is the infrequently traded stock of TM in the beginning of 2010. At first, we see an outstanding increase followed by a similar decrease of $\hat{\mu}$. The PIN model expects a more than thirteen times higher rate of transactions initiated by insiders for TM which equals nearly 1776 insider transactions per day and then drops to a little more than 300 trades.

The probability of informed trading of NYSE-listed stocks lies in a relative small band spanning from about 10% to 20% for most quarters in the time from 2007 to 2010. Smallest and highest values are given by 0.06716 for JCI in the second quarter of 2007 and 0.21991 for HMC in the first quarter of 2009. For GM we see an overall higher insider trading probability after the first half of 2008 compared to the previous periods. The equity was delisted on NYSE in June 2009 since the company filed its bankruptcy and temporary had a price as low as 27 cents. Figure 10.1 shows the price crash in the time span from January 2008 to June 2009.

Figure 10.1: Daily close prices (in US $) of GM since 2008.

As mentioned before, the level of noise trading signals a downward-movement since 2008 for GM, whereas the activity of informed traders is relative high in 2008 with a peak in the fourth quarter and showing another peak in the second quarter of 2009. This parameter constellation may be a consequence of the EHO model assumption discussed at the beginning of chapter 6. The ratio of informed buys (sells) to the total number of buys (sells) on a good-news (bad-news) trading day is constant, as well as the intensity of noise trading. Therefore the EHO model labels the additional number of buys or sells on the corresponding information events completely as informed, neglecting the possibility that there is also higher activity by noise traders due to increased market activities.

The estimates of α are on low level and the estimates of δ exhibit a boundary solution. Hence, the static PIN model classifies all information-based trading as driven by private news with negative direction in the second quarter of 2009. There are only few days for which price-relevant information influence the market but we expect a high intensity of insiders selling on these days in the last period before the delisting.[114]

JCI exhibits the smallest variation in the estimates of the probability of informed trading. The difference between the lowest and highest value of $\widehat{\text{PIN}}$ is smaller than 7%. For the majority of

[114] Intraday prices for the last two trading days before the delisting of GM are displayed by figure 11.47 in section 11.5.

quarters the values lie between 8% and 10%. The biggest quarter-to-quarter movement happens from first to second quarter in 2009. The US scrappage program *CARS (Car Allowance Rebate System)* officially started on July 1, 2009. All US equities may be influenced by this federal program or rather by traders possessing private information about it. F displays relative high PIN estimates in 2009, but the value drops in the third quarter. HMC has ups and downs in 2009 with the highest estimate at the beginning, moreover TM exhibits relative low estimates from the end of 2008 until the third quarter of 2009 with an increasing probability of informed trading afterwards. The probability of informed trading for the supplier, JCI, shows an upward movement in the beginning of 2009 followed by a negative trend.

The empirical results displayed in this section show that the estimations of the probability of informed trading according to the EHO model do not suffer from the identification problem explained in section 4.3.4 for the US equities. For each quarter in the years from 2007 to 2010, at least two different types of trading are reflected by the parameter estimates and the intensity of informed traders μ is never small relative to the intensities of noise traders, ϵ_b and ϵ_s.

Quarter	F	GM	HMC	JCI	TM
2007(Q1)	0.47556	0.32789	0.39897	0.29505	0.27867
	(0.06397)	(0.06011)	(0.06339)	(0.05839)	(0.05740)
2007(Q2)	0.41261	0.38095	0.33426	0.17454	0.49998
	(0.06204)	(0.06118)	(0.05956)	(0.04784)	(0.06351)
2007(Q3)	0.08063	0.45161	0.32641	0.32259	0.31146
	(0.03457)	(0.06320)	(0.06032)	(0.05937)	(0.05929)
2007(Q4)	0.14516	0.30560	0.45418	0.50025	0.41776
	(0.04474)	(0.05846)	(0.06469)	(0.06353)	(0.06289)
2008(Q1)	0.42620	0.40984	0.40249	0.31153	0.29483
	(0.06332)	(0.06297)	(0.06860)	(0.05930)	(0.05853)
2008(Q2)	0.34375	0.28123	0.33916	0.40641	0.36666
	(0.05937)	(0.05620)	(0.06019)	(0.06141)	(0.06221)
2008(Q3)	0.06350	0.36508	0.40302	0.36505	0.32644
	(0.03072)	(0.06066)	(0.06232)	(0.06066)	(0.06101)
2008(Q4)	0.38712	0.33882	0.17743	0.45161	0.35441
	(0.06186)	(0.06011)	(0.04852)	(0.06320)	(0.06087)
2009(Q1)	0.29507	0.32787	0.45904	0.24590	0.32774
	(0.05839)	(0.06010)	(0.06380)	(0.05513)	(0.06011)
2009(Q2)	0.42839	0.16669	0.35572	0.33334	0.25417
	(0.06237)	(0.05751)	(0.06236)	(0.05939)	(0.05488)
2009(Q3)	0.14288	— — —	0.29242	0.49206	0.11290
	(0.04409)	— — —	(0.05986)	(0.06299)	(0.04019)
2009(Q4)	0.45151	— — —	0.32741	0.38705	0.46230
	(0.06320)	— — —	(0.06330)	(0.06186)	(0.06552)
2010(Q1)	0.34426	— — —	0.06897	0.36072	0.15529
	(0.06083)	— — —	(0.03327)	(0.06149)	(0.04757)
2010(Q2)	0.07936	— — —	0.38341	0.31747	0.36476
	(0.03405)	— — —	(0.06278)	(0.05865)	(0.06259)
2010(Q3)	0.25000	— — —	0.45802	0.34375	0.39683
	(0.05413)	— — —	(0.06494)	(0.05937)	(0.06164)
2010(Q4)	0.31750	— — —	0.46106	0.33311	0.40125
	(0.05865)	— — —	(0.06853)	(0.05942)	(0.06870)

Table 10.1: Quarterly estimates of the probability of an information event α for all NYSE equities. Figures in parentheses denote standard errors.

Quarter	F	GM	HMC	JCI	TM
2007(Q1)	0.62045	0.05002	0.50052	0.72225	0.47056
	(0.09013)	(0.04873)	(0.10236)	(0.10556)	(0.12106)
2007(Q2)	0.73092	0.83325	0.23755	0.54565	0.61283
	(0.08701)	(0.07610)	(0.09283)	(0.15020)	(0.08750)
2007(Q3)	1.00000	0.07142	0.89955	0.54995	1.00000
	(— — —)	(0.04865)	(0.06739)	(0.11125)	(— — —)
2007(Q4)	0.44445	0.47441	0.60322	0.22570	0.30506
	(0.16563)	(0.12058)	(0.09355)	(0.07512)	(0.09096)
2008(Q1)	0.26932	0.95999	0.28866	0.80489	0.50157
	(0.08701)	(0.03918)	(0.09655)	(0.11266)	(0.11810)
2008(Q2)	0.36363	0.38876	0.47690	0.73075	0.50007
	(0.10256)	(0.11489)	(0.10901)	(0.08701)	(0.10660)
2008(Q3)	0.75050	0.52172	0.60026	0.13038	0.10551
	(0.21621)	(0.10416)	(0.09803)	(0.07023)	(0.07078)
2008(Q4)	0.33364	0.14273	0.72725	0.64286	0.58506
	(0.09627)	(0.07630)	(0.13429)	(0.09055)	(0.10953)
2009(Q1)	0.88889	0.25002	1.00000	0.46686	0.65007
	(0.07407)	(0.09683)	(— — —)	(0.12882)	(0.10671)
2009(Q2)	0.70397	1.00000	0.19016	0.33334	0.48237
	(0.08792)	(— — —)	(0.08576)	(0.10287)	(0.12426)
2009(Q3)	0.88887	— — —	0.53073	0.67742	0.71427
	(0.10476)	— — —	(0.12139)	(0.08396)	(0.17075)
2009(Q4)	0.42871	— — —	0.16639	0.50132	0.80782
	(0.09353)	— — —	(0.08783)	(0.10245)	(0.07677)
2010(Q1)	0.61905	— — —	0.74996	0.54715	1.00000
	(0.10597)	— — —	(0.21652)	(0.10675)	(— — —)
2010(Q2)	0.39898	— — —	0.95653	0.35001	0.82583
	(0.21890)	— — —	(0.04250)	(0.10665)	(0.07947)
2010(Q3)	0.12487	— — —	0.18481	0.27273	0.04088
	(0.08260)	— — —	(0.07472)	(0.09495)	(0.04010)
2010(Q4)	0.75001	— — —	0.32131	0.52365	0.24938
	(0.09682)	— — —	(0.09383)	(0.10920)	(0.09676)

Table 10.2: Quarterly estimates of the conditional probability of bad-news trading days δ for all NYSE equities. Figures in parentheses denote standard errors.

Quarter	F	GM	HMC	JCI	TM
2007(Q1)	1755.81463	2658.15696	540.68703	1086.56334	593.96231
	(5.77432)	(7.92034)	(3.30287)	(4.31337)	(3.28969)
2007(Q2)	1428.93400	2734.78976	411.27067	895.23394	496.77911
	(4.98560)	(6.70174)	(2.93662)	(3.93789)	(3.04256)
2007(Q3)	2155.01601	2474.42657	381.56218	1076.17469	371.04951
	(5.88590)	(8.19442)	(2.52016)	(4.40700)	(2.46626)
2007(Q4)	1949.03348	2785.90729	254.31181	1264.67637	342.58348
	(5.74379)	(7.92242)	(2.20540)	(5.62409)	(2.73377)
2008(Q1)	1978.33319	3597.34805	328.47003	1628.70558	472.13702
	(6.62705)	(7.75134)	(3.70913)	(5.19613)	(3.06658)
2008(Q2)	2052.60206	2591.20169	276.55128	1326.99175	286.36690
	(6.29294)	(6.82730)	(2.27495)	(4.71175)	(2.34085)
2008(Q3)	2220.42187	1835.09086	268.39437	1059.69840	213.47505
	(5.98423)	(5.75314)	(2.23294)	(4.92658)	(2.45189)
2008(Q4)	1511.47801	1314.24753	372.22256	1344.98903	291.71590
	(5.58833)	(5.33956)	(2.49847)	(4.92387)	(3.24635)
2009(Q1)	1107.62645	646.97691	399.77071	1926.04908	400.26486
	(4.29093)	(3.72193)	(2.55985)	(5.88044)	(2.67570)
2009(Q2)	2009.21995	895.73761	209.93560	1309.83771	319.54983
	(6.15204)	(4.61898)	(2.21807)	(5.06580)	(2.35290)
2009(Q3)	1739.35763	— — —	148.82974	1322.63233	188.13397
	(5.30121)	— — —	(1.82575)	(4.84433)	(1.76512)
2009(Q4)	1725.03996	— — —	149.74482	1092.50513	162.68903
	(5.87413)	— — —	(1.93018)	(4.97493)	(2.01814)
2010(Q1)	2421.20099	— — —	235.83671	984.59476	743.34570
	(6.63836)	— — —	(2.03592)	(5.01329)	(3.58062)
2010(Q2)	2782.33953	— — —	340.30298	1477.57639	421.66846
	(6.74985)	— — —	(2.38927)	(5.31474)	(2.63378)
2010(Q3)	1427.63466	— — —	154.88140	917.42007	161.02610
	(5.35659)	— — —	(2.05045)	(4.26954)	(2.12813)
2010(Q4)	1790.39607	— — —	113.93090	836.12991	134.81503
	(5.47328)	— — —	(1.99828)	(3.98489)	(1.93576)

Table 10.3: Quarterly estimates of the intensity of uninformed buys ϵ_b for all NYSE equities. Figures in parentheses denote standard errors.

Quarter	F	GM	HMC	JCI	TM
2007(Q1)	1375.73999	2505.90446	293.61349	1005.41229	474.19293
	(5.44524)	(6.43116)	(2.48186)	(4.49637)	(2.95408)
2007(Q2)	1259.02486	2112.12524	260.15514	898.37945	325.76516
	(5.27598)	(6.85206)	(2.10767)	(3.89323)	(2.62628)
2007(Q3)	2217.38374	3416.13456	314.06583	1094.45789	294.04751
	(6.27790)	(7.49335)	(2.97993)	(4.51207)	(2.64563)
2007(Q4)	1891.99318	3279.10286	265.46824	1456.97260	451.94894
	(5.74465)	(7.28306)	(2.76609)	(5.04916)	(3.07319)
2008(Q1)	2298.46039	3026.16209	402.42553	1455.27732	487.24764
	(6.35728)	(9.04513)	(3.47594)	(5.71108)	(3.05501)
2008(Q2)	2232.90292	2850.21048	352.18229	1256.99889	270.83812
	(6.12668)	(6.88343)	(2.56874)	(5.15922)	(2.30774)
2008(Q3)	2097.64406	1598.72993	309.95711	1245.27911	217.64963
	(6.11294)	(5.46106)	(2.52770)	(4.50732)	(2.08668)
2008(Q4)	1369.66328	1694.28038	451.09830	1348.17323	316.98663
	(4.93527)	(5.30438)	(2.85942)	(5.25214)	(3.46206)
2009(Q1)	790.07258	807.17780	300.42415	2005.96820	362.40106
	(4.18878)	(3.71286)	(3.01654)	(6.00857)	(2.70483)
2009(Q2)	1656.96085	613.88561	211.31473	1368.71935	335.13610
	(5.90219)	(4.18962)	(1.94290)	(4.83029)	(2.46786)
2009(Q3)	1450.47478	— — —	145.33722	1056.98380	189.66490
	(5.11626)	— — —	(1.68389)	(4.74551)	(1.81681)
2009(Q4)	1452.97565	— — —	220.86008	1011.22973	139.20155
	(5.22549)	— — —	(2.04902)	(4.84468)	(2.06070)
2010(Q1)	2085.33074	— — —	226.55725	967.69002	501.69466
	(6.46631)	— — —	(2.01884)	(5.12906)	(3.19761)
2010(Q2)	2979.85404	— — —	246.85920	1620.07791	330.58244
	(6.96267)	— — —	(2.55728)	(5.26054)	(4.19902)
2010(Q3)	1524.91437	— — —	200.90216	1051.21457	241.60655
	(4.93083)	— — —	(1.89156)	(4.17764)	(2.02301)
2010(Q4)	1315.79833	— — —	139.40389	775.82530	148.52725
	(5.18208)	— — —	(1.69089)	(3.85179)	(2.08481)

Table 10.4: Quarterly estimates of the intensity of uninformed sells ϵ_s for all NYSE equities. Figures in parentheses denote standard errors.

Quarter	F	GM	HMC	JCI	TM
2007(Q1)	853.98790	1782.06864	273.86763	621.14515	464.63534
	(9.98121)	(17.27236)	(5.77629)	(10.19705)	(7.97376)
2007(Q2)	978.58313	1285.38334	246.87026	739.87276	261.56879
	(10.46062)	(13.31546)	(5.86382)	(12.64807)	(5.03341)
2007(Q3)	3973.66563	1927.07313	332.86343	751.11793	448.42533
	(33.44268)	(14.97564)	(6.57647)	(10.14700)	(6.78748)
2007(Q4)	2818.72337	1989.96370	171.75441	577.44495	291.18265
	(23.56042)	(17.55935)	(4.53417)	(9.06519)	(5.75775)
2008(Q1)	1232.64878	1754.67793	219.57047	930.70830	370.51005
	(12.35580)	(16.78211)	(7.34994)	(12.40308)	(7.60414)
2008(Q2)	1789.80346	1577.90026	234.33177	729.53133	283.84383
	(13.79079)	(16.06256)	(5.44528)	(9.67353)	(5.32935)
2008(Q3)	8380.20406	1374.35625	322.58466	849.90682	194.87375
	(96.07541)	(12.21875)	(5.34785)	(10.23153)	(5.75406)
2008(Q4)	1523.01424	1675.36128	530.26602	586.42866	230.51791
	(11.95092)	(12.77002)	(9.60614)	(9.17201)	(5.43805)
2009(Q1)	1088.13319	676.47718	430.00369	1309.38739	342.67937
	(11.01866)	(8.78142)	(5.92919)	(15.36041)	(6.37352)
2009(Q2)	1322.85320	1709.40124	197.41057	1243.52482	369.84419
	(11.88267)	(18.78680)	(4.82247)	(11.65185)	(6.87153)
2009(Q3)	2315.72932	— — —	262.71373	683.00855	457.78041
	(21.39051)	— — —	(5.34378)	(8.48999)	(9.70734)
2009(Q4)	1178.32464	— — —	209.73307	662.81951	135.48208
	(10.81341)	— — —	(4.85033)	(9.06684)	(3.99091)
2010(Q1)	2248.88480	— — —	610.54808	754.93535	1775.99211
	(15.68463)	— — —	(14.55299)	(9.39993)	(16.20178)
2010(Q2)	7643.03916	— — —	299.90807	1117.25504	303.57386
	(41.59035)	— — —	(5.48671)	(12.27625)	(7.35022)
2010(Q3)	2621.73297	— — —	155.26591	821.29334	244.16563
	(16.90493)	— — —	(3.86340)	(9.60032)	(4.51288)
2010(Q4)	2493.28099	— — —	148.92801	476.44721	129.87164
	(14.49065)	— — —	(3.72122)	(8.40326)	(4.44211)

Table 10.5: Quarterly estimates of the intensity of informed trading μ for all NYSE equities. Figures in parentheses denote standard errors.

Quarter	F	GM	HMC	JCI	TM
2007(Q1)	0.11480	0.10165	0.11580	0.08055	0.10811
2007(Q2)	0.13060	0.09176	0.10945	0.06716	0.13718
2007(Q3)	0.06827	0.12873	0.13509	0.10042	0.17355
2007(Q4)	0.09627	0.09113	0.13049	0.09595	0.13277
2008(Q1)	0.10940	0.09794	0.10787	0.08594	0.10222
2008(Q2)	0.12554	0.07540	0.11222	0.10293	0.15738
2008(Q3)	0.10971	0.12749	0.18354	0.11864	0.12858
2008(Q4)	0.16987	0.15873	0.10255	0.08953	0.11833
2009(Q1)	0.14471	0.13234	0.21991	0.07569	0.12836
2009(Q2)	0.13388	0.15878	0.14288	0.13401	0.12556
2009(Q3)	0.09398	— — —	0.20707	0.12376	0.12034
2009(Q4)	0.14340	— — —	0.15632	0.10869	0.17182
2010(Q1)	0.14661	— — —	0.08346	0.12241	0.18135
2010(Q2)	0.09524	— — —	0.16377	0.10274	0.12831
2010(Q3)	0.18166	— — —	0.16658	0.12542	0.19397
2010(Q4)	0.20309	— — —	0.21325	0.08963	0.15534

Table 10.6: Quarterly estimates of the probability of informed trading PIN for all NYSE equities.

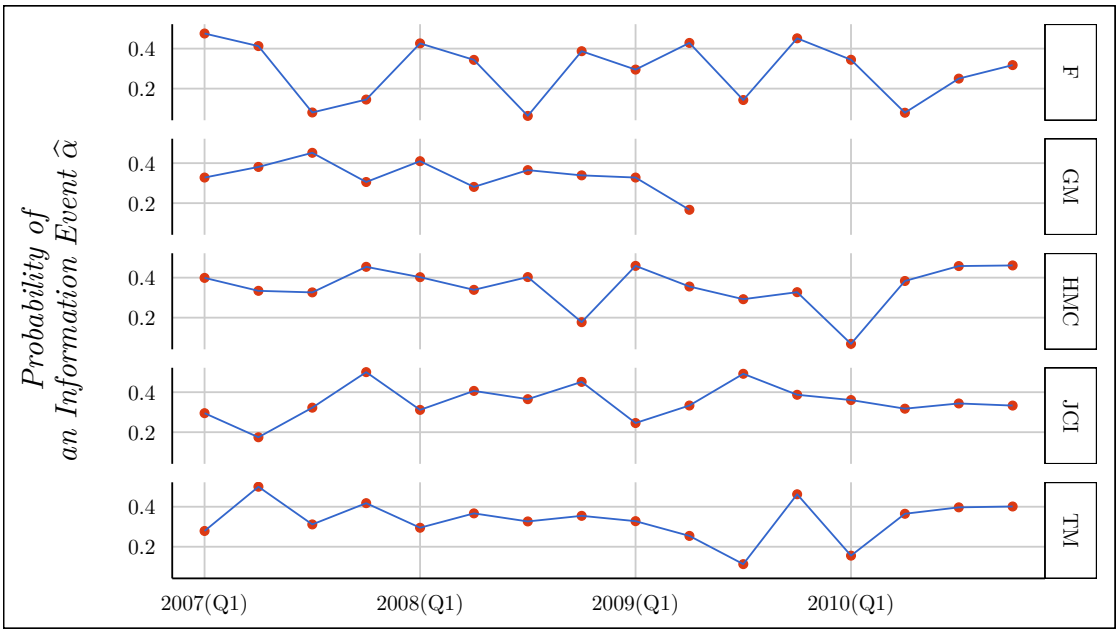

Figure 10.2: Quarterly estimates of the probability of an information event α for all NYSE equities. Detailed results can be found in table 10.1.

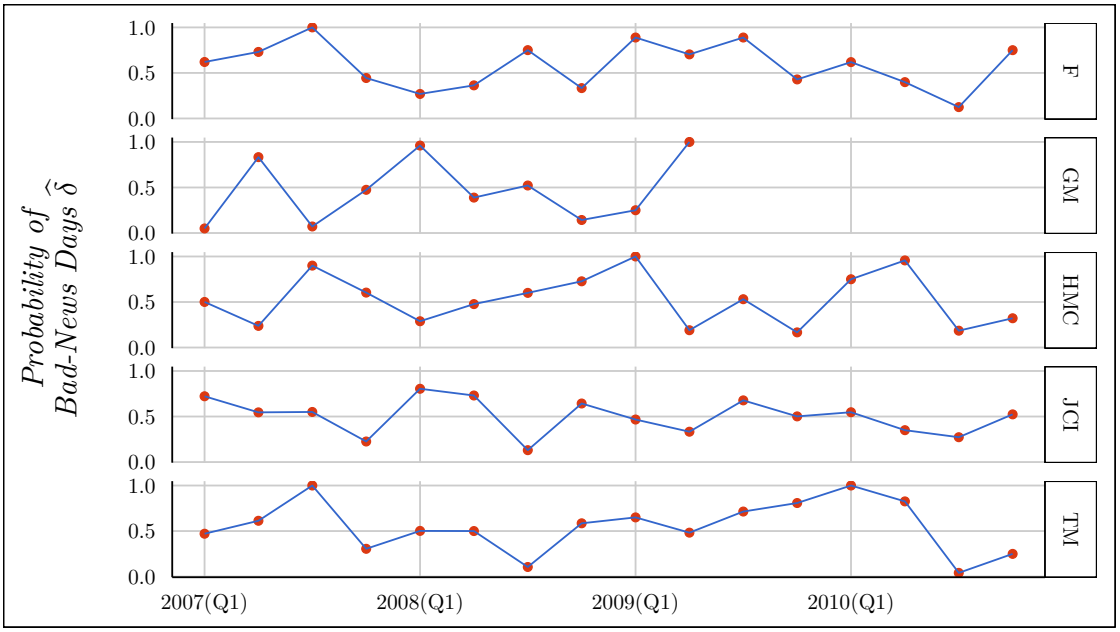

Figure 10.3: Quarterly estimates of the probability of bad-news trading days δ for all NYSE equities. Detailed results can be found in table 10.2.

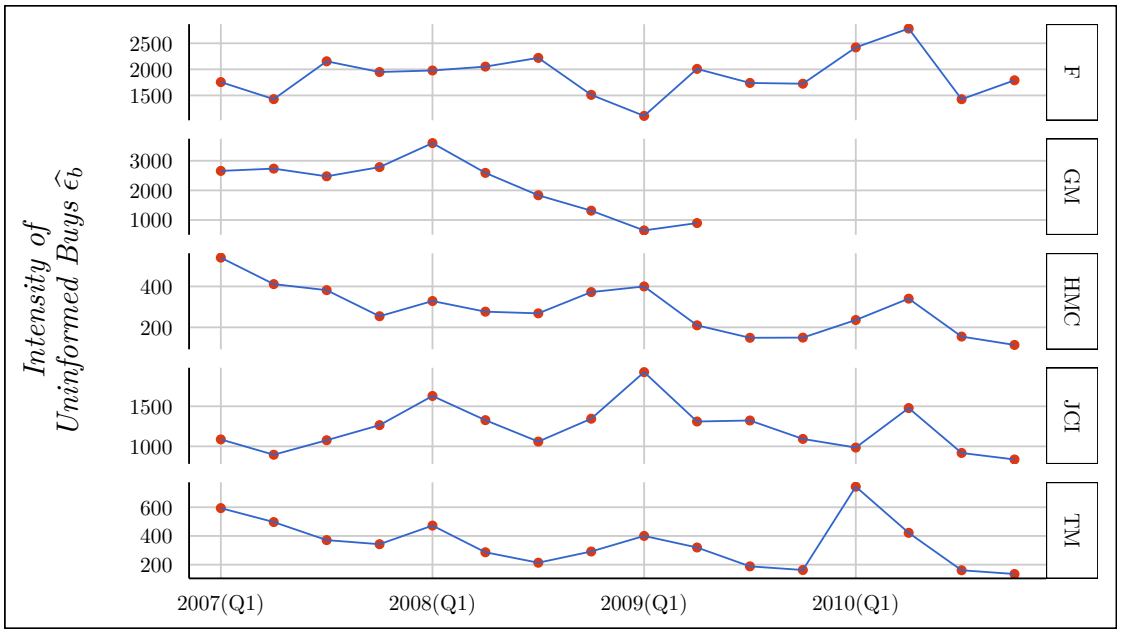

Figure 10.4: Quarterly estimates of the intensity of uninformed buys ϵ_b for all NYSE equities. Detailed results can be found in table 10.3.

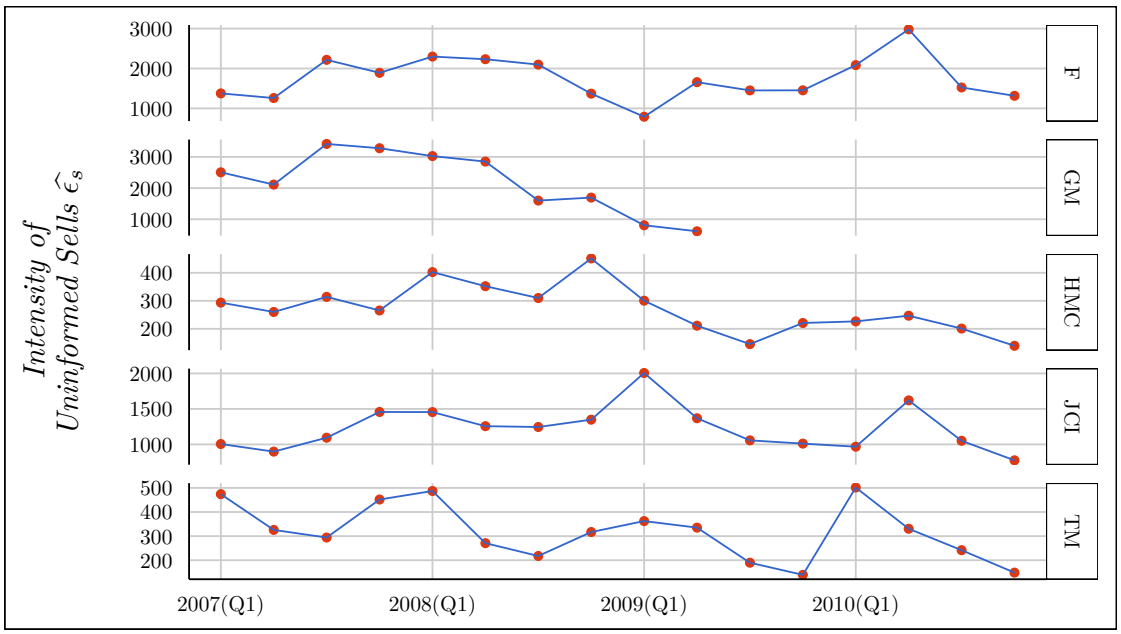

Figure 10.5: Quarterly estimates of the intensity of uninformed sells ϵ_s for all NYSE equities. Detailed results can be found in table 10.4.

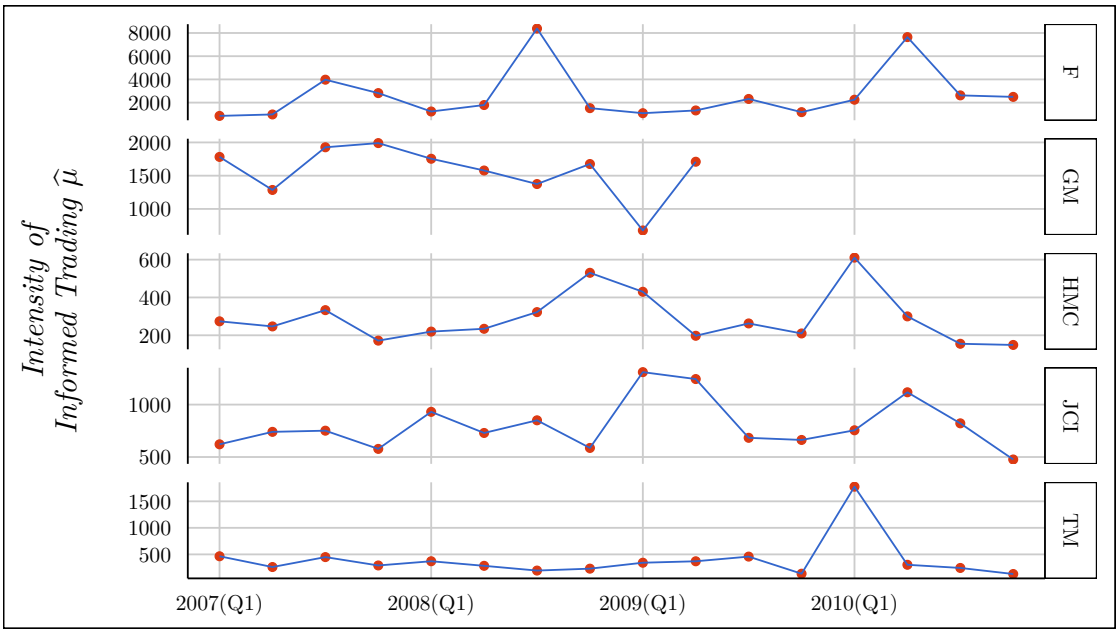

Figure 10.6: Quarterly estimates of the intensity of informed trading μ for all NYSE equities. Detailed results can be found in table 10.5.

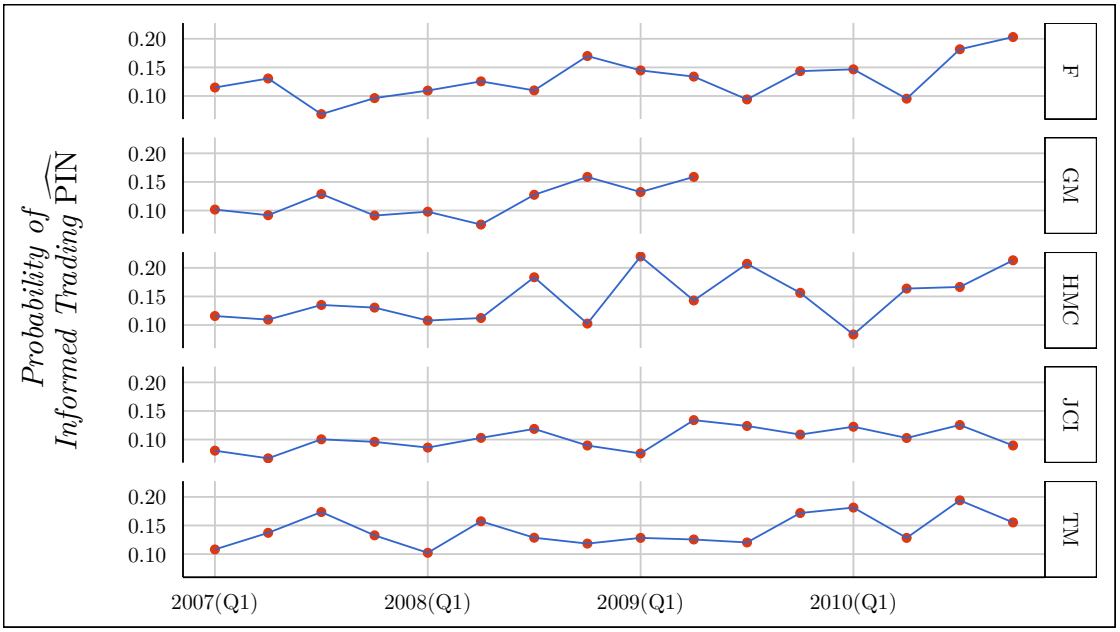

Figure 10.7: Quarterly estimates of the probability of informed trading PIN for all NYSE equities. Detailed results can be found in table 10.6.

10.2 Estimation Results (Xetra)

The estimates of α for equities traded on Xetra have a similar range like the US stocks. The values reach nearly 55% for BMW in the second quarter of 2008, whereas $\widehat{\alpha}$ is as low as 11% for VOW in the first quarter of 2008. Mostly, small ups and downs dictate the behavior of the plots for the estimates of the probability of an information event for all German stocks. However, there are outstanding high jumps for the estimates of α in the first half of 2008 for BMW and during 2010 for VOW. Furthermore, the graph of CON displays a period with an increased proportion of information events from the end of 2007 to the first half of 2008 compared with the remaining values of $\widehat{\alpha}$ for this equity.

Likewise to the NYSE equities, the estimates of the conditional probability of bad-news trading days exhibit several boundary solutions. Hence, the EHO model offers a (pseudo-)sureness about the direction of private information for specific quarters of the underlying data. Each of the manufacturers, BMW, DAI and VOW, has at least one value of $\hat{\delta}$ which equals either the lower or upper limit, while the estimates for CON never hit 0 and 1. The estimates of δ for CON lie in an interval from about 33% to 88%. According to the EHO model, all insider information in the fourth quarter of 2008 for the VOW stock have negative direction and attract solely informed sellers to be active on the market. Due to the price bubble in autumn of 2008, VOW is somewhat special compared to the other German equities. At the end of the section, we will analyze and explain the estimation results of all model parameters for the corresponding periods.

Both noise trading estimates $\widehat{\epsilon_b}$ and $\widehat{\epsilon_s}$ exhibit a positive trend for all manufacturers in the first two years, while an overall decline is observable in the last two years. Estimates for the supplier, CON, steadily increase in 2007 with the overall maximum at the beginning of 2008, followed by a decrease of both intensities of uninformed trading over the remaining time range. Highest values of uninformed trading activities for the manufacturers can be found in the second half of 2008. Furthermore, plots for BMW, DAI and VOW have in common that a positive jump is displayed from the end of 2007 to the first quarter of 2008 followed by a decline of $\widehat{\epsilon_b}$ and $\widehat{\epsilon_s}$.

The highest level of insider trading is visible in the second half of 2008 for all stocks. Intensity of informed trading for VOW even exceeds 10,000 transactions in the last quarter and reaches almost 7000 in the third. It is more than eight times and five times higher compared to the second quarter of the year, respectively. In the first quarter of 2009 the estimated insider intensity decreases by about 9000, compared to the fourth quarter of 2008. The supplier, CON, has the biggest amount of transactions initiated by insiders in the third quarter of 2008 followed by a huge decline. For the remaining quarters the estimated intensity of informed trading $\hat{\mu}$ resides on a *typical* level.

There is not much movement in the plots of the estimates of the probability of informed trading for BMW, CON and DAI. The difference between the minimum and maximum values is less than 8% for all three stocks, but we see little upward movements at the end of 2008. The high values of the PIN estimates in the second half of 2008 may be induced (to a high degree)

by insiders harnessing their advantage in information. But, similar to the GM equity, as explained in the previous section, the high estimate for the intensity of informed trading may be caused by a feedback of noise traders to the increased market activities due to prominent events. For instance, we can think of the German scrappage program for old cars. This program was introduced on January 13, 2009.

The graph of the PIN estimates for VOW stands out from the other plots and is a real eye-catcher. The fourth quarter of 2008 displays the maximum value for estimate of the probability of informed trading, which is higher than 20%. The other three German stocks do not show values for $\widehat{\text{PIN}}$ that are higher than 15%. As for the other stocks, the scrappage program may play an important role for the increased estimates but there were some special trading days during the fourth quarter of 2008 which probably add another impact. In October 2008 the price for VOW shares literally exploded and went beyond 1000 EUR as shown in figure 10.8.

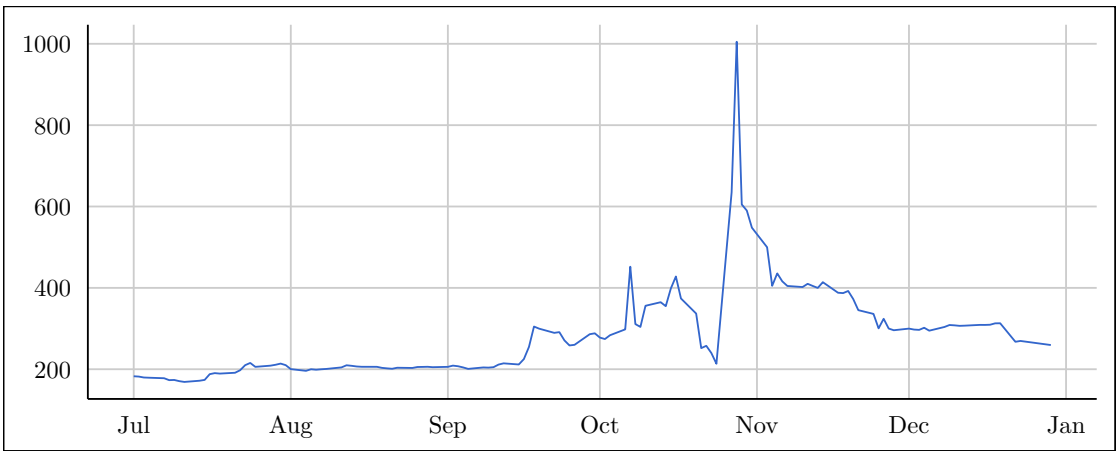

Figure 10.8: Daily high prices (in EUR) of the VOW stock in the third and fourth quarter of 2008.

Only a small amount of shares were available on the market and the majority of short sellers were obliged to buy shares in order to fulfill their contracts. Porsche Holding announced to hold more than 74% of VOW options and shares. The state of Niedersachsen also had a holding in VOW with more than 20%. Due to this market situation the price dramatically increased. After few trading days the situation on the market relaxed and the price dropped down. The probability of an information event is nearly twice as high as the estimate for the preceding quarter. The static EHO model constitutes a rate of trading days with private information in the fourth quarter of 25%. One might interpret the value of the estimate in a way that information events could be concentrated around the price-bubble in the fourth quarter.

We see an overall increase in trading activities in the second half of the year for VOW. The estimates of the intensities of noise trading ($\widehat{\epsilon_b} + \widehat{\epsilon_s}$) and information-based trading are on a similar level in the third quarter but differ by more than 1500 expected transactions per day in the fourth quarter. Hence, in the third quarter, on information events the market is split in nearly equal parts for noise traders and insiders, whereas the transactions initiated by informed traders clearly dominate those special trading days at the end of the year. The parameters of

the EHO model react to the few extreme trading days in October 2008 and the substantially increased trading activity in this time range.[115] As mentioned in the beginning of chapter 6, one may argue if the increased trading activity on such special trading days should be completely assigned to the group of informed traders. There might be a significant number of (wrongly labeled) *informed* transactions initiated by uninformed traders responding to the higher market activities.

Similar to the empirical results for the US symbols, the parameter estimates for the German equities exhibit that the identification problem discussed in section 4.3.4 is not present for the underlying data in the years from 2007 to 2010.

[115] The initiator for the extreme rush in the price for VOW equities was an announcement from Porsche Holding in which they published to hold 74% of VOW shares and options. However, Porsche Holding repudiated cases against them to deliberately influence the market price.

Quarter	BMW	CON	DAI	VOW
2007(Q1)	0.29673	0.37500	0.31251	0.40625
	(0.05712)	(0.06052)	(0.05794)	(0.06139)
2007(Q2)	0.27846	0.26229	0.29503	0.45941
	(0.05738)	(0.05632)	(0.05839)	(0.06386)
2007(Q3)	0.24615	0.36896	0.18462	0.30768
	(0.05343)	(0.05989)	(0.04812)	(0.05725)
2007(Q4)	0.27852	0.50815	0.37705	0.32777
	(0.05742)	(0.06402)	(0.06205)	(0.06010)
2008(Q1)	0.25807	0.43539	0.41933	0.11293
	(0.05557)	(0.06298)	(0.06267)	(0.04020)
2008(Q2)	0.54692	0.34374	0.42187	0.31250
	(0.06224)	(0.05937)	(0.06173)	(0.05794)
2008(Q3)	0.25721	0.21210	0.34867	0.13639
	(0.05378)	(0.05032)	(0.05867)	(0.04225)
2008(Q4)	0.42619	0.31148	0.27951	0.24590
	(0.06332)	(0.05929)	(0.05751)	(0.05513)
2009(Q1)	0.22222	0.24537	0.39682	0.22222
	(0.05238)	(0.05519)	(0.06164)	(0.05238)
2009(Q2)	0.25789	0.28342	0.30635	0.29043
	(0.05560)	(0.05820)	(0.05854)	(0.05766)
2009(Q3)	0.37849	0.22221	0.36350	0.12133
	(0.05975)	(0.05238)	(0.05920)	(0.04020)
2009(Q4)	0.22580	0.33804	0.30645	0.25807
	(0.05310)	(0.06176)	(0.05855)	(0.05557)
2010(Q1)	0.35125	0.24195	0.22224	0.24192
	(0.06107)	(0.05439)	(0.05238)	(0.05439)
2010(Q2)	0.30157	0.32258	0.42859	0.40935
	(0.05782)	(0.05937)	(0.06235)	(0.06277)
2010(Q3)	0.30304	0.30396	0.33370	0.12697
	(0.05657)	(0.05678)	(0.05810)	(0.04194)
2010(Q4)	0.36507	0.43549	0.38096	0.36508
	(0.06066)	(0.06297)	(0.06118)	(0.06066)

Table 10.7: Quarterly estimates of the probability of an information event α for all Xetra equities. Figures in parentheses denote standard errors.

Quarter	BMW	CON	DAI	VOW
2007(Q1)	0.15760	0.66660	0.00000	1.00000
	(0.08371)	(0.09624)	(– – –)	(– – –)
2007(Q2)	0.64662	0.49982	0.83333	0.24925
	(0.11598)	(0.12500)	(0.08784)	(0.08167)
2007(Q3)	0.18751	0.87490	0.74999	0.64999
	(0.09758)	(0.06756)	(0.12500)	(0.10666)
2007(Q4)	0.64685	0.41929	0.47826	0.45022
	(0.11597)	(0.08864)	(0.10416)	(0.11125)
2008(Q1)	0.12500	0.74091	0.07881	0.71447
	(0.08267)	(0.08436)	(0.07245)	(0.17068)
2008(Q2)	0.59994	0.72727	0.11112	0.35002
	(0.08282)	(0.09495)	(0.06048)	(0.10666)
2008(Q3)	0.48540	0.49978	0.18865	0.41068
	(0.12347)	(0.13363)	(0.10654)	(0.16198)
2008(Q4)	0.03855	0.57895	0.04143	1.00000
	(0.03778)	(0.11327)	(0.23248)	(– – –)
2009(Q1)	1.00000	0.33372	0.36009	0.50001
	(– – –)	(0.12199)	(0.09603)	(0.13363)
2009(Q2)	0.18769	0.64678	0.57955	0.72285
	(0.09767)	(0.11595)	(0.11322)	(0.10542)
2009(Q3)	0.43953	0.71418	0.20851	0.37473
	(0.09937)	(0.12077)	(0.08297)	(0.17111)
2009(Q4)	0.92860	0.49870	0.94729	0.56250
	(0.06879)	(0.11217)	(0.05128)	(0.12402)
2010(Q1)	0.40534	0.39996	0.07964	0.66652
	(0.10511)	(0.12649)	(0.03234)	(0.12174)
2010(Q2)	0.42101	0.39991	0.11100	0.53706
	(0.11327)	(0.10954)	(0.06043)	(0.09877)
2010(Q3)	0.24970	0.70081	0.68215	0.15719
	(0.09688)	(0.10235)	(0.09925)	(0.29798)
2010(Q4)	0.43485	0.70372	0.16667	0.30434
	(0.10337)	(0.08787)	(0.07607)	(0.09594)

Table 10.8: Quarterly estimates of the conditional probability of bad-news trading days δ for all Xetra equities. Figures in parentheses denote standard errors.

Quarter	BMW	CON	DAI	VOW
2007(Q1)	1150.58402	1195.50901	1963.83666	1901.81209
	(4.86234)	(4.53615)	(6.67149)	(5.45203)
2007(Q2)	1466.63894	1242.45243	3099.99536	1363.26855
	(5.05215)	(4.71264)	(7.24815)	(5.74996)
2007(Q3)	1547.11981	1572.36810	3659.10653	1874.55481
	(5.38595)	(4.98559)	(7.60279)	(5.61022)
2007(Q4)	1619.37842	1648.45981	3399.72551	2473.44370
	(5.31427)	(5.89961)	(8.02659)	(6.86278)
2008(Q1)	2104.51895	2560.44281	3700.31637	3231.12794
	(6.56666)	(6.72539)	(9.69125)	(7.29467)
2008(Q2)	1844.16401	2034.19871	2934.17377	2198.75308
	(5.81834)	(5.77919)	(8.58078)	(6.41259)
2008(Q3)	2140.88127	1731.23810	3481.43221	3689.21124
	(6.32036)	(5.38188)	(8.72623)	(7.60977)
2008(Q4)	2522.38788	725.09892	4903.30094	5938.83366
	(8.29353)	(3.63763)	(10.78575)	(9.71580)
2009(Q1)	2411.09528	274.07273	3744.28396	2061.04583
	(6.22058)	(2.39679)	(8.76841)	(5.95062)
2009(Q2)	1839.42664	275.39308	4031.13118	1821.12269
	(6.30987)	(2.24379)	(8.45419)	(5.57317)
2009(Q3)	1508.71394	383.01481	2962.78116	1795.63137
	(5.20720)	(2.53710)	(7.84003)	(5.39842)
2009(Q4)	1684.69745	345.45881	3243.07563	1079.46554
	(5.23166)	(2.59299)	(7.22744)	(4.32529)
2010(Q1)	1489.20194	839.11047	2925.91429	774.67248
	(7.24223)	(3.90427)	(7.27856)	(3.62626)
2010(Q2)	1954.54617	871.61437	3459.08105	512.35129
	(5.99874)	(4.09330)	(9.24810)	(3.34588)
2010(Q3)	2087.47671	726.84310	3213.16944	492.52758
	(6.51291)	(3.43198)	(7.22161)	(2.99246)
2010(Q4)	2300.16506	775.96509	2596.85614	681.09310
	(6.49458)	(3.66818)	(7.53856)	(3.76897)

Table 10.9: Quarterly estimates of the intensity of uninformed buys ϵ_b for all Xetra equities. Figures in parentheses denote standard errors.

Quarter	BMW	CON	DAI	VOW
2007(Q1)	1425.89042	1250.51022	3119.91896	1786.29514
	(4.83695)	(4.88089)	(6.96699)	(6.88914)
2007(Q2)	1524.21544	1320.30810	3242.16532	1784.40577
	(5.41547)	(4.90113)	(8.34842)	(5.54598)
2007(Q3)	1765.97021	1235.27458	3548.74900	1867.13792
	(5.31515)	(5.47038)	(7.80413)	(5.81235)
2007(Q4)	1524.44608	1693.84204	3607.90630	2474.17041
	(5.52509)	(5.69858)	(8.22655)	(6.78852)
2008(Q1)	2457.76592	2291.18681	4989.94410	3392.47654
	(6.40180)	(7.05764)	(8.94157)	(7.75434)
2008(Q2)	1762.58870	2026.97691	3407.15809	2620.86169
	(6.02543)	(6.30920)	(7.41533)	(6.66779)
2008(Q3)	2311.29587	1989.61588	4033.98940	3802.75368
	(5.98652)	(5.67032)	(7.89551)	(7.82079)
2008(Q4)	3049.03441	644.57493	5319.15758	3345.36829
	(7.08948)	(3.49765)	(9.29154)	(8.24735)
2009(Q1)	2119.00408	301.06927	3566.37948	2091.59118
	(6.57839)	(2.29342)	(7.96642)	(6.01328)
2009(Q2)	2151.97345	246.41769	3825.23144	1805.04650
	(5.99998)	(2.19266)	(8.46979)	(5.95581)
2009(Q3)	1576.16594	311.04080	3331.92094	1759.17933
	(5.38472)	(2.38205)	(7.25059)	(5.25084)
2009(Q4)	1461.68265	326.65977	2786.59724	896.09122
	(5.45286)	(2.73483)	(7.95341)	(4.05777)
2010(Q1)	1512.25807	857.54993	3005.95297	662.29066
	(5.41197)	(3.87159)	(7.13708)	(3.52908)
2010(Q2)	2071.80233	968.05731	4285.95452	428.73101
	(5.97258)	(4.11162)	(8.41179)	(3.34486)
2010(Q3)	2259.55548	643.14294	3073.26115	411.88895
	(6.20251)	(3.65935)	(7.88882)	(2.55705)
2010(Q4)	2292.63471	692.06529	3071.80191	576.68529
	(6.35746)	(3.93443)	(7.06324)	(3.12367)

Table 10.10: Quarterly estimates of the intensity of uninformed sells ϵ_s for all Xetra equities. Figures in parentheses denote standard errors.

Quarter	BMW	CON	DAI	VOW
2007(Q1)	760.79904	684.89650	2223.36113	2127.42367
	(11.10619)	(9.62753)	(16.10280)	(13.98460)
2007(Q2)	800.53575	724.85032	2265.69484	615.77826
	(12.40576)	(11.72447)	(18.62717)	(9.79156)
2007(Q3)	1530.07151	1099.29982	3052.03198	1742.26206
	(15.03609)	(11.18050)	(23.20216)	(14.03581)
2007(Q4)	931.57364	887.57015	1530.05755	1766.39756
	(13.10711)	(10.00217)	(16.21947)	(15.51254)
2008(Q1)	1591.71208	1219.79839	2556.23491	4594.82492
	(16.34480)	(12.70866)	(17.39867)	(35.71557)
2008(Q2)	690.84775	1131.89657	1596.57836	1306.51885
	(9.55687)	(12.75756)	(15.46976)	(14.89345)
2008(Q3)	1887.87198	2719.20597	2626.35500	6813.46469
	(16.39304)	(17.56017)	(18.79196)	(29.73289)
2008(Q4)	1899.38560	683.20435	5936.57765	10843.96131
	(15.33918)	(8.85845)	(27.07387)	(22.26821)
2009(Q1)	1479.93853	359.27069	1538.77213	1865.92947
	(17.65543)	(7.13004)	(16.00921)	(17.15499)
2009(Q2)	1529.95657	273.16866	1728.36051	1210.49999
	(16.42825)	(5.86284)	(19.13381)	(13.99583)
2009(Q3)	824.81622	479.72979	1224.88820	2684.16709
	(10.66151)	(7.85901)	(15.14728)	(24.25545)
2009(Q4)	1315.32420	341.25411	1893.24550	897.65631
	(15.06354)	(6.45705)	(17.29990)	(11.13843)
2010(Q1)	674.61155	1069.88787	2137.90253	737.35675
	(13.33752)	(11.57553)	(19.78107)	(10.14171)
2010(Q2)	1227.50881	626.27083	1916.68847	322.88099
	(13.90898)	(9.26877)	(16.74539)	(6.76448)
2010(Q3)	1293.83291	442.54045	1258.85128	1838.72596
	(13.85829)	(8.35876)	(15.79756)	(17.36135)
2010(Q4)	1184.03542	565.11344	1401.90015	729.08785
	(13.01905)	(7.56520)	(14.28604)	(8.26269)

Table 10.11: Quarterly estimates of the intensity of informed trading μ for all Xetra equities. Figures in parentheses denote standard errors.

Quarter	BMW	CON	DAI	VOW
2007(Q1)	0.08056	0.09502	0.12024	0.18985
2007(Q2)	0.06936	0.06906	0.09535	0.08246
2007(Q3)	0.10208	0.12623	0.07250	0.12531
2007(Q4)	0.07624	0.11890	0.07606	0.10476
2008(Q1)	0.08260	0.09866	0.10980	0.07265
2008(Q2)	0.09482	0.08743	0.09602	0.07810
2008(Q3)	0.09834	0.13420	0.10861	0.11035
2008(Q4)	0.12686	0.13447	0.13965	0.22312
2009(Q1)	0.06768	0.13290	0.07709	0.09079
2009(Q2)	0.08996	0.12920	0.06314	0.08838
2009(Q3)	0.09190	0.13314	0.06606	0.08392
2009(Q4)	0.08625	0.14649	0.08778	0.10495
2010(Q1)	0.07317	0.13238	0.07416	0.11043
2010(Q2)	0.08420	0.09895	0.09589	0.12315
2010(Q3)	0.08273	0.08941	0.06264	0.20518
2010(Q4)	0.08602	0.14357	0.08610	0.17466

Table 10.12: Quarterly estimates of the probability of informed trading PIN for all Xetra equities.

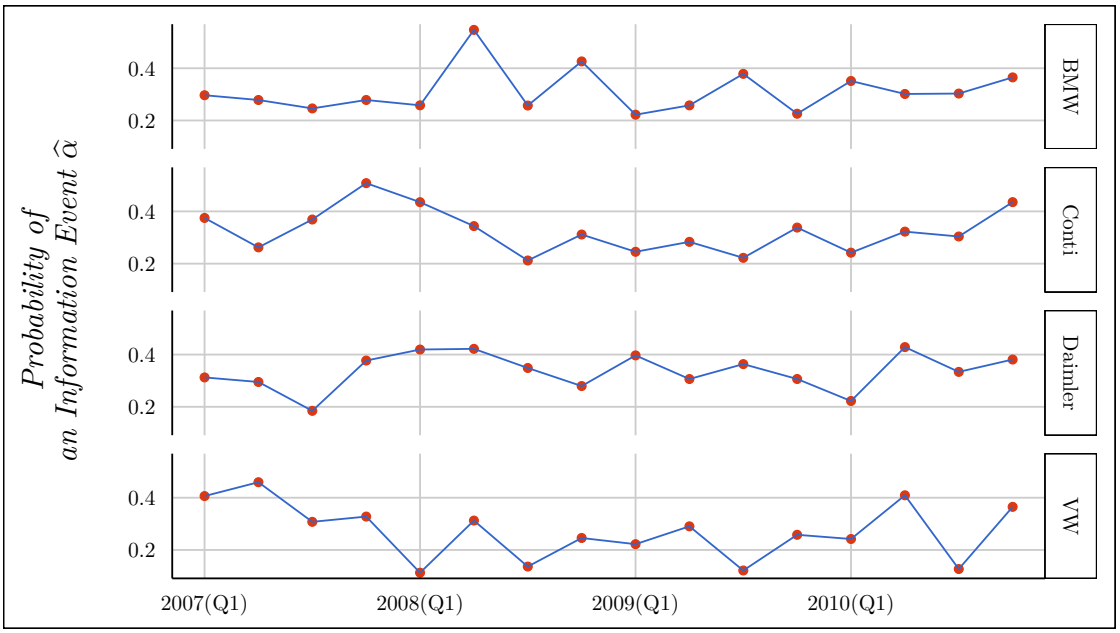

Figure 10.9: Quarterly estimates of the probability of an information event α for all Xetra equities. Detailed results can be found in table 10.7.

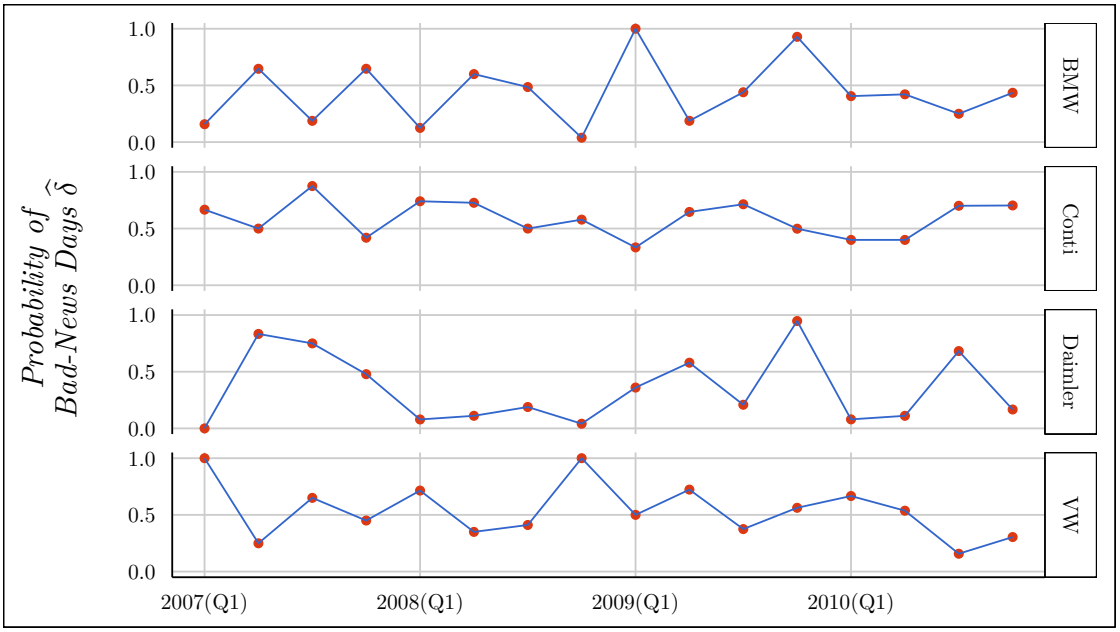

Figure 10.10: Quarterly estimates of the probability of bad-news trading days δ for all Xetra equities. Detailed results can be found in table 10.8.

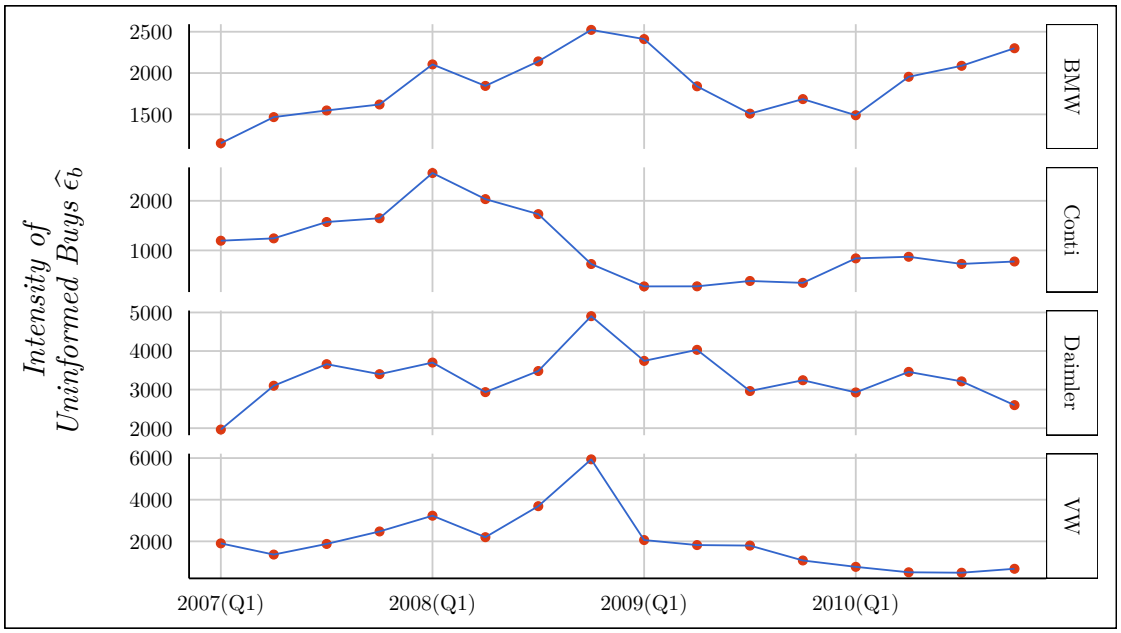

Figure 10.11: Quarterly estimates of the intensity of uninformed buys ϵ_b for all Xetra equities. Detailed results can be found in table 10.9.

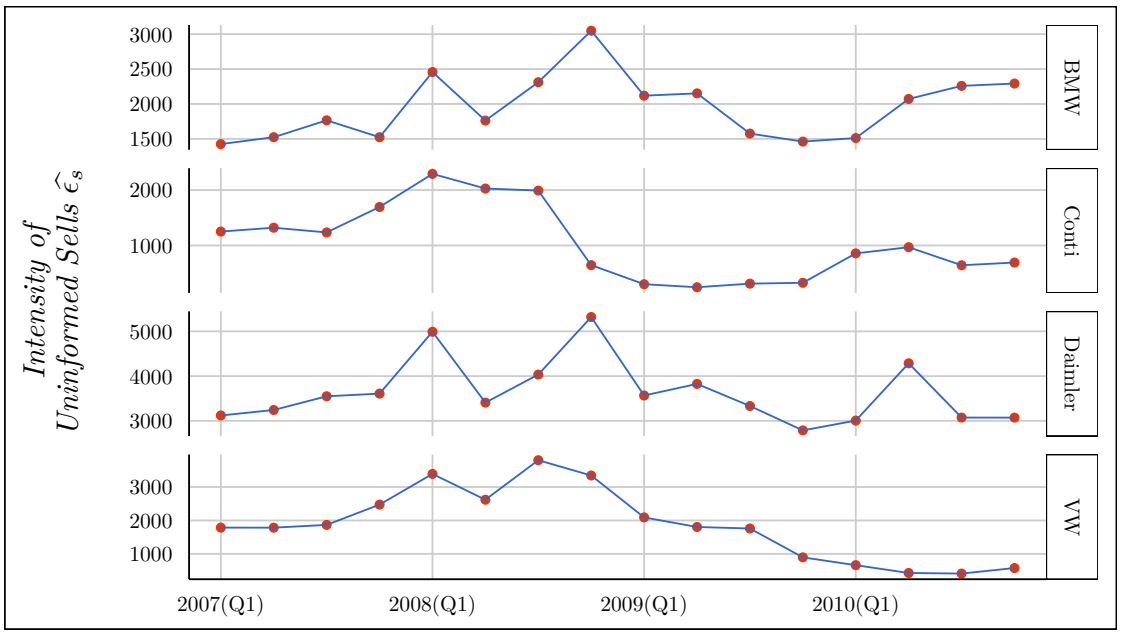

Figure 10.12: Quarterly estimates of the intensity of uninformed sells ϵ_s for all Xetra equities. Detailed results can be found in table 10.10.

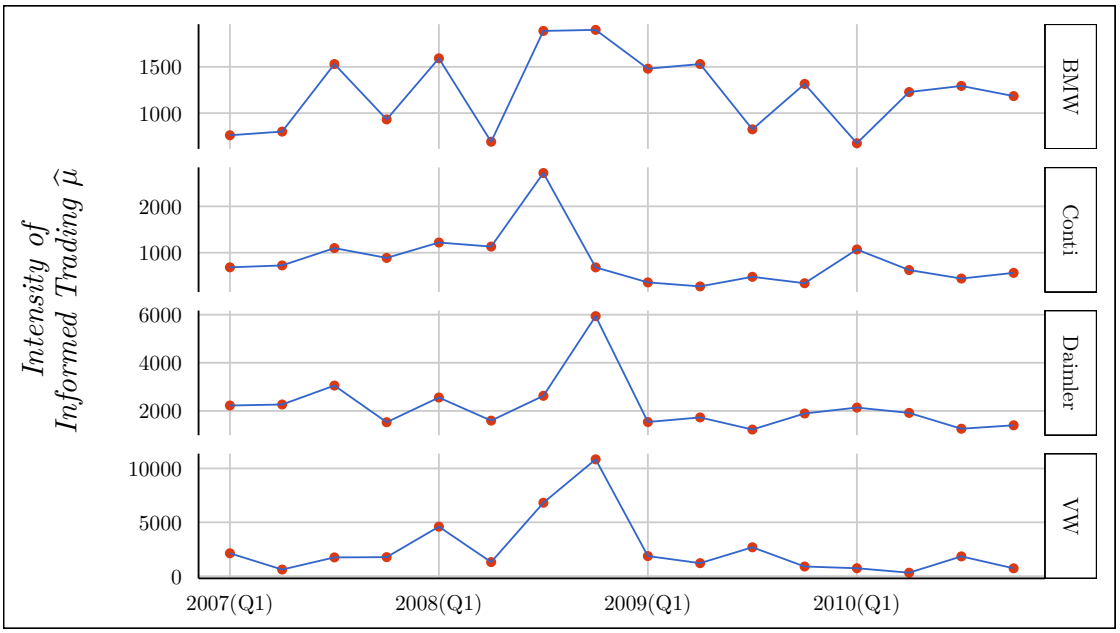

Figure 10.13: Quarterly estimates of the intensity of informed trading μ for all Xetra equities. Detailed results can be found in table 10.11.

Figure 10.14: Quarterly estimates of the probability of informed trading PIN for all Xetra equities. Detailed results can be found in table 10.12.

10.3 Posterior Probabilities and Tests for Independence of Trading Days' States

To reduce the amount of graphs, we will not display posterior probabilities for all nine equities and the total range of four years. Instead, we concentrate on the periods which were already emphasized in the previous sections, the second quarter of 2009 for GM (delisting) and the third and fourth quarter of 2008 for VOW (price temporarily over 1000 EUR).

The model parameter estimates for GM in the second quarter of 2009 can be interpreted that informed sellers are only active on few trading days but act with high intensity (see tables 10.1 and 10.2).

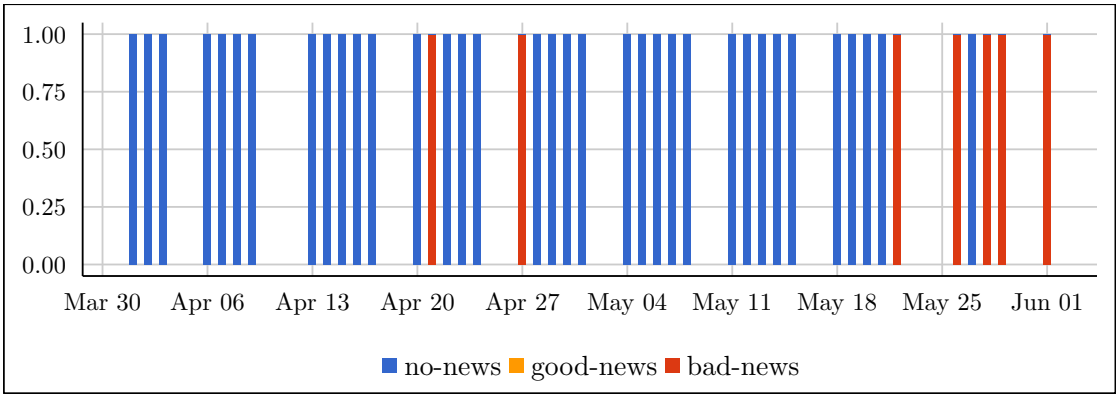

Figure 10.15: Posterior probabilities for conditions of trading days for GM in the last months before its delisting.

According to figure 10.15 two single days in the last half of April and five out of the last six trading days are bad-news days with probability 1. Hence, we see high concentration of days with negative private information shortly before delisting. Furthermore, posteriors are able to clearly classify each trading day as no-news or bad-news trading day.[116]

Figure 10.16 visualize posterior probabilities of trading days' conditions of VOW in the third and fourth quarter of 2008. We can see a clustering of bad-news days in the middle of September and a series of subsequent information events in the last week of October. EHO model classifies October 28, on which the price exceeded 1000 EUR, as bad-news day. One can think of insiders holding VOW shares which anticipate the peaks in the price series (see figure 10.8). Similar to GM, posterior probabilities can clearly identify the conditions of trading days.[117]

In figures 10.15 and 10.16 clustering of information events, either with positive or negative direction of news, is visible. Posterior probabilities for the remaining symbols and trading periods fortify this finding. We create dichotomous series of non-information and information events

[116] Since $\hat{\delta} = 1$, there are no good-news days.

[117] Likewise to the GM case, $\hat{\delta}$ lies on the upper boundary in the fourth quarter of 2008 and therefore only no-news and bad-news trading days occur.

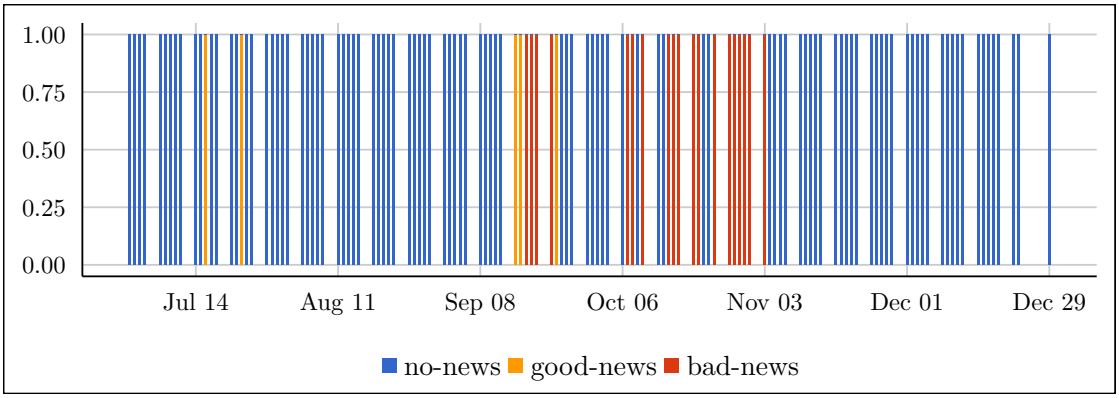

Figure 10.16: Posterior probabilities for conditions of trading days for VOW in the third and fourth quarter of 2008.

for each symbol and quarter. A *Run-Test* (e.g., see Wald and Wolfowitz 1940), which checks for randomness, is then conducted for each series. Almost 70% of the resulting p-values are smaller than 0.05 as shown in table 10.13. This indicates that the assumption of independence of trading days may not always be appropriate and should be relaxed to allow for dependencies. The PIN-HMM model, which is introduced in section 6.2, considers dependencies in the series of trading days' conditions by modeling them with hidden Markov chains.

	F	GM	HMC	JCI	TM	BMW	CON	DAI	VOW
2007(Q1)	0.91214	0.02060	0.03960	0.00908	0.24594	0.00117	0.01720	0.00001	0.00000
2007(Q2)	0.00573	0.01587	0.00419	0.06358	0.04621	0.03537	0.42224	0.09401	0.26409
2007(Q3)	0.00406	0.00034	0.00000	0.00002	0.00024	0.00200	0.00000	0.61851	0.00000
2007(Q4)	0.21019	0.00471	0.16858	0.51757	0.00101	0.41556	0.02827	0.01111	0.00031
2008(Q1)	0.01963	0.00031	0.02619	0.00215	0.06093	0.00733	0.24275	0.02432	0.00000
2008(Q2)	0.27832	0.78428	0.00203	0.00986	0.00001	0.45243	0.00235	0.04379	0.30344
2008(Q3)	0.56632	0.00511	0.19764	0.00003	0.04321	0.00007	0.00001	0.00637	0.00004
2008(Q4)	0.00204	0.00013	0.00679	0.48353	0.03688	0.00000	0.00177	0.00000	0.00000
2009(Q1)	0.02159	0.04321	0.00139	0.20156	0.15145	0.00399	0.00005	0.11592	0.00030
2009(Q2)	0.01056	0.00739	0.24518	0.00419	0.33300	0.00926	0.04042	0.10568	0.62917
2009(Q3)	0.07039		0.10622	0.38267	0.35192	0.10980	0.08575	0.22251	0.00031
2009(Q4)	0.65844		0.00316	0.00489	0.89556	0.00002	0.00557	0.00008	0.00755
2010(Q1)	0.00253		0.00017	0.00450	0.00021	0.03380	0.79440	0.16200	0.00489
2010(Q2)	0.00124		0.43926	0.40163	0.00009	0.00001	0.00755	0.00486	0.00039
2010(Q3)	0.01804		0.00649	0.00001	0.00048	0.06581	0.31777	0.35124	0.33198
2010(Q4)	0.00003		0.03004	0.02196	0.32983	0.00150	0.00016	0.01882	0.04805

Table 10.13: P-values of Run-Tests for each equity and quarter in our datasource. Values lower than 0.05 indicate that the null hypothesis of independence is rejected.

10.4 Confidence Intervals

In the results for our datasets, shown in tables 10.14 (NYSE) and 10.15 (Xetra), we see that the smallest range of confidence intervals for NYSE-listed stocks can be found for JCI in the fourth quarter of 2007 (0.043), whereas TM exhibits the widest range in the early months of 2010 (0.241). The average difference between lower and upper limit of the confidence intervals lies below 0.1 for the US stocks. However, the range seems to be driven by the intensity of transactions initiated by informed traders or moreover by the relation of information-based trading to noise trading. The more insiders dominate the market the wider are the ranges of PIN confidence intervals. This behavior is visualized in figure 10.17.

The confidence intervals of the German stocks traded on the electronic Xetra marketplace show similar properties. The mean range is also less than 0.1 for all stocks (e.g., for BMW and DAI it is about 0.06). The confidence interval for BMW in the second quarter of 2008 shows the smallest difference between lower and upper limit (0.038), whereas the range is highest in the fourth quarter of 2008 for VOW (0.28). Likewise to the symbols listed on NYSE, high levels of information-based trading compared to the intensities of liquidity traders span the confidence intervals wider, as displayed in figure 10.18.

	F	GM	HMC	JCI	TM
2007(Q1)	[0.087, 0.141]	[0.059, 0.134]	[0.083, 0.147]	[0.052, 0.109]	[0.067, 0.147]
2007(Q2)	[0.096, 0.163]	[0.065, 0.117]	[0.075, 0.143]	[0.033, 0.101]	[0.106, 0.166]
2007(Q3)	[0.006, 0.126]	[0.095, 0.159]	[0.088, 0.177]	[0.068, 0.132]	[0.026, 0.224]
2007(Q4)	[0.045, 0.151]	[0.060, 0.122]	[0.098, 0.162]	[0.074, 0.117]	[0.097, 0.166]
2008(Q1)	[0.081, 0.137]	[0.063, 0.124]	[0.076, 0.137]	[0.056, 0.115]	[0.065, 0.138]
2008(Q2)	[0.089, 0.163]	[0.048, 0.102]	[0.077, 0.147]	[0.075, 0.129]	[0.112, 0.200]
2008(Q3)	[0.003, 0.198]	[0.091, 0.163]	[0.135, 0.228]	[0.082, 0.152]	[0.082, 0.169]
2008(Q4)	[0.125, 0.213]	[0.111, 0.203]	[0.057, 0.151]	[0.066, 0.112]	[0.082, 0.153]
2009(Q1)	[0.093, 0.191]	[0.090, 0.173]	[0.039, 0.264]	[0.046, 0.106]	[0.087, 0.167]
2009(Q2)	[0.099, 0.167]	[0.019, 0.243]	[0.097, 0.183]	[0.093, 0.174]	[0.078, 0.172]
2009(Q3)	[0.035, 0.147]		[0.137, 0.271]	[0.095, 0.150]	[0.038, 0.192]
2009(Q4)	[0.107, 0.177]		[0.104, 0.205]	[0.078, 0.138]	[0.131, 0.209]
2010(Q1)	[0.103, 0.191]		[0.011, 0.156]	[0.086, 0.158]	[0.019, 0.260]
2010(Q2)	[0.008, 0.173]		[0.042, 0.206]	[0.069, 0.136]	[0.090, 0.164]
2010(Q3)	[0.111, 0.242]		[0.125, 0.204]	[0.088, 0.162]	[0.066, 0.240]
2010(Q4)	[0.142, 0.258]		[0.161, 0.261]	[0.061, 0.118]	[0.109, 0.198]

Table 10.14: Quarterly 95% confidence intervals for the probability of informed trading PIN for all NYSE equities.

	BMW	CON	DAI	VOW
2007(Q1)	[0.052, 0.108]	[0.068, 0.122]	[0.012, 0.157]	[0.019, 0.234]
2007(Q2)	[0.043, 0.096]	[0.043, 0.097]	[0.060, 0.129]	[0.062, 0.103]
2007(Q3)	[0.064, 0.140]	[0.089, 0.161]	[0.038, 0.106]	[0.085, 0.164]
2007(Q4)	[0.047, 0.104]	[0.092, 0.144]	[0.054, 0.098]	[0.071, 0.137]
2008(Q1)	[0.048, 0.115]	[0.073, 0.124]	[0.079, 0.138]	[0.022, 0.119]
2008(Q2)	[0.075, 0.113]	[0.060, 0.114]	[0.070, 0.121]	[0.052, 0.104]
2008(Q3)	[0.061, 0.134]	[0.081, 0.188]	[0.077, 0.140]	[0.052, 0.171]
2008(Q4)	[0.081, 0.158]	[0.089, 0.177]	[0.079, 0.186]	[0.016, 0.296]
2009(Q1)	[0.017, 0.097]	[0.083, 0.182]	[0.056, 0.099]	[0.054, 0.130]
2009(Q2)	[0.054, 0.125]	[0.082, 0.174]	[0.041, 0.085]	[0.056, 0.119]
2009(Q3)	[0.065, 0.117]	[0.080, 0.186]	[0.046, 0.086]	[0.033, 0.137]
2009(Q4)	[0.045, 0.120]	[0.099, 0.190]	[0.052, 0.117]	[0.065, 0.143]
2010(Q1)	[0.050, 0.096]	[0.083, 0.182]	[0.039, 0.106]	[0.068, 0.154]
2010(Q2)	[0.055, 0.113]	[0.066, 0.131]	[0.070, 0.120]	[0.090, 0.154]
2010(Q3)	[0.055, 0.111]	[0.059, 0.119]	[0.043, 0.082]	[0.060, 0.299]
2010(Q4)	[0.060, 0.111]	[0.107, 0.178]	[0.062, 0.111]	[0.126, 0.220]

Table 10.15: Quarterly 95% confidence intervals for the probability of informed trading PIN for all Xetra equities.

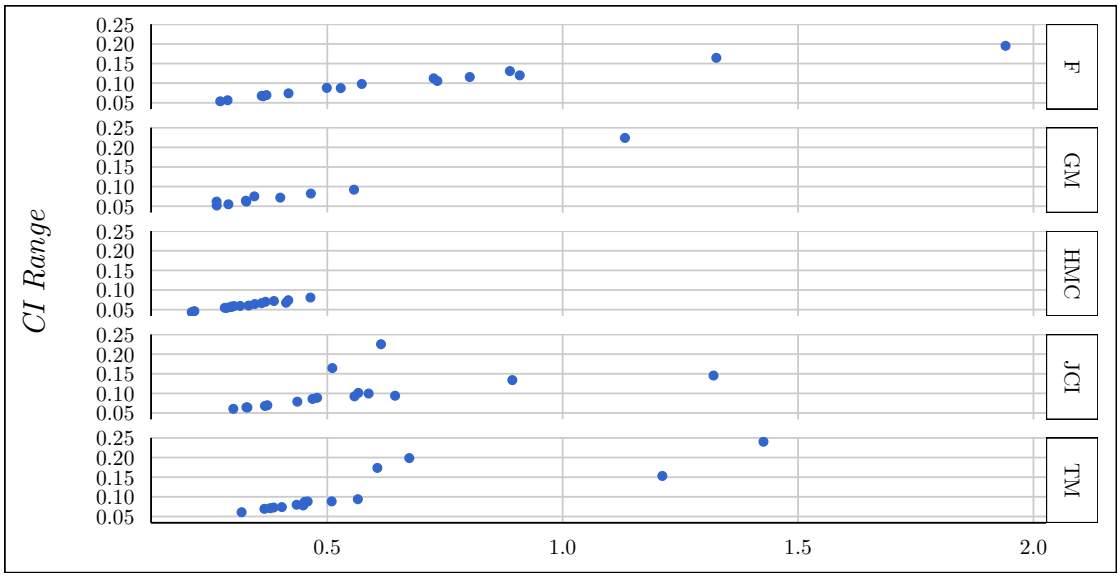

Figure 10.17: Relation of information-based trading to noise trading plotted against the range of corresponding confidence intervals for the probability of informed trading of US equities.

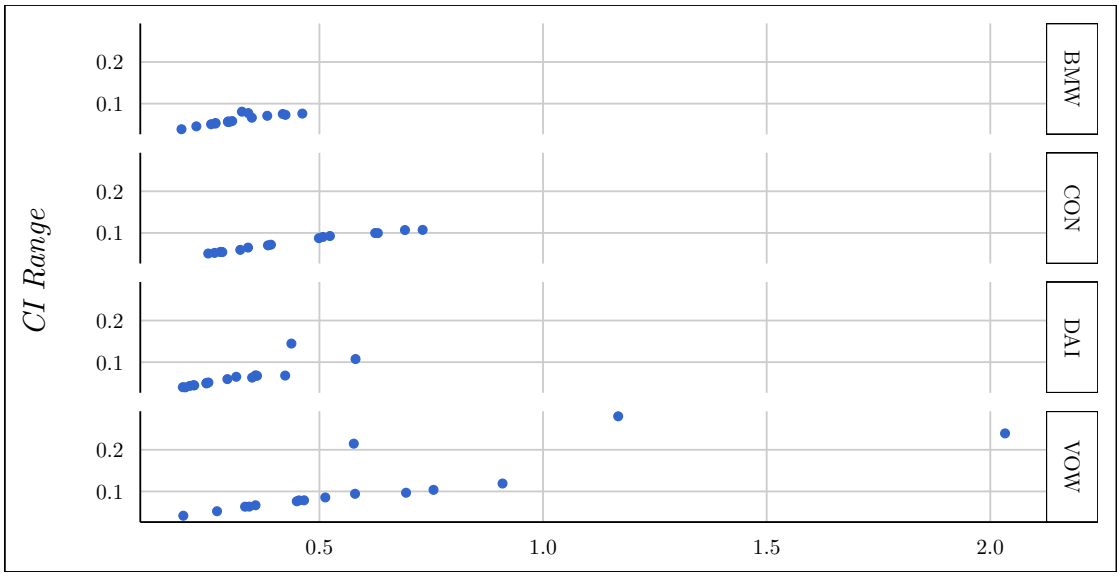

Figure 10.18: Relation of information-based trading to noise trading plotted against the range of corresponding confidence intervals for the probability of informed trading of German equities.

11 Empirical Applications (Dynamic Models)

We apply the previously presented PIN-HMM and PIN-ALACD models to the same equities mentioned in the previous chapter. Hence, we have a total of nine symbols listed on two different marketplaces.[118] For optimization purposes we split datasets by years.[119] By calculating confidence intervals for model parameters in every year and inspecting the intersection thereof, we see that the null hypothesis of constant parameters over the complete time span of four years does not hold.[120]

Initially, probabilities of informed trading and conditions of trading periods are calculated on a daily basis. However, we also offer intraday estimates for five minute intervals in section 11.5. In contrast to the EHO and EKOP model no algorithms for generating *good* sets of initial values exist. Therefore we are forced to use random sets of starting values which are drawn form uniform distributions whose minimum and maximum values correspond to lower and upper bounds for model parameters we incorporate in optimization runs (see section 11.1).

In total, we create 50 sets of randomly drawn initial values[121] and therefore perform 50 optimization runs for each equity and year. Our estimation results show that it may be dangerous to utilize too few maximizations due to the potential finding of only local maxima. For each symbol, we take the set of estimates which yields the highest likelihood function value into account for further computations and analyses.

Before we begin with in-depth explanations of the estimation results for both dynamic approaches, we will shortly discuss some findings about the estimates of the shape parameters of Weibull distributions and the parameters which are responsible for adjusting for information-based trading in the PIN-HMM approach, which are displayed in tables 11.1 and 11.2. They fortify the assumptions of separate shape parameters for the conditional distributions of buys'

[118] A detailed description of the underlying datasource can be found in chapter 8.

[119] Estimating the probability of informed trading on a daily basis incorporating a shorter time range of the underlying data is possible. However, we are in line with the work by Tay, Ting, Tse, and Warachka (2009) in which datasets also covered trading days for one year.

[120] To save some space, we do not explicitly display the results in this work. However, they are available upon request.

[121] See code chunk 7.2 for the implementation of how sets of initial values are drawn from uniform distributions.

and sells' waiting times as well as unique parameters for the adjustment of insider trading triggered by either positive or negative signals.

In the working paper by Tay, Ting, Tse, and Warachka (2007) Weibull distributions with identical shape parameters for durations of buys and sells are investigated. Table 11.1 shows p-values which belong to t-tests for the significance of the difference of both trade directions' shape parameters. The vast majority of entries are lower than any commonly used significance level which means that the assumption of unique shape parameters in the PIN-HMM model is appropriate and improves the quality of results. Only 7 out of 35 p-values[122] are higher than the typically used significance level of 5%.

	2007	2008	2009	2010
F	0.50864	0.00000	0.00000	0.00000
GM	0.00000	0.00000	0.00000	– – –
HMC	0.00000	0.00000	0.00000	0.00001
JCI	0.00000	0.12229	0.10020	0.00166
TM	0.00000	0.23857	0.00117	0.02330
BMW	0.00000	0.00000	0.17982	0.27494
CON	0.00000	0.03218	0.00001	0.00000
DAI	0.09558	0.00000	0.00000	0.00000
VOW	0.00000	0.00000	0.00000	0.00580

Table 11.1: Table of p-values of t-tests with null hypotheses that the difference of shape parameters of buys' and sells' interarrival times' distributions in the PIN-HMM model is 0.

Our PIN-HMM model is also more flexible than PIN-ALACD in terms of adjusting for informed trading. While one parameter controls for both directions of private information in the latter, the former distinguishes between adjustment for positive and negative private news.[123]

Likewise to 11.1, most values in table 11.2 are very small and the null hypothesis of identical parameters adjusting for insider trading is rejected in those cases. Hence, the setting of the PIN-ALACD model seems not sufficient. Only in 2 out of 35 possible cases the restrictive assumption is acceptable if the significance level is assumed to be 5%. To summarize the results in both tables, we can see that the PIN-ALACD model may probably lack some flexibility if any recent dataset is utilized.

Similar to sections 10.1 and 10.2, which offer estimation results for the static model, we place corresponding tables of estimation results at the ends of sections 11.1 and 11.2.

[122] GM was not listed on NYSE in 2010.

[123] Tay, Ting, Tse, and Warachka (2009) mention this possibility to extend their model in a footnote but do not investigate it any further.

	2007	2008	2009	2010
F	0.00000	0.00000	0.00000	0.00000
GM	0.00000	0.00000	0.00000	– – –
HMC	0.00000	0.00000	0.00000	0.00000
JCI	0.00001	0.00000	0.00104	0.00000
TM	0.00000	0.00000	0.42350	0.00000
BMW	0.00001	0.00000	0.00000	0.00000
CON	0.00000	0.00719	0.00000	0.00000
DAI	0.00000	0.00000	0.99959	0.00000
VOW	0.00000	0.00000	0.00000	0.00000

Table 11.2: Table of p-values of t-tests with null hypotheses that the difference of parameters adjusting for informed trading in the PIN-HMM model is 0.

11.1 Estimation Results (NYSE)

In this section we will discuss and interpret the estimation results for the PIN-ALACD as well as the PIN-HMM model for the US equities in our datasource. Firstly, we mention some constraints we impose on model parameters or linear combinations thereof in optimization runs. Restrictions concerning ALACD specifications are valid for both dynamic approaches. Therefore imposed boundaries only differ in terms of parameters which are part of the modeling of probabilities of trading days' conditions and the additional shape parameters of Weibull distributions.

According to Tay, Ting, Tse, and Warachka (2009), we expect the parameters δ_2, δ_3 and δ_4, which are involved in the calculation of state probabilities in the PIN-ALACD model, to be positive. Hence, a high value for the aggregated total volume for trading day d in comparison with its average yields a low probability of no-news condition. Similar, values for the aggregated volume of buyer- or seller-initiated transactions which exceed their means induce higher probabilities of good- or bad-news, respectively. There is no positivity restriction for the intercept δ_1. The assumptions for δ_2, δ_3 and δ_4 translate to lower bounds of 0 in optimization.

All transition probabilities in the PIN-HMM model have lower and upper bounds of 0 and 1, respectively. In addition, each sum of $a_{q\mathcal{N}}$, $a_{q\mathcal{G}}$ and $a_{q\mathcal{B}}$ with $q \in Q$ must be identical to unity. Therefore one can easily build transition matrices for each equity and year from the estimation results in tables 11.7 – 11.10 by calculating the missing diagonal elements of the matrices. Due to brevity reasons, we will not explicitly display any transition matrix but will exemplarily reconstruct one later.

The constraints we impose on ALACD specifications concern the autoregressive parameters α_1 and α_{-1} and the coefficients of lagged observed durations β_1 and β_{-1}. To ensure covariance stationarity of the durations processes of buys and sells, sums of the parameters α_1 and β_1

as well as α_{-1} and β_{-1} need to be less than unity as stated in the work by Bauwens and Giot (2000).

Finally, we expect parameters adjusting for informed trading, τ in the PIN-ALACD model and τ_1 and τ_{-1} in the PIN-HMM model, to be positive. We can see from section 6.3 that these restrictions are necessary to ensure that the number of buys on good-news days and sells on bad-news days is at least as large as their counterparts on non-information events. Information events with a lower intensity of buys or sells, depending on the direction of private information, are not reasonable in the context of estimation of the probability of informed trading. The shape parameters of Weibull distributions in our PIN-HMM model, which also influence the scaling of the intensity of buys or sells on the corresponding information events, as shown in equation (6.70), are positive by definition.

A remarkable finding is the high number of boundary solutions for estimates of δ_3 and δ_4 in the PIN-ALACD model. Every symbol in our datasource exhibits at least one coefficient which equals 0.

Exemplarily, the coefficient of the comparison between aggregated volume of sells with its average equals 0 for F in 2007. Hence, aggregated volume of sells does not influence the decision between good- and bad-news condition, given an information event. This means that the aggregated transaction volume on day d of only one trade direction decides about two possible types of information events. If the aggregated volume of buys on trading day d is larger than its average the conditional probability of a good-news day is larger than 0.5, if it is lower the same holds for the conditional probability of a bad-news day. If the aggregated volume of buys is identical to its average both conditional probabilities equal 0.5.

Estimates of δ_3 and δ_4 lie both on the lower boundary for HMC in 2007 and 2008. Therefore, neither aggregated volume of buys nor aggregated volume of sells impacts in any way the probabilities of good- and bad-news states. They are identical for every trading day d and cannot exceed a value of 0.5. In this special constellation, probabilities of good- and bad-news for trading day d are given by

$$\pi_{\mathcal{G},d} = \pi_{\mathcal{B},d} = \frac{1}{2} - \frac{\pi_{\mathcal{N},d}}{2} \tag{11.1}$$

The empirical results show that the modeling of states' conditions is problematic in the context of the approach by Tay, Ting, Tse, and Warachka (2009). While the parameter estimates in their original work are reasonable, they are often not for equities in our data. These findings may be fortified due to the recent time range of our underlying tick data, however, the shortcomings of the approach, in addition to those explained in section 6.1, are clearly revealed.

In contrast to Tay, Ting, Tse, and Warachka (2009) the reasonable relations $\hat{\nu}_{1,1} < \hat{\nu}_{1,-1}$ and $\hat{\nu}_{-1,-1} < \hat{\nu}_{-1,1}$ do not hold for most symbols and years in our estimation results for the PIN-ALACD model. This can be interpreted that consecutive buys or sells introduce larger values

for the intercepts in the corresponding ALACD specifications and thus lead to higher values of the conditional expected durations than switching from the opposite trade direction.

Furthermore, Tay, Ting, Tse, and Warachka (2009) report that the relations $\hat{\zeta}_1 < 0$ and $\hat{\zeta}_{-1} > 0$ are valid for all equities under consideration in their work. Hence, buyer-initiated transactions with large size would substantially reduce the conditional expected duration for subsequent buys but increase it for sells. The opposite is true for seller-initiated transactions with high volume. Our findings are somewhat contrary since for the majority of equities and years it holds that $\hat{\zeta}_1 > 0$ and $\hat{\zeta}_{-1} < 0$. This means that our estimates of the coefficients of signed volume suggest that large sell orders lead to a decrease of conditional expected durations of subsequent buys and vice versa. Both sums $\hat{\alpha}_1 + \hat{\beta}_1$ and $\hat{\alpha}_{-1} + \hat{\beta}_{-1}$ are below and not close to 1.

Estimation results of ALACD parameters in the context of the PIN-HMM model are much more reasonable compared to the parameter estimates for the PIN-ALACD model. The majority of them satisfy the intuitive assumptions about relations of intercepts and signs of coefficients of signed volume.

Estimates of shape parameters of Weibull distributions for interarrival times are less than 1 for all symbols and years. Therefore the exponential distribution, which is a special case of the Weibull distribution in case the shape parameter equals unity, is rejected for every equity and year. This reflects that the distributional assumption for waiting times in the PIN-ALACD model does not match the characteristics of waiting times in our data sufficiently.

Small shape parameters for the Weibull distribution of waiting times of buys and sells induce decreasing conditional intensity functions for both trade directions. Although the values are similar for buys and sells across all equities and years their differences are statistically significant in most cases (see table 11.1). Therefore an assumption of equal shape parameters would not be appropriate.

We also see boundary solutions of the transition probabilities for PIN-HMM. Exemplary, we will shortly explain the results for TM in 2008 which exhibit estimates on the boundary of the parameter space for $a_{\mathcal{BG}}$, $a_{\mathcal{GB}}$ and $a_{\mathcal{GN}}$. If trading day d is an information event on which negative private information enters the market, the next trading day will not be an information event with positive private news ($\hat{a}_{\mathcal{BG}} = 0$). Since $\hat{a}_{\mathcal{BN}} = 0.28291$, the probability that it will remain in bad-news state is $\hat{a}_{\mathcal{BB}} = 0.71709$. Additionally, if insiders are triggered to buy on trading day d, they will neither buy nor sell on $d + 1$ because $\hat{a}_{\mathcal{GB}} = 0$ and $\hat{a}_{\mathcal{GN}} = 1$ and therefore $\hat{a}_{\mathcal{GG}} = 0$. The last missing diagonal element is then given by $\hat{a}_{\mathcal{NN}} = 1 - \hat{a}_{\mathcal{NG}} - \hat{a}_{\mathcal{NB}} = 0.77879$. Hence, if the Markov chain resides in no-news state for the current trading day it is very likely that subsequent trading days are also non-information events.

It is obvious from the estimation results that the rows of the transition matrix are not identical for every symbol and year. Hence, this shows that the independence of conditions of trading days, which is nested in the HMM approach, is inappropriate for our data.

In addition to the transition probabilities we gain stationary probability distributions of trading days (see section 6.2.1) which are presented in tables 11.11 – 11.14. These probabilities give an

overview about the proportion of trading days each state occupies. Hence, stationary probability distribution for F in 2007 suggests that approximately 2% of all trading days reside in a bad-news state and that on roughly 32% of trading days there are market attendees which are triggered by positive private news. For the remaining 66% of trading days only noise traders are active. The elements of the stationary probability distribution can be interpreted as counterparts of the probabilities $1 - \alpha$ (no-news), $\alpha(1 - \delta)$ (good-news) and $\alpha\delta$ (bad-news) from the static EKOP/EHO model.

We can see in tables 11.11 - 11.14 that non-information events are dominating for almost every equity and year. Only TM in 2007 exhibits a slightly higher proportion of trading days on which private information enter the market, 50.1% vs. 49.9%. The same security display the highest value for the proportion of no-news day in 2010 with about 87%.

The PIN-HMM model sections information events to good-news and bad-news conditions very differently depending on the underlying time range and equity. However, we can often identify a similar ranking of types of information events over years for each security. The results show that the proportion of good-news day is always higher than that of bad-news days from 2007 to 2010 for F. In 2007 and 2008 there is a difference of more than 30% between the two types of information events. For the remaining equities, in most years the probability of bad-news trading days exhibits higher values than the probability of information events driven by positive private information.

Compared with the results in section 10.1, we see that the proportions of trading days' conditions diverge in the static model approach and the PIN-HMM model. Exemplary, we compare the results of both models for TM in 2007. For the EHO model we receive quarterly estimates of the proportion of bad-news days, by calculating $\widehat{\alpha}\hat{\delta}$, which equal 13.11%, 30.64%, 31.15% and 12.74% for the first to the fourth quarter of 2007. Hence, the EHO model assigns on average a bad-news condition to 21.91% of all trading days. The proportion of bad-news days in our dynamic approach is about 38% and therefore almost two times higher.

The amount of information events with positive direction also differs among the two approaches. In the EHO model we receive estimated proportions with $\widehat{\alpha}(1 - \hat{\delta})$ as 14.75%, 19.36%, 0% and 29.03% which means that the model on average sees 15.79% of all trading days as good-news days. This is nearly 4% higher as the proportion returned by the PIN-HMM model which equals 11.99%. In two out of four quarters in 2007, the parameter estimates for the static model represent that if private information enter the market it has more likely a positive direction. In the third quarter $\widehat{\alpha}(1 - \hat{\delta})$ equals 0 and therefore the model identifies all information events as bad-news days. This is contradictory to the findings in the PIN-HMM model where the proportion of bad-news day is more than three times higher than that of good-news days.

The proportion of no-news days in the EHO model is substantially higher than in our dynamic model. The estimates of $1 - \widehat{\alpha}$ for the four quarters of 2007 are given by 72.13%, 50%, 68.85% and 58.22%, which means that 62.30% of all trading days in 2007 are non-information events for TM according to the static model. As mentioned before, the proportion of no-news days is even slightly lower than 50% in the PIN-HMM model.

We can also compare the amount of informed buys or sells on good-news or bad-news trading days relative to the total number of transactions belonging to the corresponding trade direction in our dynamic model and the static EHO model, respectively.[124] Each yearly series of quarterly estimates of the involved parameters in the EHO model ($\epsilon_b, \epsilon_s, \mu$) is summed up to be comparable, in terms of the underlying time range, to the results of the PIN-HMM model. We decided to exclude the PIN-ALACD approach in this comparison due to the previously discussed spurious findings for the parameter estimates and the fact that one is not able to differentiate between the fraction of informed buys and informed sells. According to equation (6.69) in section 6.3, the proportions are identical due to the design of the approach.

It is obvious from tables 11.15 and 11.17 that the fraction of informed buys on good-news days is substantially lower for almost all equities and years in our dynamic model than it is in the EHO model. Only for TM in 2008 the former exhibit a higher value. The remaining entries in table 11.15 are in the range from about 0.14 to 0.25, which can be interpreted that at most a quarter of all buys on a good-news trading day is initiated by insiders (TM in 2009).

The minimum and maximum values in table 11.17 are about 0.36 and 0.64. The results for the EHO model tell us that more than 60% of all buys on information events with positive direction for TM in 2010, F in 2008 and 2010 and GM in 2009 belong to the group of information-based trading. To stick with the GM symbol in 2009, we can read a more than two times higher value from table 11.17 compared to the corresponding entry in table 11.15 for the PIN-HMM model. This year, and especially the second quarter of it, had some very special trading days for GM, with very high market activity, as discussed in section 10.1. Since the static model assumes a constant trading rate for noise traders, the increased trading intensity on these trading days is assigned solely to the informed traders. The results in tables 11.15 and 11.17 indicate that the static EHO model seems to overestimate the fraction of informed buys of the total number of buys on information events driven by positive private news.

Tables 11.16 and 11.18 display the percentage of sells initiated by informed traders relative to the total number of sells on bad-news trading days. Entries in table 11.18 do not vary much from the ones in table 11.17 which results from similar estimates for the intensity of uninformed buys and uninformed sells in the EHO model, ϵ_b and ϵ_s. For the PIN-HMM model we see a overall increase of the proportion of information-based transactions. Following the estimation results of our dynamic approach, there are noticeable more informed sellers active on bad-news trading days than informed buyers on good-news trading days.

Again, we exemplarily pick the entries for GM in 2009 in tables 11.16 and 11.18. Values are higher than their counterparts on good-news trading days for both models. We see about 0.07 increase for the PIN-HMM model and 0.02 for the static EHO model. Similar to the number of informed buys on good-news trading days, the estimation results show that the static EHO model seem to overestimate the fraction of informed sells of the total amount of sells on bad-news days.

[124] The corresponding expressions can be found in section 6.3.

	Param.	F	GM	HMC	JCI	TM
State Probabilities						
Intercept	δ_1	-2.15933	-0.32914	-0.81641	-0.32043	-0.84784
		(0.24905)	(0.16287)	(0.20863)	(0.21360)	(0.22868)
Cmp. total volumes	δ_2	3.94772	4.27192	5.37413	7.29462	6.84656
		(0.66458)	(0.56619)	(0.72990)	(0.96071)	(0.94355)
Cmp. sell volumes	δ_3	**0**	**0**	**0**	**0**	**0**
		— — —	— — —	— — —	— — —	— — —
Cmp. buy volumes	δ_4	0.79432	1.02764	**0**	3.49313	**0**
		(0.81372)	(0.49725)	— — —	(0.78016)	— — —
ALACD Specifications						
Intercept (Buy after Buy)	$\nu_{1,1}$	0.60117	0.40789	0.50163	0.28742	0.39830
		(0.00492)	(0.00494)	(0.01726)	(0.01001)	(0.01481)
Intercept (Buy after Sell)	$\nu_{1,-1}$	0.39170	0.31344	0.51220	0.29826	0.43103
		(0.00490)	(0.00478)	(0.01886)	(0.01084)	(0.01717)
Cond. Duration (Buys)	α_1	0.68509	0.74261	0.80078	0.84529	0.82255
		(0.00187)	(0.00193)	(0.00483)	(0.00385)	(0.00441)
Lagged Duration (Buys)	β_1	0.15250	0.12863	0.08557	0.07290	0.07773
		(0.00079)	(0.00070)	(0.00161)	(0.00120)	(0.00131)
Signed Volume (Buys)	ζ_1	0.01782	0.01245	0.02249	0.01390	0.02153
		(0.00063)	(0.00071)	(0.00237)	(0.00141)	(0.00236)
Intercept (Sell after Buy)	$\nu_{-1,1}$	0.53469	0.28073	0.29984	0.09187	0.21211
		(0.00509)	(0.00449)	(0.01329)	(0.00595)	(0.01069)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.46048	0.20164	0.29857	0.13400	0.25093
		(0.00464)	(0.00368)	(0.01098)	(0.00572)	(0.01050)
Cond. Duration (Sells)	α_{-1}	0.68381	0.81301	0.87221	0.93387	0.89541
		(0.00179)	(0.00163)	(0.00350)	(0.00146)	(0.00290)
Lagged Duration (Sells)	β_{-1}	0.15225	0.11192	0.06164	0.04465	0.05563
		(0.00077)	(0.00073)	(0.00135)	(0.00078)	(0.00114)
Signed Volume (Sells)	ζ_{-1}	-0.03916	-0.02218	-0.01102	-0.00308	-0.00619
		(0.00064)	(0.00065)	(0.00165)	(0.00103)	(0.00161)
Adj. informed trading	τ	0.39904	0.28767	0.35902	0.30955	0.35061
		(0.00303)	(0.00237)	(0.00605)	(0.00402)	(0.00510)

Table 11.3: Estimation results for the PIN-ALACD model for all US stocks in 2007. Figures in parentheses denote standard errors. Boundary solutions are marked with red color.

	Param.	F	GM	HMC	JCI	TM
State Probabilities						
Intercept	δ_1	-0.79968	-0.46281	-1.47829	0.01149	-1.25316
		(0.18221)	(0.18027)	(0.25948)	(0.18323)	(0.30503)
Cmp. total volumes	δ_2	4.96020	5.17025	6.19418	6.58751	9.37939
		(0.68550)	(0.66851)	(0.93451)	(0.83171)	(1.46119)
Cmp. sell volumes	δ_3	2.39595	0	0	0	0
		(0.67187)	— — —	— — —	— — —	— — —
Cmp. buy volumes	δ_4	0	3.87484	0	2.07227	5.76414
		— — —	(0.86024)	— — —	(0.63694)	(1.41118)
ALACD Specifications						
Intercept (Buy after Buy)	$\nu_{1,1}$	0.68016	0.37317	0.33832	0.32778	0.63113
		(0.00389)	(0.00441)	(0.01026)	(0.00741)	(0.02029)
Intercept (Buy after Sell)	$\nu_{1,-1}$	0.05859	0.19909	0.24493	0.26482	0.54429
		(0.00309)	(0.00379)	(0.01097)	(0.00723)	(0.02169)
Cond. Duration (Buys)	α_1	0.76576	0.81208	0.86969	0.82744	0.77105
		(0.00107)	(0.00147)	(0.00245)	(0.00239)	(0.00516)
Lagged Duration (Buys)	β_1	0.15016	0.09988	0.06999	0.07579	0.07995
		(0.00061)	(0.00059)	(0.00113)	(0.00075)	(0.00134)
Signed Volume (Buys)	ζ_1	-0.01532	0.00554	0.00677	0.01271	0.01759
		(0.00048)	(0.00061)	(0.00178)	(0.00118)	(0.00320)
Intercept (Sell after Buy)	$\nu_{-1,1}$	0.34762	0.12758	0.45405	0.19428	0.22419
		(0.00413)	(0.00306)	(0.01645)	(0.00691)	(0.01412)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.71623	0.30006	0.60662	0.25323	0.41182
		(0.00461)	(0.00354)	(0.01609)	(0.00677)	(0.01447)
Cond. Duration (Sells)	α_{-1}	0.67182	0.84951	0.78388	0.86269	0.86192
		(0.00140)	(0.00118)	(0.00358)	(0.00183)	(0.00326)
Lagged Duration (Sells)	β_{-1}	0.15388	0.09299	0.09241	0.07001	0.06540
		(0.00063)	(0.00056)	(0.00130)	(0.00072)	(0.00120)
Signed Volume (Sells)	ζ_{-1}	-0.01400	-0.00249	-0.01194	-0.01012	0.00164
		(0.00060)	(0.00051)	(0.00262)	(0.00122)	(0.00240)
Adj. informed trading	τ	0.44299	0.34832	0.41660	0.25927	0.53154
		(0.00248)	(0.00251)	(0.00679)	(0.00333)	(0.00723)

Table 11.4: Estimation results for the PIN-ALACD model for all US stocks in 2008. Figures in parentheses denote standard errors. Boundary solutions are marked with red color.

	Param.	F	GM	HMC	JCI	TM
State Probabilities						
Intercept	δ_1	-0.47821	-1.40319	-1.82512	-0.45538	-1.97365
		(0.17104)	(0.31487)	(0.30399)	(0.18258)	(0.37159)
Cmp. total volumes	δ_2	3.98386	1.72313	5.61854	5.49744	8.10310
		(0.53794)	(0.41932)	(0.84390)	(0.71285)	(1.27871)
Cmp. sell volumes	δ_3	0.82378	0	0	0	0
		(0.43867)	— — —	— — —	— — —	— — —
Cmp. buy volumes	δ_4	0	0.99807	1.97490	0	1.11973
		— — —	(0.59346)	(0.59396)	— — —	(0.44653)
ALACD Specifications						
Intercept (Buy after Buy)	$\nu_{1,1}$	0.92732	0.57468	0.56000	0.36017	0.65085
		(0.00566)	(0.01039)	(0.01622)	(0.00545)	(0.01758)
Intercept (Buy after Sell)	$\nu_{1,-1}$	0.16409	0.31579	0.38598	0.06375	0.29303
		(0.00446)	(0.00818)	(0.01697)	(0.00483)	(0.01707)
Cond. Duration (Buys)	α_1	0.65806	0.78017	0.81730	0.87183	0.81242
		(0.00147)	(0.00297)	(0.00314)	(0.00156)	(0.00324)
Lagged Duration (Buys)	β_1	0.14011	0.10598	0.08526	0.06018	0.07857
		(0.00057)	(0.00117)	(0.00127)	(0.00055)	(0.00110)
Signed Volume (Buys)	ζ_1	-0.00877	0.00675	0.00715	-0.00974	-0.00968
		(0.00068)	(0.00111)	(0.00284)	(0.00087)	(0.00300)
Intercept (Sell after Buy)	$\nu_{-1,1}$	0.17846	0.14070	0.25896	0.13161	0.34811
		(0.00428)	(0.00625)	(0.01192)	(0.00436)	(0.01504)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.97357	0.38678	0.30776	0.25511	0.39662
		(0.00568)	(0.00857)	(0.01223)	(0.00482)	(0.01492)
Cond. Duration (Sells)	α_{-1}	0.64452	0.86251	0.88107	0.88102	0.84505
		(0.00145)	(0.00262)	(0.00219)	(0.00137)	(0.00246)
Lagged Duration (Sells)	β_{-1}	0.14181	0.08039	0.06840	0.06012	0.07579
		(0.00055)	(0.00120)	(0.00112)	(0.00053)	(0.00103)
Signed Volume (Sells)	ζ_{-1}	0.01006	-0.00055	-0.01376	-0.00515	-0.01973
		(0.00066)	(0.00086)	(0.00213)	(0.00079)	(0.00273)
Adj. informed trading	τ	0.41035	0.56449	0.56908	0.29995	0.50314
		(0.00259)	(0.00731)	(0.00799)	(0.00280)	(0.00722)

Table 11.5: Estimation results for the PIN-ALACD model for all US stocks in 2009. Figures in parentheses denote standard errors. Boundary solutions are marked with red color.

	Param.	F	JCI	HMC	TM
State Probabilities					
Intercept	δ_1	-0.55512	-0.47733	-1.86455	-5.98879
		(0.19425)	(0.20587)	(0.31767)	(1.37735)
Cmp. total volumes	δ_2	5.31255	6.43511	7.30427	6.58533
		(0.70469)	(0.81189)	(1.13780)	(1.71508)
Cmp. sell volumes	δ_3	2.36186	0	0	0
		(0.56516)	— — —	— — —	— — —
Cmp. buy volumes	δ_4	0	4.35375	8.70362	34.43828
		— — —	(0.88469)	(2.55144)	(1379.92554)
ALACD Specifications					
Intercept (Buy after Buy)	$\nu_{1,1}$	0.85286	0.40554	0.50449	0.21879
		(0.00449)	(0.00951)	(0.02099)	(0.00933)
Intercept (Buy after Sell)	$\nu_{1,-1}$	-0.26158	0.40107	0.43555	0.04898
		(0.00393)	(0.00890)	(0.02156)	(0.00769)
Cond. Duration (Buys)	α_1	0.63147	0.74563	0.83029	0.94218
		(0.00133)	(0.00325)	(0.00429)	(0.00173)
Lagged Duration (Buys)	β_1	0.10555	0.05889	0.05252	0.02999
		(0.00034)	(0.00051)	(0.00105)	(0.00073)
Signed Volume (Buys)	ζ_1	-0.02941	0.03199	0.01576	-0.00749
		(0.00063)	(0.00153)	(0.00351)	(0.00149)
Intercept (Sell after Buy)	$\nu_{-1,1}$	-0.12341	0.08795	0.04749	0.01404
		(0.00409)	(0.00490)	(0.00906)	(0.00292)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.93930	0.21386	0.20171	0.05287
		(0.00480)	(0.00526)	(0.00993)	(0.00305)
Cond. Duration (Sells)	α_{-1}	0.58438	0.90460	0.94906	0.98500
		(0.00139)	(0.00171)	(0.00176)	(0.00036)
Lagged Duration (Sells)	β_{-1}	0.10797	0.03895	0.02564	0.01090
		(0.00035)	(0.00049)	(0.00074)	(0.00025)
Signed Volume (Sells)	ζ_{-1}	0.02177	-0.00240	0.00758	0.00131
		(0.00066)	(0.00088)	(0.00169)	(0.00058)
Adj. informed trading	τ	0.71168	0.58304	0.68140	0.84320
		(0.00236)	(0.00384)	(0.00965)	(0.01029)

Table 11.6: Estimation results for the PIN-ALACD model for all US stocks in 2010. Figures in parentheses denote standard errors. Boundary solutions are marked with red color.

	Param.	F	GM	HMC	JCI	TM
Transistion Probabilities						
$\mathscr{B} \longrightarrow \mathscr{G}$	$a_{\mathscr{B}\mathscr{G}}$	0.60159	0.02666	0.11398	0.39970	0.04162
		(0.21822)	(0.01837)	(0.04298)	(0.08079)	(0.02199)
$\mathscr{B} \longrightarrow \mathscr{N}$	$a_{\mathscr{B}\mathscr{N}}$	0	0.33072	0.39995	0.26802	0.32209
		− − −	(0.05291)	(0.06411)	(0.07381)	(0.05006)
$\mathscr{G} \longrightarrow \mathscr{B}$	$a_{\mathscr{G}\mathscr{B}}$	0.02464	0.05629	0.14985	0.24408	0.08459
		(0.01722)	(0.05483)	(0.05478)	(0.06481)	(0.05587)
$\mathscr{G} \longrightarrow \mathscr{N}$	$a_{\mathscr{G}\mathscr{N}}$	0.46603	0.57992	0.35281	0.38326	0.41759
		(0.05710)	(0.12235)	(0.08537)	(0.07715)	(0.09941)
$\mathscr{N} \longrightarrow \mathscr{B}$	$a_{\mathscr{N}\mathscr{B}}$	0.00610	0.18737	0.22031	0.09313	0.25747
		(0.00608)	(0.03254)	(0.03901)	(0.02417)	(0.04123)
$\mathscr{N} \longrightarrow \mathscr{G}$	$a_{\mathscr{N}\mathscr{G}}$	0.22251	0.06071	0.11631	0.09367	0.08886
		(0.03438)	(0.02016)	(0.03180)	(0.02402)	(0.02781)
ALACD Specifications						
Intercept (Buy after Buy)	$\nu_{1,1}$	0.65129	0.44555	0.51402	0.50818	0.67640
		(0.00616)	(0.00487)	(0.01890)	(0.01740)	(0.02554)
Intercept (Buy after Sell)	$\nu_{1,-1}$	0.63725	0.38036	0.52942	0.50579	0.72617
		(0.00604)	(0.00486)	(0.01580)	(0.01448)	(0.02067)
Cond. Duration (Buys)	α_1	0.67551	0.77631	0.86816	0.82389	0.83624
		(0.00198)	(0.00195)	(0.00488)	(0.00635)	(0.00625)
Lagged Duration (Buys)	β_1	0.15697	0.12912	0.07042	0.08286	0.08220
		(0.00091)	(0.00085)	(0.00201)	(0.00183)	(0.00220)
Signed Volume (Buys)	ζ_1	-0.04183	-0.02684	-0.01454	-0.01839	-0.01758
		(0.00078)	(0.00073)	(0.00240)	(0.00199)	(0.00356)
Intercept (Sell after Buy)	$\nu_{-1,1}$	0.72936	0.61310	1.01478	0.49440	1.12725
		(0.00593)	(0.00629)	(0.03049)	(0.01251)	(0.03302)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.51309	0.49077	1.05435	0.49193	1.27078
		(0.00586)	(0.00620)	(0.03624)	(0.01596)	(0.03727)
Cond. Duration (Sells)	α_{-1}	0.67449	0.70202	0.72834	0.83118	0.67607
		(0.00208)	(0.00240)	(0.00816)	(0.00537)	(0.00848)
Lagged Duration (Sells)	β_{-1}	0.16396	0.14447	0.11466	0.08815	0.12024
		(0.00094)	(0.00087)	(0.00258)	(0.00178)	(0.00214)
Signed Volume (Sells)	ζ_{-1}	0.02391	0.01741	0.04051	0.01787	0.05640
		(0.00076)	(0.00090)	(0.00420)	(0.00202)	(0.00515)
Adjustment (good news)	τ_1	0.24402	0.18818	0.28606	0.29197	0.32701
		(0.00377)	(0.00567)	(0.01218)	(0.00656)	(0.01254)
Adjustment (bad news)	τ_{-1}	0.67373	0.30711	0.53025	0.24996	0.50256
		(0.00644)	(0.00319)	(0.01252)	(0.00681)	(0.01051)
Shape Parameters						
Buys	k_1	0.83112	0.80766	0.71420	0.74811	0.63932
		(0.00089)	(0.00068)	(0.00170)	(0.00104)	(0.00141)
Sells	k_{-1}	0.83028	0.80251	0.67250	0.74095	0.62547
		(0.00089)	(0.00070)	(0.00183)	(0.00103)	(0.00143)

Table 11.7: Estimation results for the PIN-HMM model for all US stocks in 2007. Figures in parentheses denote standard errors. Boundary solutions are marked with red color.

	Param.	F	GM	HMC	JCI	TM
Transition Probabilities						
$\mathscr{B} \longrightarrow \mathscr{G}$	$a_{\mathscr{B}\mathscr{G}}$	0.33119	0.01223	0.47662	0.32786	**0**
		(0.15453)	(0.01215)	(0.13568)	(0.08372)	— — —
$\mathscr{B} \longrightarrow \mathscr{N}$	$a_{\mathscr{B}\mathscr{N}}$	0.66881	0.41850	0.52338	0.40675	0.27631
		(0.15454)	(0.05508)	(0.13568)	(0.08341)	(0.04570)
$\mathscr{G} \longrightarrow \mathscr{B}$	$a_{\mathscr{G}\mathscr{B}}$	**0**	0.22062	0.10128	0.21640	**0**
		— — —	(0.13795)	(0.04245)	(0.05669)	— — —
$\mathscr{G} \longrightarrow \mathscr{N}$	$a_{\mathscr{G}\mathscr{N}}$	0.41791	0.44802	0.49349	0.31420	**1**
		(0.05252)	(0.16556)	(0.07132)	(0.05885)	— — —
$\mathscr{N} \longrightarrow \mathscr{B}$	$a_{\mathscr{N}\mathscr{B}}$	0.01925	0.20613	0.03647	0.11345	0.21614
		(0.01101)	(0.03216)	(0.01432)	(0.02895)	(0.03644)
$\mathscr{N} \longrightarrow \mathscr{G}$	$a_{\mathscr{N}\mathscr{G}}$	0.23444	0.03135	0.14422	0.15990	0.00732
		(0.03419)	(0.01380)	(0.02910)	(0.03179)	(0.00730)
ALACD Specifications						
Intercept (Buy after Buy)	$\nu_{1,1}$	0.51057	0.37977	0.91883	0.56529	1.00520
		(0.00573)	(0.00542)	(0.02749)	(0.01207)	(0.03489)
Intercept (Buy after Sell)	$\nu_{1,-1}$	1.02859	0.70040	1.10009	0.70012	1.26153
		(0.00642)	(0.00622)	(0.02717)	(0.01277)	(0.03391)
Cond. Duration (Buys)	α_1	0.66487	0.77699	0.75393	0.77234	0.74069
		(0.00163)	(0.00191)	(0.00500)	(0.00359)	(0.00612)
Lagged Duration (Buys)	β_1	0.17165	0.12737	0.12078	0.09846	0.12092
		(0.00084)	(0.00083)	(0.00215)	(0.00111)	(0.00221)
Signed Volume (Buys)	ζ_1	-0.01773	-0.00473	-0.02857	-0.01871	-0.02389
		(0.00082)	(0.00089)	(0.00457)	(0.00203)	(0.00612)
Intercept (Sell after Buy)	$\nu_{-1,1}$	1.03479	0.74050	0.77394	0.62691	1.43615
		(0.00577)	(0.00681)	(0.01786)	(0.01111)	(0.03905)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.31732	0.44114	0.65936	0.49639	1.33601
		(0.00471)	(0.00599)	(0.01993)	(0.01055)	(0.04101)
Cond. Duration (Sells)	α_{-1}	0.71574	0.75070	0.83618	0.80317	0.67870
		(0.00141)	(0.00211)	(0.00379)	(0.00290)	(0.00726)
Lagged Duration (Sells)	β_{-1}	0.17829	0.12666	0.09636	0.09472	0.12382
		(0.00083)	(0.00082)	(0.00191)	(0.00105)	(0.00225)
Signed Volume (Sells)	ζ_{-1}	-0.00375	0.00877	0.01779	0.01625	0.04644
		(0.00071)	(0.00096)	(0.00328)	(0.00185)	(0.00672)
Adjustment (good news)	τ_1	0.34873	0.30885	0.42624	0.29319	1.48118
		(0.00379)	(0.00785)	(0.01317)	(0.00544)	(0.06849)
Adjustment (bad news)	τ_{-1}	1.11377	0.34986	0.60044	0.18999	0.59531
		(0.00871)	(0.00360)	(0.01787)	(0.00636)	(0.01357)
Shape Parameters						
Buys	k_1	0.73201	0.71628	0.61456	0.70491	0.55782
		(0.00073)	(0.00068)	(0.00164)	(0.00091)	(0.00147)
Sells	k_{-1}	0.72457	0.72492	0.63172	0.70688	0.56028
		(0.00072)	(0.00069)	(0.00152)	(0.00090)	(0.00148)

Table 11.8: Estimation results for the PIN-HMM model for all US stocks in 2008. Figures in parentheses denote standard errors. Boundary solutions are marked with red color.

	Param.	F	GM	HMC	JCI	TM
Transistion Probabilities						
$\mathscr{B} \longrightarrow \mathscr{G}$	$a_{\mathscr{B}\mathscr{G}}$	0.04762	**0**	0.02004	0.11395	0.11395
		(0.04173)	— — —	(0.01985)	(0.04445)	(0.04227)
$\mathscr{B} \longrightarrow \mathscr{N}$	$a_{\mathscr{B}\mathscr{N}}$	0.55225	0.34550	0.33481	0.41383	0.35730
		(0.08817)	(0.26815)	(0.06868)	(0.06923)	(0.06313)
$\mathscr{G} \longrightarrow \mathscr{B}$	$a_{\mathscr{G}\mathscr{B}}$	0.14113	**0**	0.18899	0.13902	0.40644
		(0.04655)	— — —	(0.12110)	(0.05781)	(0.14451)
$\mathscr{G} \longrightarrow \mathscr{N}$	$a_{\mathscr{G}\mathscr{N}}$	0.34402	0.67436	0.52505	0.46305	0.35597
		(0.06353)	(0.10119)	(0.15643)	(0.08303)	(0.14440)
$\mathscr{N} \longrightarrow \mathscr{B}$	$a_{\mathscr{N}\mathscr{B}}$	0.07395	0.02531	0.09537	0.14299	0.15132
		(0.02159)	(0.01770)	(0.02285)	(0.02897)	(0.02896)
$\mathscr{N} \longrightarrow \mathscr{G}$	$a_{\mathscr{N}\mathscr{G}}$	0.16413	0.20004	0.03757	0.09920	0.01221
		(0.03014)	(0.04679)	(0.01498)	(0.02402)	(0.00859)
ALACD Specifications						
Intercept (Buy after Buy)	$\nu_{1,1}$	0.46582	0.60163	0.78793	0.47486	0.94353
		(0.00768)	(0.01428)	(0.02528)	(0.00922)	(0.03040)
Intercept (Buy after Sell)	$\nu_{1,-1}$	1.79008	1.15820	0.96329	0.76118	1.20655
		(0.01021)	(0.01928)	(0.02565)	(0.01075)	(0.03106)
Cond. Duration (Buys)	α_1	0.60915	0.75661	0.83395	0.82778	0.79522
		(0.00187)	(0.00429)	(0.00389)	(0.00246)	(0.00424)
Lagged Duration (Buys)	β_1	0.19020	0.12579	0.10777	0.08421	0.11490
		(0.00093)	(0.00188)	(0.00225)	(0.00093)	(0.00207)
Signed Volume (Buys)	ζ_1	0.01506	-0.00314	-0.02359	-0.00599	-0.02808
		(0.00118)	(0.00200)	(0.00453)	(0.00170)	(0.00564)
Intercept (Sell after Buy)	$\nu_{-1,1}$	1.80710	1.19963	1.13708	0.86620	1.37778
		(0.01037)	(0.01815)	(0.02966)	(0.01064)	(0.03475)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.48111	0.68052	0.91140	0.41849	1.00013
		(0.00799)	(0.01371)	(0.03134)	(0.00926)	(0.03512)
Cond. Duration (Sells)	α_{-1}	0.61395	0.73731	0.77553	0.81903	0.75390
		(0.00189)	(0.00397)	(0.00520)	(0.00260)	(0.00538)
Lagged Duration (Sells)	β_{-1}	0.19074	0.14442	0.11896	0.08441	0.11569
		(0.00097)	(0.00190)	(0.00232)	(0.00092)	(0.00211)
Signed Volume (Sells)	ζ_{-1}	-0.01224	0.00940	0.02288	-0.00687	0.01850
		(0.00122)	(0.00200)	(0.00526)	(0.00167)	(0.00625)
Adjustment (good news)	τ_1	0.43301	0.42876	0.44880	0.35768	0.52783
		(0.00561)	(0.01215)	(0.02688)	(0.00680)	(0.02408)
Adjustment (bad news)	τ_{-1}	0.32558	0.60004	0.63591	0.32801	0.55021
		(0.00726)	(0.01758)	(0.01574)	(0.00597)	(0.01448)
Shape Parameters						
Buys	k_1	0.59999	0.62335	0.57472	0.64314	0.53948
		(0.00067)	(0.00162)	(0.00179)	(0.00080)	(0.00155)
Sells	k_{-1}	0.59006	0.60511	0.59143	0.64127	0.54656
		(0.00069)	(0.00156)	(0.00177)	(0.00082)	(0.00153)

Table 11.9: Estimation results for the PIN-HMM model for all US stocks in 2009. Figures in parentheses denote standard errors. Boundary solutions are marked with red color.

	Param.	F	JCI	HMC	TM
Transistion Probabilities					
$\mathscr{B} \longrightarrow \mathscr{G}$	$a_{\mathscr{B}\mathscr{G}}$	0.48592	0.42286	0.11663	0.18174
		(0.16273)	(0.08424)	(0.07022)	(0.11634)
$\mathscr{B} \longrightarrow \mathscr{N}$	$a_{\mathscr{B}\mathscr{N}}$	0.21525	0.00000	0.58670	0.36563
		(0.13585)	(0.00000)	(0.10079)	(0.14506)
$\mathscr{G} \longrightarrow \mathscr{B}$	$a_{\mathscr{G}\mathscr{B}}$	0.06215	0.32869	0.16707	0.04871
		(0.03012)	(0.11939)	(0.07782)	(0.04767)
$\mathscr{G} \longrightarrow \mathscr{N}$	$a_{\mathscr{G}\mathscr{N}}$	0.40626	0.59047	0.68000	0.50262
		(0.06285)	(0.12284)	(0.10257)	(0.12287)
$\mathscr{N} \longrightarrow \mathscr{B}$	$a_{\mathscr{N}\mathscr{B}}$	0.01685	0.11650	0.07903	0.02419
		(0.00965)	(0.02476)	(0.02164)	(0.01070)
$\mathscr{N} \longrightarrow \mathscr{G}$	$a_{\mathscr{N}\mathscr{G}}$	0.14054	0.03318	0.10727	0.04472
		(0.02643)	(0.01340)	(0.02878)	(0.01533)
ALACD Specifications					
Intercept (Buy after Buy)	$\nu_{1,1}$	0.67172	0.96222	0.99353	0.97065
		(0.01031)	(0.01711)	(0.04216)	(0.02715)
Intercept (Buy after Sell)	$\nu_{1,-1}$	2.89956	1.48154	1.20329	1.10808
		(0.01263)	(0.02057)	(0.04074)	(0.02613)
Cond. Duration (Buys)	α_1	0.53455	0.66325	0.80428	0.85599
		(0.00175)	(0.00501)	(0.00771)	(0.00452)
Lagged Duration (Buys)	β_1	0.20198	0.09929	0.08069	0.08486
		(0.00083)	(0.00100)	(0.00225)	(0.00205)
Signed Volume (Buys)	ζ_1	0.02273	-0.02184	-0.01936	-0.02191
		(0.00162)	(0.00319)	(0.00698)	(0.00487)
Intercept (Sell after Buy)	$\nu_{-1,1}$	2.90406	1.51120	1.41874	1.37538
		(0.01282)	(0.02200)	(0.04582)	(0.03202)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.53726	1.21037	1.22793	1.09948
		(0.01058)	(0.01849)	(0.04673)	(0.03237)
Cond. Duration (Sells)	α_{-1}	0.53724	0.61833	0.74490	0.78573
		(0.00175)	(0.00467)	(0.00742)	(0.00527)
Lagged Duration (Sells)	β_{-1}	0.19559	0.10212	0.09144	0.10001
		(0.00083)	(0.00105)	(0.00213)	(0.00190)
Signed Volume (Sells)	ζ_{-1}	-0.03033	0.05255	0.03365	0.02530
		(0.00165)	(0.00344)	(0.00811)	(0.00589)
Adjustment (good news)	τ_1	0.54638	0.47120	0.48410	0.33062
		(0.00644)	(0.01603)	(0.02691)	(0.01964)
Adjustment (bad news)	τ_{-1}	1.03469	0.62769	0.84790	1.12550
		(0.00946)	(0.00873)	(0.02094)	(0.01856)
Shape Parameters					
Buys	k_1	0.42847	0.50492	0.53908	0.49973
		(0.00042)	(0.00075)	(0.00186)	(0.00131)
Sells	k_{-1}	0.43207	0.50155	0.52733	0.49554
		(0.00043)	(0.00075)	(0.00181)	(0.00130)

Table 11.10: Estimation results for the PIN-HMM model for all US stocks in 2010. Figures in parentheses denote standard errors.

	F	GM	HMC	JCI	TM
Bad	0.01988	0.32684	0.28212	0.16181	0.38111
Good	0.32256	0.07114	0.18685	0.19859	0.11988
No	0.65756	0.60202	0.53103	0.63960	0.49900

Table 11.11: Stationary distribution of trading days' conditions for US equities in 2007.

	F	GM	HMC	JCI	TM
Bad	0.01207	0.32448	0.04876	0.16681	0.43710
Good	0.36117	0.03592	0.21709	0.27215	0.00409
No	0.62676	0.63959	0.73415	0.56104	0.55880

Table 11.12: Stationary distribution of trading days' conditions for US equities in 2008.

	F	GM	HMC	JCI	TM
Bad	0.13258	0.05348	0.22116	0.21231	0.26371
Good	0.22899	0.21654	0.04483	0.14593	0.05040
No	0.63842	0.72998	0.73402	0.64177	0.68589

Table 11.13: Stationary distribution of trading days' conditions for US equities in 2009.

	F	HMC	JCI	TM
Bad	0.03944	0.11379	0.27377	0.04600
Good	0.25317	0.11352	0.14686	0.08561
No	0.70739	0.77269	0.57937	0.86839

Table 11.14: Stationary distribution of trading days' conditions for US equities in 2010.

	2007	2008	2009	2010
F	0.18357	0.22530	0.22880	0.20873
GM	0.14100	0.19846	0.23453	
HMC	0.18478	0.23045	0.22736	0.22970
JCI	0.19622	0.18671	0.20550	0.21173
TM	0.18866	0.56231	0.24780	0.15229

Table 11.15: Proportion informed buys occupy of the total number of buys on information events driven by positive private news in the context of our dynamic PIN-HMM model (US equities).

	2007	2008	2009	2010
F	0.42844	0.55381	0.17479	0.36050
GM	0.21843	0.22401	0.30448	
HMC	0.29994	0.31567	0.31346	0.36054
JCI	0.16907	0.12568	0.18969	0.27008
TM	0.26973	0.28361	0.25972	0.42749

Table 11.16: Proportion informed sells occupy of the total number of sells on information events driven by negative private news in the context of our dynamic PIN-HMM model (US equities).

	2007	2008	2009	2010
F	0.54198	0.62478	0.47292	0.64054
GM	0.39600	0.40599	0.60731	
HMC	0.39238	0.51197	0.54770	0.58975
JCI	0.38356	0.36616	0.40826	0.42920
TM	0.44824	0.46075	0.54948	0.62681

Table 11.17: Proportion informed buys occupy of the total number of buys on information events driven by positive private news in the context of the static EHO model (US equities). Yearly series of quarterly estimates of ϵ_b and μ are summed up to achieve a better comparability with the estimates in table 11.15.

	2007	2008	2009	2010
F	0.56119	0.61773	0.52463	0.65496
GM	0.38171	0.41039	0.62672	
HMC	0.47500	0.46299	0.55610	0.59883
JCI	0.37644	0.36854	0.41735	0.41794
TM	0.48670	0.45511	0.55990	0.66746

Table 11.18: Proportion informed sells occupy of the total number of sells on information events driven by negative private news in the context of the static EHO model (US equities). Yearly series of quarterly estimates of ϵ_s and μ are summed up to achieve a better comparability with the estimates in table 11.16.

11.2 Estimation Results (Xetra)

This section covers estimation results for the German equities (BMW, CON, DAI and VOW) listed on the electronic trading system Xetra. The structure of the results pretty much resembles those for the US stocks presented and discussed in the previous section.

For most symbols and years we see that $\hat{\nu}_{1,1} > \hat{\nu}_{1,-1}$ and $\hat{\nu}_{-1,-1} > \hat{\nu}_{-1,1}$ for PIN-ALACD. Again, this means that sells induce lower subsequent conditional expected durations for buys' process and vice versa. In combination with the findings of positive $\hat{\zeta}_1$ and negative $\hat{\zeta}_{-1}$ for the majority of symbols and years, it seems problematic for the ALACD recursion in the PIN-ALACD model to determine trade directions (winner of the race between conditional expected durations of buys and sells to be the first to arrive) in a meaningful way.

Likewise to the results for NYSE-listed equities there is at least one boundary solution for every equity and year for the parameters $\hat{\delta}_3$ or $\hat{\delta}_4$ which are involved in the calculation of probabilities of states of trading days. We are faced with the situation that both estimates equal 0 for BMW in 2007 and VOW in 2009. Hence, total aggregated volume influences the decision about information events but given that private news hit the market, the model is indifferent about its direction. There is no chance to clearly mark trading days with good- or bad-news condition.

As for the US symbols, PIN-HMM applied on the data of the German equities shows much more intuitive and reasonable results. The relations $\hat{\nu}_{1,1} < \hat{\nu}_{1,-1}$ and $\hat{\nu}_{-1,-1} < \hat{\nu}_{-1,1}$ hold for the majority of equities and years, $\hat{\zeta}_1 < 0$ and $\hat{\zeta}_{-1} > 0$ is valid for all cases. Similar to the findings in section 11.1, $\hat{\alpha}_1 + \hat{\beta}_1$ and $\hat{\alpha}_{-1} + \hat{\beta}_{-1}$ are not close to 1. Estimates of shape parameters of Weibull distribution are all less than unity which yield decreasing conditional intensity functions, as it is the case for NYSE stocks, and signal that large durations are less probable than short durations. Similar to the estimation results in section 11.1, this indicates that the exponential distribution, which is applied in the PIN-ALACD model, is not flexible enough for the German symbols.

To summarize the estimation results for the PIN-ALACD and PIN-HMM model, it seems that the general structure of results or relations between certain estimates are independent of the marketplace under consideration. Applying the former on our datasets results in estimates that are counterintuitive, as explained above. A reason for this may be the big difference in origins of the datasources utilized by Tay, Ting, Tse, and Warachka (2009) and in this work. Datasets covering the time span from July 1, 1994 to June 30, 1995 were used by the former. Therefore datasets utilized in this work consist of much higher proportion of (very) short durations which were already reported to be somewhat problematic for PIN-ALACD (see Tay, Ting, Tse, and Warachka 2009).

The stationary distributions of conditions of trading days in tables 11.27 - 11.30 display that the proportion of no-news days is always above 50% for the German equities. Its minimum and maximum value equals 57.8% for CON in 2008 and 83.25% for VOW in 2008, respectively. The results of our dynamic model exhibit that there are always more good-news days than

bad-news days from 2007 to 2010 for BMW. The same is valid for CON, except in 2009 where the proportion of bad-news days is almost three times higher than that of information events driven by positive news. For the remaining two securities, DAI and VOW, the majority of years show a higher proportion of information events on which only informed sellers are active.

Likewise to the previous section, the results for the stationary distribution of trading days' states for one equity and year are compared with the outcomes of the EHO model. We decided to consider DAI in 2010, since the PIN-HMM model almost exclusively assigns a bad-news label to information events in this time span. Only 0.4% of all trading days in this year belong to the group of good-news days, while there are about 37% of bad-news days.

Calculating the probability of good-news days in the EHO model with $\widehat{\alpha}(1 - \hat{\delta})$, we receive 20.45%, 38.10%, 10.61% and 31.75% for the first to the fourth quarter. Hence, the static approach on average sees 25.23% of good-news days in 2010. In each of the four quarters the probability of good-news days is substantially higher than that in the PIN-HMM model.

For the probability of bad-news days, $\widehat{\alpha}\hat{\delta}$, we get the values 1.77%, 4.76%, 22.76% and 6.35% for DAI in 2010. Therefore the average proportion of bad-news trading days equals 8.91% according to the EHO model. It is obvious that the static approach returns proportions of good-news days which are higher than their equivalents for bad-news trading days for three out of four quarters in 2010. These findings are the complete opposite of the results of our dynamic approach.

Finally, the estimates for the proportion of non-information events in the EHO model, $(1 - \widehat{\alpha})$, are given by 77.78%, 57.14%, 66.63% and 61.90% which yield an average of 65.86%. The estimates in the third and fourth quarter and the average value are on a similar level as for the PIN-HMM model. Hence, the proportions of no-news days do not differ that much, but the two approaches return totally different relations of types of information events.

Similar to section 11.1, we compare the fraction of informed buys on good-news trading days and informed sells on bad-news trading days in our dynamic PIN-HMM model and the static EHO model. Again, each yearly series of quarterly estimates of the intensity parameters in the EHO model is summed up. Likewise to the comparison for the symbols listed on the NYSE, we decided to exclude the PIN-ALACD model.

Tables 11.31 and 11.33 display the proportion the buyer-initiated, information-based transactions occupy of the total number of buys on information events triggered by positive private news in the PIN-HMM and EHO model. Analogously, tables 11.32 and 11.34 consist of proportions of sells initiated by insiders on bad-news trading days.

The results pretty much resemble the findings already discussed in the context of the NYSE-listed symbols. All percentages of informed buys on good-news days are substantially lower in our dynamic model than they are in the EHO model. For sells initiated by informed traders our model reports values that are higher than their counterparts in the EHO model for Conti in 2007 and 2010. Similar to section 11.1, we see an overall higher level of the fraction of informed sells compared to the fraction of insiders triggered by positive private news, in terms of the PIN-HMM model. In addition, the levels of reported fractions of informed buys and sells for the German equities are akin to those of the US symbols for both approaches. However, the

results of both models show us that there is on average a slightly higher activity of insiders on the NYSE.

It is obvious from the entries in tables 11.31 – 11.34 that they fortify the assumption that the static EHO model overestimates the amount of information-based trading due to the fixed activity of noise traders.

As for GM in 2009, there were some really special trading days for the VOW symbol in 2008. We already explained the circumstances in section 10.2. Values in tables 11.33 and 11.34 exhibit that the EHO model assigns more than 60% of buys and sells on the corresponding information events to the informed traders. Those entries state the maximum values over the complete time span under consideration, from 2007 to 2010. The entries in tables 11.33 and 11.34 for the PIN-HMM model are by far lower. Our dynamic approach labels about 34% of all sells on bad-news trading days and 16% of all buys on good-news trading days as information-based. Those values are the highest and lowest for VOW over the years from 2007 to 2010, respectively. We can identify a switching of the preferred trade direction by the group of insiders for the VOW equity, compared with the years 2007 and 2009.

	Param.	BMW	CON	DAI	VOW
State Probabilities					
Intercept	δ_1	-1.55576	-0.86965	-1.41037	-0.72681
		(0.23233)	(0.21872)	(0.24299)	(0.18776)
Cmp. total volumes	δ_2	6.12298	6.98146	7.72351	3.34667
		(0.90298)	(0.91591)	(1.11589)	(0.43801)
Cmp. sell volumes	δ_3	**0**	4.92129	**0**	**0**
		— — —	(1.07013)	— — —	— — —
Cmp. buy volumes	δ_4	**0**	**0**	2.46592	1.87939
		— — —	— — —	(0.75816)	(0.51554)
ALACD Specifications					
Intercept (Buy after Buy)	$\nu_{1,1}$	0.35431	0.35001	0.25130	0.31373
		(0.00468)	(0.00381)	(0.00262)	(0.00292)
Intercept (Buy after Sell)	$\nu_{1,-1}$	0.30918	0.16476	0.23895	0.22116
		(0.00579)	(0.00345)	(0.00283)	(0.00274)
Cond. Duration (Buys)	α_1	0.80822	0.85546	0.82688	0.83869
		(0.00188)	(0.00128)	(0.00111)	(0.00119)
Lagged Duration (Buys)	β_1	0.08918	0.07989	0.09257	0.08690
		(0.00063)	(0.00057)	(0.00046)	(0.00051)
Signed Volume (Buys)	ζ_1	0.01431	0.00053	0.01420	0.00886
		(0.00069)	(0.00053)	(0.00038)	(0.00039)
Intercept (Sell after Buy)	$\nu_{-1,1}$	0.26267	0.24879	0.14844	0.12592
		(0.00531)	(0.00474)	(0.00226)	(0.00242)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.26341	0.40339	0.19137	0.32556
		(0.00401)	(0.00479)	(0.00226)	(0.00302)
Cond. Duration (Sells)	α_{-1}	0.85066	0.82483	0.88458	0.86397
		(0.00182)	(0.00184)	(0.00091)	(0.00109)
Lagged Duration (Sells)	β_{-1}	0.07369	0.08091	0.07121	0.07666
		(0.00063)	(0.00064)	(0.00045)	(0.00050)
Signed Volume (Sells)	ζ_{-1}	-0.01404	-0.00612	-0.00907	0.00163
		(0.00061)	(0.00063)	(0.00031)	(0.00038)
Adj. informed trading	τ	0.29527	0.33449	0.29014	0.33976
		(0.00392)	(0.00331)	(0.00215)	(0.00238)

Table 11.19: Estimation results for the PIN-ALACD model for all German stocks in 2007. Figures in parentheses denote standard errors.

	Param.	BMW	CON	DAI	VOW
State Probabilities					
Intercept	δ_1	-0.90127	0.50675	-3.33287	-3.06998
		(0.20501)	(0.22567)	(0.48688)	(0.45627)
Cmp. total volumes	δ_2	6.86164	7.00464	7.66290	5.01874
		(0.88442)	(1.02235)	(1.23858)	(0.82660)
Cmp. sell volumes	δ_3	4.91246	**0**	1.86391	**0**
		(1.03839)	— — —	(0.72658)	— — —
Cmp. buy volumes	δ_4	**0**	5.14298	**0**	5.04521
		— — —	(1.01659)	— — —	(1.95192)
ALACD Specifications					
Intercept (Buy after Buy)	$\nu_{1,1}$	0.26292	0.36148	0.24899	0.25368
		(0.00308)	(0.00369)	(0.00235)	(0.00206)
Intercept (Buy after Sell)	$\nu_{1,-1}$	0.21874	0.26825	0.26106	0.16579
		(0.00305)	(0.00354)	(0.00246)	(0.00195)
Cond. Duration (Buys)	α_1	0.84172	0.83144	0.78941	0.85395
		(0.00118)	(0.00119)	(0.00101)	(0.00098)
Lagged Duration (Buys)	β_1	0.08194	0.08488	0.10430	0.07607
		(0.00046)	(0.00047)	(0.00038)	(0.00038)
Signed Volume (Buys)	ζ_1	0.01328	0.01335	0.02246	0.01260
		(0.00044)	(0.00054)	(0.00036)	(0.00036)
Intercept (Sell after Buy)	$\nu_{-1,1}$	0.26711	0.19906	0.24635	0.12244
		(0.00359)	(0.00304)	(0.00262)	(0.00154)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.30957	0.22698	0.23314	0.19325
		(0.00346)	(0.00281)	(0.00245)	(0.00191)
Cond. Duration (Sells)	α_{-1}	0.81293	0.87523	0.79605	0.89748
		(0.00144)	(0.00105)	(0.00108)	(0.00092)
Lagged Duration (Sells)	β_{-1}	0.08368	0.07240	0.09547	0.06724
		(0.00047)	(0.00046)	(0.00038)	(0.00045)
Signed Volume (Sells)	ζ_{-1}	-0.01456	-0.01358	-0.02110	-0.00729
		(0.00050)	(0.00047)	(0.00039)	(0.00028)
Adj. informed trading	τ	0.34098	0.44710	0.40176	0.64122
		(0.00258)	(0.00315)	(0.00200)	(0.00218)

Table 11.20: Estimation results for the PIN-ALACD model for all German stocks in 2008. Figures in parentheses denote standard errors.

	Param.	BMW	CON	DAI	VOW
State Probabilities					
Intercept	δ_1	-0.59903	-1.90370	-0.46395	-0.83888
		(0.18708)	(0.26878)	(0.16602)	(0.19039)
Cmp. total volumes	δ_2	5.34024	4.00074	4.95678	4.27126
		(0.70554)	(0.61677)	(0.69254)	(0.58314)
Cmp. sell volumes	δ_3	**0**	2.20418	**0**	**0**
		— — —	(0.69666)	— — —	— — —
Cmp. buy volumes	δ_4	3.49544	**0**	0.37099	**0**
		(0.82893)	— — —	(0.51265)	— — —
ALACD Specifications					
Intercept (Buy after Buy)	$\nu_{1,1}$	0.42619	0.26961	0.25098	0.40621
		(0.00480)	(0.00605)	(0.00340)	(0.00379)
Intercept (Buy after Sell)	$\nu_{1,-1}$	0.33395	0.15134	0.39848	0.27887
		(0.00477)	(0.00506)	(0.00374)	(0.00350)
Cond. Duration (Buys)	α_1	0.72487	0.91535	0.69400	0.77459
		(0.00207)	(0.00159)	(0.00147)	(0.00192)
Lagged Duration (Buys)	β_1	0.09148	0.04355	0.11092	0.07933
		(0.00048)	(0.00071)	(0.00039)	(0.00048)
Signed Volume (Buys)	ζ_1	0.02007	0.00157	0.04298	0.02364
		(0.00074)	(0.00079)	(0.00058)	(0.00069)
Intercept (Sell after Buy)	$\nu_{-1,1}$	0.33871	0.29782	0.37416	0.17376
		(0.00481)	(0.00758)	(0.00374)	(0.00317)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.34328	0.38395	0.24046	0.34804
		(0.00449)	(0.00812)	(0.00352)	(0.00371)
Cond. Duration (Sells)	α_{-1}	0.73396	0.86643	0.68461	0.82813
		(0.00205)	(0.00229)	(0.00150)	(0.00207)
Lagged Duration (Sells)	β_{-1}	0.09116	0.05632	0.11368	0.07067
		(0.00049)	(0.00080)	(0.00039)	(0.00056)
Signed Volume (Sells)	ζ_{-1}	-0.02740	-0.00854	-0.04243	-0.00464
		(0.00076)	(0.00105)	(0.00060)	(0.00059)
Adj. informed trading	τ	0.34147	0.57941	0.23460	0.42987
		(0.00284)	(0.00783)	(0.00193)	(0.00272)

Table 11.21: Estimation results for the PIN-ALACD model for all German stocks in 2009. Figures in parentheses denote standard errors.

	Param.	BMW	CON	DAI	VOW
State Probabilities					
Intercept	δ_1	-0.33886	-0.85287	0.34178	-0.65300
		(0.18192)	(0.18083)	(0.22004)	(0.21846)
Cmp. total volumes	δ_2	7.62558	3.83331	9.07680	6.91315
		(1.00073)	(0.50918)	(1.25656)	(0.94238)
Cmp. sell volumes	δ_3	4.50058	0.05283	4.58747	**0**
		(1.00700)	(0.41644)	(0.94139)	— — —
Cmp. buy volumes	δ_4	**0**	**0**	**0**	1.05199
		— — —	— — —	— — —	(0.43730)
ALACD Specifications					
Intercept (Buy after Buy)	$\nu_{1,1}$	0.35160	0.69898	0.21276	0.44689
		(0.00408)	(0.00757)	(0.00276)	(0.00886)
Intercept (Buy after Sell)	$\nu_{1,-1}$	0.31669	0.27166	0.29437	0.28551
		(0.00440)	(0.00711)	(0.00304)	(0.00742)
Cond. Duration (Buys)	α_1	0.73144	0.72428	0.78891	0.83707
		(0.00200)	(0.00326)	(0.00130)	(0.00323)
Lagged Duration (Buys)	β_1	0.08943	0.08401	0.09489	0.05883
		(0.00045)	(0.00065)	(0.00042)	(0.00076)
Signed Volume (Buys)	ζ_1	0.02643	-0.00369	0.02974	0.00792
		(0.00072)	(0.00114)	(0.00049)	(0.00098)
Intercept (Sell after Buy)	$\nu_{-1,1}$	0.37696	0.48223	0.37627	0.12224
		(0.00459)	(0.00736)	(0.00388)	(0.00316)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.39770	0.50329	0.34041	0.41274
		(0.00428)	(0.00731)	(0.00347)	(0.00573)
Cond. Duration (Sells)	α_{-1}	0.71212	0.71595	0.70203	0.87657
		(0.00206)	(0.00311)	(0.00163)	(0.00170)
Lagged Duration (Sells)	β_{-1}	0.08966	0.08115	0.10739	0.05719
		(0.00044)	(0.00062)	(0.00040)	(0.00060)
Signed Volume (Sells)	ζ_{-1}	-0.03081	-0.03499	-0.03488	0.00862
		(0.00072)	(0.00117)	(0.00060)	(0.00060)
Adj. informed trading	τ	0.33192	0.40668	0.35313	0.51152
		(0.00261)	(0.00371)	(0.00225)	(0.00452)

Table 11.22: Estimation results for the PIN-ALACD model for all German stocks in 2010. Figures in parentheses denote standard errors.

	Param.	BMW	CON	DAI	VOW
Transition Probabilities					
$\mathscr{B} \longrightarrow \mathscr{G}$	$a_{\mathscr{B}\mathscr{G}}$	0.43182	**0**	0.30070	0.14017
		(0.10682)	— — —	(0.09041)	(0.04806)
$\mathscr{B} \longrightarrow \mathscr{N}$	$a_{\mathscr{B}\mathscr{N}}$	0.39304	**1**	0.43781	0.34437
		(0.10676)	— — —	(0.09869)	(0.06604)
$\mathscr{G} \longrightarrow \mathscr{B}$	$a_{\mathscr{G}\mathscr{B}}$	0.11647	**0**	0.16081	0.42634
		(0.04531)	— — —	(0.05113)	(0.11414)
$\mathscr{G} \longrightarrow \mathscr{N}$	$a_{\mathscr{G}\mathscr{N}}$	0.55453	0.36070	0.47907	0.21055
		(0.07033)	(0.05133)	(0.06989)	(0.09338)
$\mathscr{N} \longrightarrow \mathscr{B}$	$a_{\mathscr{N}\mathscr{B}}$	0.07174	0.00639	0.06410	0.09892
		(0.01988)	(0.00637)	(0.02021)	(0.02331)
$\mathscr{N} \longrightarrow \mathscr{G}$	$a_{\mathscr{N}\mathscr{G}}$	0.14436	0.21642	0.15594	0.02573
		(0.02789)	(0.03379)	(0.02865)	(0.01272)
ALACD Specifications					
Intercept (Buy after Buy)	$\nu_{1,1}$	0.81929	0.76536	0.52767	0.54661
		(0.00922)	(0.00881)	(0.00442)	(0.00510)
Intercept (Buy after Sell)	$\nu_{1,-1}$	0.81510	1.00095	0.59228	0.89868
		(0.00818)	(0.00886)	(0.00429)	(0.00607)
Cond. Duration (Buys)	α_1	0.77838	0.75318	0.82836	0.81055
		(0.00263)	(0.00267)	(0.00149)	(0.00192)
Lagged Duration (Buys)	β_1	0.10906	0.11490	0.10199	0.10907
		(0.00102)	(0.00100)	(0.00070)	(0.00088)
Signed Volume (Buys)	ζ_1	-0.03103	-0.02041	-0.01934	-0.00267
		(0.00118)	(0.00127)	(0.00059)	(0.00077)
Intercept (Sell after Buy)	$\nu_{-1,1}$	0.90694	0.99664	0.61898	0.81969
		(0.00835)	(0.00790)	(0.00427)	(0.00534)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.84403	0.64243	0.58924	0.58805
		(0.00932)	(0.00711)	(0.00501)	(0.00508)
Cond. Duration (Sells)	α_{-1}	0.74557	0.78901	0.78590	0.79930
		(0.00262)	(0.00214)	(0.00161)	(0.00175)
Lagged Duration (Sells)	β_{-1}	0.12495	0.11848	0.11971	0.11460
		(0.00104)	(0.00099)	(0.00071)	(0.00082)
Signed Volume (Sells)	ζ_{-1}	0.03025	0.00626	0.02435	0.01052
		(0.00123)	(0.00112)	(0.00067)	(0.00077)
Adjustment (good news)	τ_1	0.29686	0.35872	0.19971	0.41672
		(0.00617)	(0.00570)	(0.00380)	(0.00705)
Adjustment (bad news)	τ_{-1}	0.34032	1.26957	0.33702	0.29943
		(0.00777)	(0.03440)	(0.00510)	(0.00466)
Shape Parameters					
Buys	k_1	0.60204	0.60420	0.66345	0.61017
		(0.00073)	(0.00074)	(0.00056)	(0.00064)
Sells	k_{-1}	0.59558	0.59056	0.66216	0.62174
		(0.00071)	(0.00073)	(0.00053)	(0.00062)

Table 11.23: Estimation results for the PIN-HMM model for all German stocks in 2007. Figures in parentheses denote standard errors.

	Param.	BMW	CON	DAI	VOW
Transistion Probabilities					
$\mathscr{B} \longrightarrow \mathscr{G}$	$a_{\mathscr{B}\mathscr{G}}$	0.17874	0.40193	0.20418	0.30664
		(0.08171)	(0.15598)	(0.06284)	(0.12786)
$\mathscr{B} \longrightarrow \mathscr{N}$	$a_{\mathscr{B}\mathscr{N}}$	0.64213	0.20237	0.49978	0.23360
		(0.10345)	(0.12790)	(0.07916)	(0.11741)
$\mathscr{G} \longrightarrow \mathscr{B}$	$a_{\mathscr{G}\mathscr{B}}$	0.16429	0.04075	0.15243	0.19894
		(0.05785)	(0.01998)	(0.06350)	(0.07276)
$\mathscr{G} \longrightarrow \mathscr{N}$	$a_{\mathscr{G}\mathscr{N}}$	0.31109	0.38056	0.39177	0.33620
		(0.07035)	(0.04957)	(0.08406)	(0.08598)
$\mathscr{N} \longrightarrow \mathscr{B}$	$a_{\mathscr{N}\mathscr{B}}$	0.05868	0.01371	0.13912	0.00472
		(0.01810)	(0.00964)	(0.02651)	(0.00471)
$\mathscr{N} \longrightarrow \mathscr{G}$	$a_{\mathscr{N}\mathscr{G}}$	0.09204	0.25242	0.05851	0.05673
		(0.02138)	(0.03624)	(0.01802)	(0.01593)
ALACD Specifications					
Intercept (Buy after Buy)	$\nu_{1,1}$	0.88515	0.94459	0.74891	0.71708
		(0.00787)	(0.00823)	(0.00485)	(0.00412)
Intercept (Buy after Sell)	$\nu_{1,-1}$	0.97717	1.06670	0.73102	1.00472
		(0.00763)	(0.00822)	(0.00476)	(0.00485)
Cond. Duration (Buys)	α_1	0.72058	0.74241	0.71663	0.75346
		(0.00235)	(0.00233)	(0.00162)	(0.00161)
Lagged Duration (Buys)	β_1	0.12749	0.12651	0.13277	0.13293
		(0.00084)	(0.00089)	(0.00061)	(0.00067)
Signed Volume (Buys)	ζ_1	-0.03425	-0.03931	-0.04173	-0.02425
		(0.00116)	(0.00128)	(0.00073)	(0.00080)
Intercept (Sell after Buy)	$\nu_{-1,1}$	1.00762	1.04698	0.88945	0.85629
		(0.00768)	(0.00735)	(0.00506)	(0.00395)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.86296	0.81985	0.78397	0.61168
		(0.00768)	(0.00741)	(0.00513)	(0.00349)
Cond. Duration (Sells)	α_{-1}	0.73273	0.76932	0.70610	0.77900
		(0.00217)	(0.00198)	(0.00153)	(0.00148)
Lagged Duration (Sells)	β_{-1}	0.13190	0.12904	0.14799	0.11385
		(0.00085)	(0.00089)	(0.00065)	(0.00060)
Signed Volume (Sells)	ζ_{-1}	0.03114	0.02665	0.03767	0.02111
		(0.00114)	(0.00119)	(0.00075)	(0.00066)
Adjustment (good news)	τ_1	0.37853	0.38353	0.41662	0.31387
		(0.00519)	(0.00571)	(0.00354)	(0.00439)
Adjustment (bad news)	τ_{-1}	0.27443	0.41521	0.20973	0.71561
		(0.00763)	(0.01037)	(0.00441)	(0.00439)
Shape Parameters					
Buys	k_1	0.56258	0.55125	0.60824	0.55894
		(0.00054)	(0.00061)	(0.00042)	(0.00043)
Sells	k_{-1}	0.55727	0.54943	0.58440	0.58894
		(0.00054)	(0.00059)	(0.00043)	(0.00043)

Table 11.24: Estimation results for the PIN-HMM model for all German stocks in 2008. Figures in parentheses denote standard errors.

	Param.	BMW	CON	DAI	VOW
Transistion Probabilities					
$\mathscr{B} \longrightarrow \mathscr{G}$	$a_{\mathscr{B}\mathscr{G}}$	0.17253	0.06108	0.08408	0.12277
		(0.06001)	(0.04032)	(0.03138)	(0.04751)
$\mathscr{B} \longrightarrow \mathscr{N}$	$a_{\mathscr{B}\mathscr{N}}$	0.47208	0.47191	0.50491	0.42310
		(0.07850)	(0.07717)	(0.05669)	(0.07568)
$\mathscr{G} \longrightarrow \mathscr{B}$	$a_{\mathscr{G}\mathscr{B}}$	0.23216	0.34109	0.40317	0.25285
		(0.06835)	(0.12564)	(0.10661)	(0.09306)
$\mathscr{G} \longrightarrow \mathscr{N}$	$a_{\mathscr{G}\mathscr{N}}$	0.49692	0.53393	0.24181	0.51429
		(0.07917)	(0.13076)	(0.09389)	(0.10793)
$\mathscr{N} \longrightarrow \mathscr{B}$	$a_{\mathscr{N}\mathscr{B}}$	0.10785	0.10584	0.25507	0.11325
		(0.02567)	(0.02411)	(0.03726)	(0.02402)
$\mathscr{N} \longrightarrow \mathscr{G}$	$a_{\mathscr{N}\mathscr{G}}$	0.14765	0.06442	0.04867	0.06630
		(0.02994)	(0.01945)	(0.01974)	(0.01851)
ALACD Specifications					
Intercept (Buy after Buy)	$\nu_{1,1}$	1.27291	1.17698	1.20097	1.03030
		(0.01141)	(0.01716)	(0.00773)	(0.00813)
Intercept (Buy after Sell)	$\nu_{1,-1}$	1.43564	1.47757	1.05743	1.51429
		(0.01160)	(0.01913)	(0.00756)	(0.00924)
Cond. Duration (Buys)	α_1	0.58390	0.83083	0.56158	0.62630
		(0.00317)	(0.00410)	(0.00214)	(0.00358)
Lagged Duration (Buys)	β_1	0.15037	0.09836	0.17147	0.13141
		(0.00101)	(0.00194)	(0.00076)	(0.00098)
Signed Volume (Buys)	ζ_1	-0.06027	-0.01296	-0.08446	-0.03610
		(0.00185)	(0.00260)	(0.00121)	(0.00170)
Intercept (Sell after Buy)	$\nu_{-1,1}$	1.49497	1.81258	1.04696	1.53416
		(0.01140)	(0.02931)	(0.00777)	(0.00951)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	1.20659	1.38978	1.19655	1.10389
		(0.01119)	(0.02462)	(0.00797)	(0.00861)
Cond. Duration (Sells)	α_{-1}	0.59057	0.75252	0.57885	0.64012
		(0.00312)	(0.00665)	(0.00211)	(0.00324)
Lagged Duration (Sells)	β_{-1}	0.14920	0.10174	0.16537	0.13459
		(0.00100)	(0.00205)	(0.00075)	(0.00098)
Signed Volume (Sells)	ζ_{-1}	0.04799	0.01766	0.08236	0.05165
		(0.00180)	(0.00356)	(0.00125)	(0.00175)
Adjustment (good news)	τ_1	0.24190	0.45588	0.25916	0.56170
		(0.00829)	(0.02243)	(0.00871)	(0.00876)
Adjustment (bad news)	τ_{-1}	0.43404	0.73885	0.25916	0.42716
		(0.00728)	(0.02008)	(0.00421)	(0.00798)
Shape Parameters					
Buys	k_1	0.47977	0.45472	0.51266	0.49629
		(0.00052)	(0.00122)	(0.00041)	(0.00058)
Sells	k_{-1}	0.48076	0.44699	0.52008	0.48745
		(0.00052)	(0.00124)	(0.00042)	(0.00057)

Table 11.25: Estimation results for the PIN-HMM model for all German stocks in 2009. Figures in parentheses denote standard errors.

	Param.	BMW	CON	DAI	VOW
Transistion Probabilities					
$\mathscr{B} \longrightarrow \mathscr{G}$	$a_{\mathscr{B}\mathscr{G}}$	0.87526	**1**	**0**	0.08176
		(0.11665)	— — —	— — —	(0.03519)
$\mathscr{B} \longrightarrow \mathscr{N}$	$a_{\mathscr{B}\mathscr{N}}$	**0**	**0**	0.40294	0.46552
		— — —	— — —	(0.05041)	(0.06913)
$\mathscr{G} \longrightarrow \mathscr{B}$	$a_{\mathscr{G}\mathscr{B}}$	0.04812	**0**	**1**	0.07457
		(0.02369)	— — —	— — —	(0.05065)
$\mathscr{G} \longrightarrow \mathscr{N}$	$a_{\mathscr{G}\mathscr{N}}$	0.49539	0.57259	**0**	0.48084
		(0.05814)	(0.05688)	— — —	(0.09307)
$\mathscr{N} \longrightarrow \mathscr{B}$	$a_{\mathscr{N}\mathscr{B}}$	0.01830	0.00588	0.23243	0.19156
		(0.01053)	(0.00586)	(0.03359)	(0.03161)
$\mathscr{N} \longrightarrow \mathscr{G}$	$a_{\mathscr{N}\mathscr{G}}$	0.24026	0.26745	0.00619	0.06833
		(0.05007)	(0.03932)	(0.00617)	(0.01998)
ALACD Specifications					
Intercept (Buy after Buy)	$\nu_{1,1}$	1.28803	1.56902	1.14212	1.18730
		(0.01120)	(0.01716)	(0.00803)	(0.01351)
Intercept (Buy after Sell)	$\nu_{1,-1}$	1.39081	1.92768	1.10702	1.89979
		(0.01022)	(0.01828)	(0.00780)	(0.01691)
Cond. Duration (Buys)	α_1	0.56994	0.58141	0.59782	0.69012
		(0.00294)	(0.00439)	(0.00212)	(0.00438)
Lagged Duration (Buys)	β_1	0.14005	0.13365	0.16596	0.12348
		(0.00089)	(0.00137)	(0.00074)	(0.00139)
Signed Volume (Buys)	ζ_1	-0.07406	-0.06912	-0.07430	-0.01129
		(0.00169)	(0.00306)	(0.00132)	(0.00248)
Intercept (Sell after Buy)	$\nu_{-1,1}$	1.36323	2.22185	1.02409	2.14798
		(0.00990)	(0.01761)	(0.00757)	(0.02282)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	1.24294	1.24331	1.20999	1.34792
		(0.00974)	(0.01557)	(0.00781)	(0.01932)
Cond. Duration (Sells)	α_{-1}	0.57851	0.60009	0.60444	0.64030
		(0.00286)	(0.00415)	(0.00220)	(0.00572)
Lagged Duration (Sells)	β_{-1}	0.14424	0.14594	0.15547	0.11045
		(0.00089)	(0.00138)	(0.00073)	(0.00150)
Signed Volume (Sells)	ζ_{-1}	0.06952	0.01360	0.08175	0.01587
		(0.00164)	(0.00291)	(0.00126)	(0.00327)
Adjustment (good news)	τ_1	0.31726	0.45941	0.71607	0.84885
		(0.00591)	(0.00990)	(0.01725)	(0.01367)
Adjustment (bad news)	τ_{-1}	0.45106	1.97225	0.36385	0.62284
		(0.01203)	(0.04076)	(0.00415)	(0.01300)
Shape Parameters					
Buys	k_1	0.49041	0.44398	0.51762	0.45725
		(0.00051)	(0.00073)	(0.00043)	(0.00083)
Sells	k_{-1}	0.48962	0.43424	0.52688	0.45379
		(0.00051)	(0.00071)	(0.00044)	(0.00094)

Table 11.26: Estimation results for the PIN-HMM model for all German stocks in 2010. Figures in parentheses denote standard errors.

	BMW	CON	DAI	VOW
Bad	0.09044	0.00398	0.10580	0.21187
Good	0.20894	0.37350	0.21519	0.07543
No	0.70062	0.62252	0.67902	0.71270

Table 11.27: Stationary distribution of trading days' conditions for German equities in 2007.

	BMW	CON	DAI	VOW
Bad	0.08781	0.03895	0.16719	0.05041
Good	0.17563	0.38331	0.13748	0.11713
No	0.73657	0.57774	0.69533	0.83246

Table 11.28: Stationary distribution of trading days' conditions for German equities in 2008.

	BMW	CON	DAI	VOW
Bad	0.17196	0.19068	0.31735	0.19142
Good	0.17329	0.06789	0.08636	0.09252
No	0.65475	0.74143	0.59629	0.71606

Table 11.29: Stationary distribution of trading days' conditions for German equities in 2009.

	BMW	CON	DAI	VOW
Bad	0.03156	0.00396	0.37050	0.24110
Good	0.33212	0.32184	0.00387	0.11474
No	0.63632	0.67420	0.62563	0.64416

Table 11.30: Stationary distribution of trading days' conditions for German equities in 2010.

	2007	2008	2009	2010
BMW	0.16366	0.19180	0.10958	0.14409
CON	0.19486	0.19056	0.18722	0.18451
DAI	0.12409	0.22384	0.12441	0.30972
VOW	0.22452	0.16091	0.24329	0.32168

Table 11.31: Proportion informed buys occupy of the total number of buys on information events driven by positive private news in the context of our dynamic PIN-HMM model (German equities).

	2007	2008	2009	2010
BMW	0.18347	0.14181	0.18834	0.19816
CON	0.52752	0.20398	0.28126	0.57532
DAI	0.20001	0.11535	0.12609	0.17445
VOW	0.16987	0.34391	0.18797	0.24621

Table 11.32: Proportion informed sells occupy of the total number of sells on information events driven by negative private news in the context of our dynamic PIN-HMM model (German equities).

	2007	2008	2009	2010
BMW	0.41023	0.41343	0.40893	0.35868
CON	0.37509	0.44936	0.53212	0.45693
DAI	0.42801	0.45847	0.31352	0.35511
VOW	0.45091	0.61007	0.49631	0.59587

Table 11.33: Proportion informed buys occupy of the total number of buys on information events driven by positive private news in the context of the static EHO model (German equities). Yearly series of quarterly estimates of ϵ_b and μ are summed up to achieve a better comparability with the estimates in table 11.31.

	2007	2008	2009	2010
BMW	0.39197	0.38784	0.41336	0.34994
CON	0.38179	0.45285	0.55083	0.46104
DAI	0.40156	0.41738	0.32094	0.33323
VOW	0.44139	0.64157	0.50403	0.63565

Table 11.34: Proportion informed sells occupy of the total number of sells on information events driven by negative private news in the context of the static EHO model (German equities). Yearly series of quarterly estimates of ϵ_s and μ are summed up to achieve a better comparability with the estimates in table 11.32.

11.3 Probability of Informed Trading

This section offers plots of the probabilities of informed trading driven by positive and negative private news in combination with corresponding state probabilities.[125] [126] Despite the counterintuitive results for the parameters incorporated in the autoregressive specification for the conditional expected waiting times of buys and sells in the approach by Tay, Ting, Tse, and Warachka (2009), we decided to include the corresponding plots for the probability of informed trading and conditions of trading days for the sake of completeness. Moreover, the problems in the modeling of probabilities of trading days' states become even clearer by visualization in direct comparison with the results of the Hidden Markov method in our dynamic model.

All figures have in common that the state probabilities which belong to the PIN-ALACD model scatter by far more than their counterparts from the PIN-HMM model do.[127] As explained in section 6.1, for most trading days of all symbols and year, the model by Tay, Ting, Tse, and Warachka (2009) is not sure about their conditions. We see probabilities for good-news and bad-news trading days which are both substantial higher than 0. Hence, for the majority of information events the model is not sure about the direction of private information.

The overall picture for our PIN-HMM model is contrary. For most equities and years the state probabilities are reminiscent of visualizations of binary variables. Therefore, our hidden Markov approach grants high (pseudo-)sureness about the conditions of trading days. This yields separated measures of the probability of informed trading, $\text{PIN}_{\mathscr{G},d}$ and $\text{PIN}_{\mathscr{B},d}$, being very close or even identical to PIN_d on most information events because state probabilities clearly identify directions of private news.

Most estimates of $\text{PIN}_{\mathscr{G},d}$ and $\text{PIN}_{\mathscr{B},d}$ in the dynamic models do not exceed a value of 20%. However, the PIN-HMM model reports also some very high estimates for both probabilities of informed trading. Exemplary, on April 30, 2008 the estimate of $\text{PIN}_{\mathscr{G},d}$ is close to 45% for TM. This means that according to our dynamic model nearly every second transaction on this trading day was initiated by an insider driven by positive private information. Only three information events are labelled as bad-news trading days for F in 2008 but the estimates of $\text{PIN}_{\mathscr{B},d}$ lie above 40% on each of them. Furthermore, the average expected number of orders by informed market participants on information events seems to be independent of the year under consideration for most equities. There is no substantial change in the level of the probability of informed trading over the time range from 2007 to 2010.

Interesting dates for analyzing the behavior of the probabilities of informed trading, we already mentioned in sections 10.1 and 10.2, are the beginnings of the car scrappage programs in the

[125] The commonly used measure for PIN_d as relation of expected number of trades initiated by insiders to the expected number of total trades on trading day d (see equation (2.1)) can be obtained by summing $\text{PIN}_{\mathscr{G},d}$ and $\text{PIN}_{\mathscr{B},d}$ (see equation (6.78)).

[126] Since the sum of all three potential trading day conditions always equals unity, it is sufficient to plot only probabilities of good- and bad-news conditions.

[127] This section contains a large number of figures. To improve the readability, all figures are placed near the end of the section after their interpretation.

United States and Germany. The American *CARS* program started on July 1, 2009, whereas the German program was introduced on January 13, 2009. Results for the US equities in 2009 are shown in figures 11.19 – 11.23, whereas we have data for GM only until June 1, 2009. Figures 11.15 – 11.18 and 11.24 – 11.27 display estimates of $\text{PIN}_{\mathcal{G},d}$ and $\text{PIN}_{\mathcal{B},d}$ as well as the probabilities of good- and bad-news states for the German symbols in 2008 and 2009, respectively.

We can see a clustering of information events from April to early August for F in 2009 in the PIN-HMM panel. The direction of private information switches multiple times in this time span. Most corresponding values of $\widehat{\text{PIN}}_{\mathcal{G},d}$ and $\widehat{\text{PIN}}_{\mathcal{B},d}$ returned by the PIN-ALACD model are smaller than in our dynamic approach. As we can see in table 11.5, there is a boundary solution for the estimate of δ_4. This implies that the volume of buys is ignored in decisions about the direction of private information. It is arguable whether to rely on results returned by the PIN-ALACD model for such constellations of estimates.

The remaining US symbols, HMC, JCI and TM, show an increased level of insider trading in the first half of 2009 for both dynamic models. From the figures belonging to the hidden Markov approach we can see a surplus of trading days on which insiders participate in market activities and solely fulfill sell orders. The magnitudes of $\widehat{\text{PIN}}_{\mathcal{G},d}$ and $\widehat{\text{PIN}}_{\mathcal{B},d}$ are similar on information events initiated by either positive or negative private news.

The PIN-ALACD model is not able to the clearly identify the direction of private information and therefore shows estimates of the probabilities of informed trading driven by good- and bad-news which both differ from 0. For the manufacturers HMC and TM we see higher values of $\widehat{\text{PIN}}_{\mathcal{G},d}$ throughout the whole year. The supplier JCI offers values of $\widehat{\text{PIN}}_{\mathcal{B},d}$ and $\widehat{\text{PIN}}_{\mathcal{G},d}$ which are very similar.

German manufacturers BMW, DAI and VOW exhibit high proportions of information events from September to the end of the year 2008 for both approaches. However, the amount of information events decreases towards the end of the year. All have in common that December consists of only very few trading days which reside in either good- or bad-news condition. To be more precise, the insider trading for the VOW equity is negligible in December for the PIN-HMM model, and in November and December for the PIN-ALACD model. In the time range from September to November, PIN-ALACD delivers higher values of $\widehat{\text{PIN}}_{\mathcal{B},d}$ for BMW and DAI which is contradictory to the results of the PIN-HMM model where the probability of informed trading driven by good news is often the preferred type of information event. The results of both models also diverge for the VOW equity. While $\widehat{\text{PIN}}_{\mathcal{G},d}$ dominates throughout the whole year of 2008 and the probability of bad-news trading is always extremely small, we see both types of information events in the PIN-HMM panel and $\widehat{\text{PIN}}_{\mathcal{B},d}$ is substantially higher than $\widehat{\text{PIN}}_{\mathcal{G},d}$. CON, which is a supplier for the automobile industry, shows only a few dates labelled as information events by both dynamic approaches from September to December.

In the beginning of 2009, which covers the official introduction of the German scrappage program, we only see a high amount of trading days identified as information events by PIN-HMM for VOW. The remaining manufacturers, BMW and DAI, show increased insider training in the second quarter of the year, while CON exhibits high proportions of insiders since September. Relative high levels of $\text{PIN}_{\mathcal{G},d}$ for BMW and $\text{PIN}_{\mathcal{B},d}$ for CON in the first months of 2009 are

reported by the PIN-ALACD model. The small estimates of $\text{PIN}_{\mathscr{G},d}$ and $\text{PIN}_{\mathscr{B},d}$ for DAI in 2009 are remarkable. Those values are always smaller than 10% in both dynamic models. For VOW, the estimates for the parameters δ_3 and δ_4 in the PIN-ALACD model equal 0 in 2009 and therefore the probabilities of good-news and bad-news states are identical on each trading day as shown in equation (11.1).

The results of both models in the context of the scrappage programs can be interpreted in a way that one can think of insiders anticipating the influences of the releases. However, the two programs may have been one of several factors triggering informed traders to become active market attendees. Thinking of the scrappage programs as the only relevant events would be too narrow. There are much more special events in the automobile industry which may influence the decisions of insiders and for which it may be interesting to analyze the probabilities of informed trading and conditions of trading days around corresponding dates. For example, the five biggest auto shows in the US: *Chicago Auto Show*, *North American International Auto Show*, *San Francisco International Auto Show*, *New York International Auto Show* and *Grand National Roadster Show* may also have a huge impact on the price of a symbol. Therefore information-based trading can happen due to private news concerning these events. Also during the German counterpart, *International Motor Show (IAA)*, new cars are often shown to the public for the first time which can influence the price of an equity to a huge degree. Hence, such conventions state good opportunities for insiders to anticipate and harness their private information for profit. Additional interesting trading periods were already mentioned and discussed in chapter 10. Beside the introduction of the scrappage programs in the United States in Germany, there is the price bubble for VOW in October 2008 and the delisting of GM in June 2009. Further typical events which may be preferred by information-based traders are *M&A*'s or earning announcements.

However, we cannot match all of those events with estimates of the probabilities of informed trading and trading days' states in this work. To reach this goal is too ambitious and would go beyond the scope of this thesis. The main focus lies on the theory and the fast and stable implementation of estimation routines for the probability of informed trading as well as the introduction of our new dynamic approach. However, analyses of the behavior of estimates of the probability of informed trading utilizing the new PIN-HMM model around certain market events of interest may be conducted in subsequent works.

Figure 11.1: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for F in 2007.

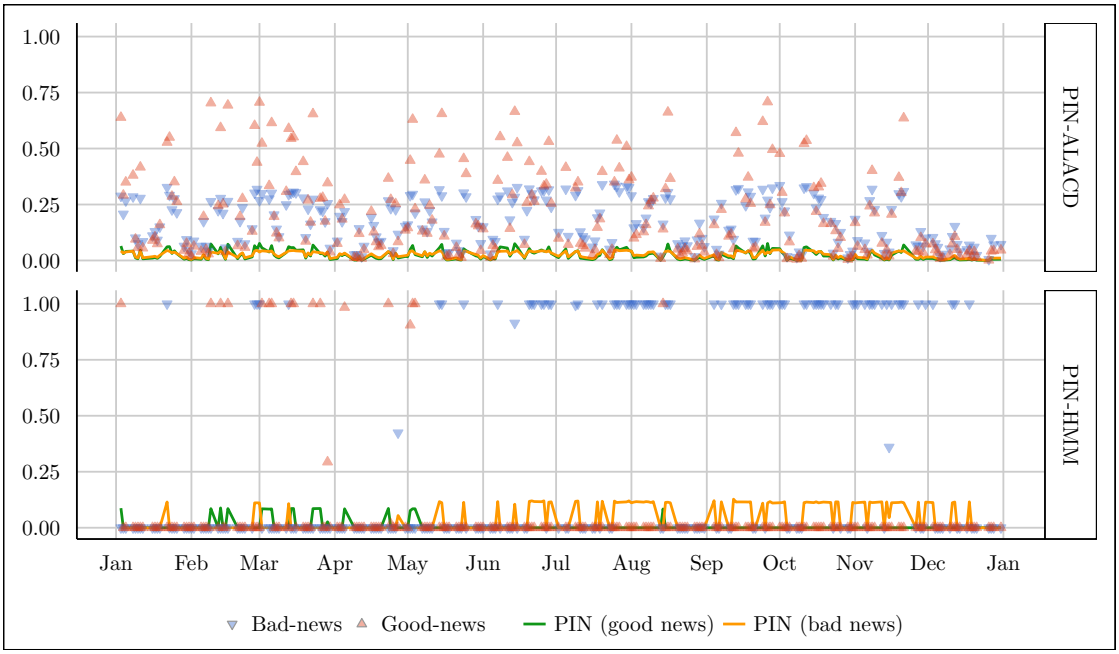

Figure 11.2: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for GM in 2007.

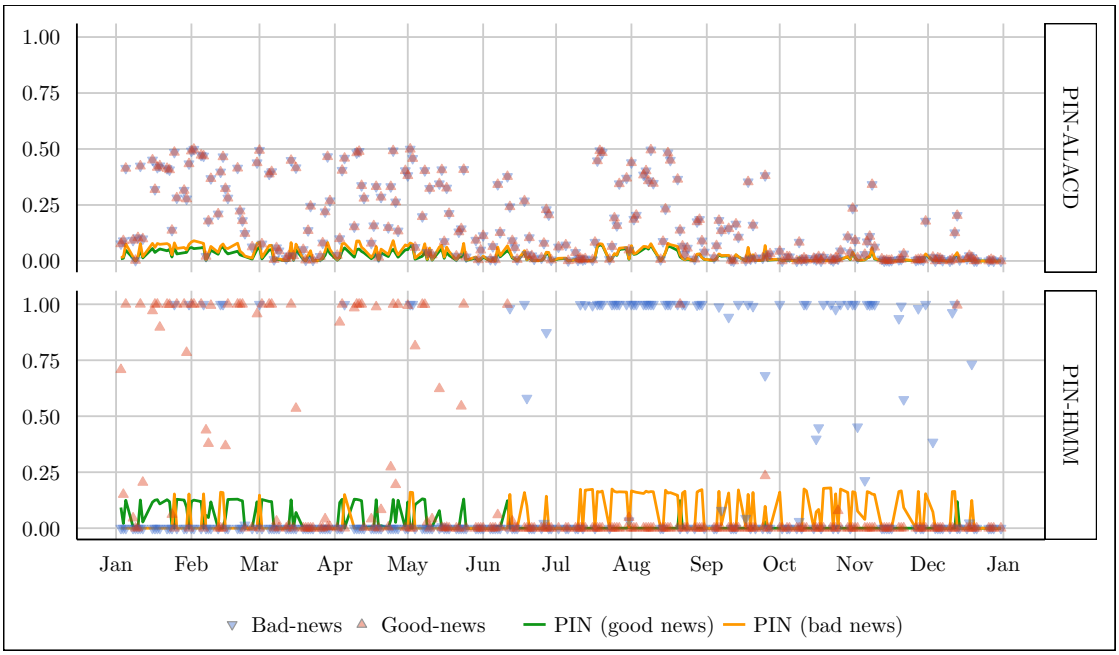

Figure 11.3: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for HMC in 2007.

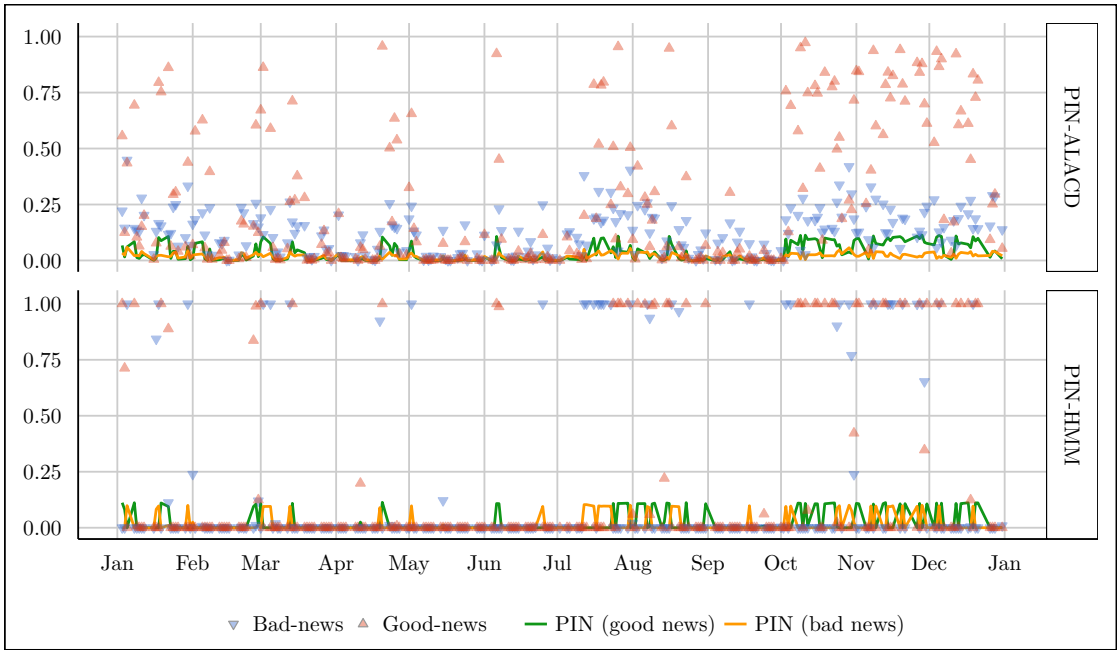

Figure 11.4: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for JCI in 2007.

Figure 11.5: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for TM in 2007.

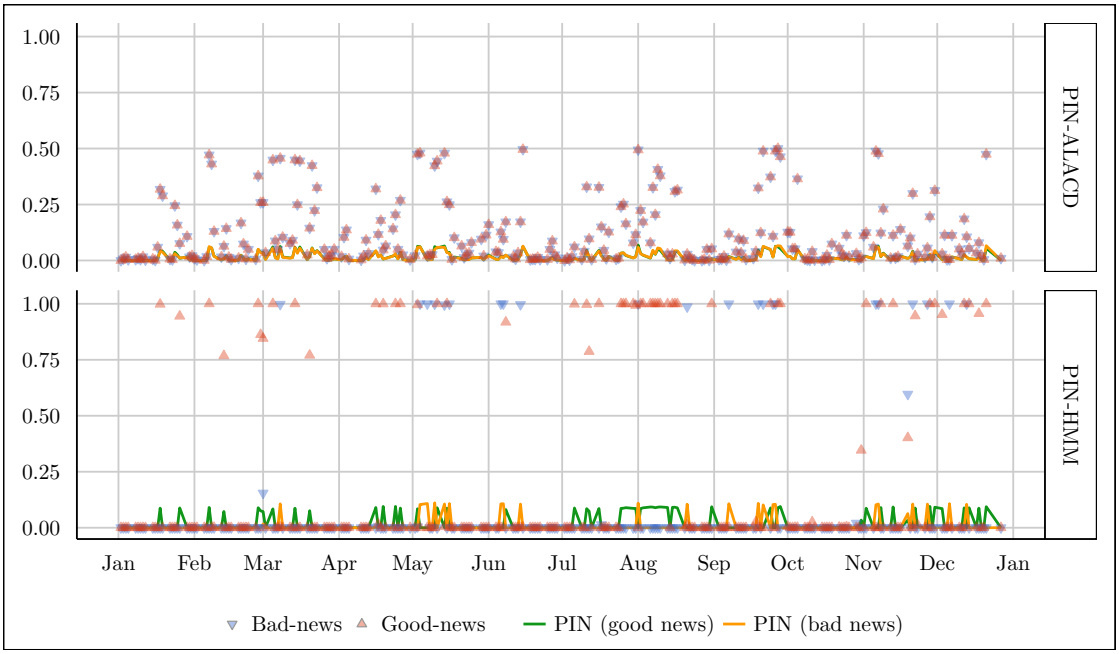

Figure 11.6: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for BMW in 2007.

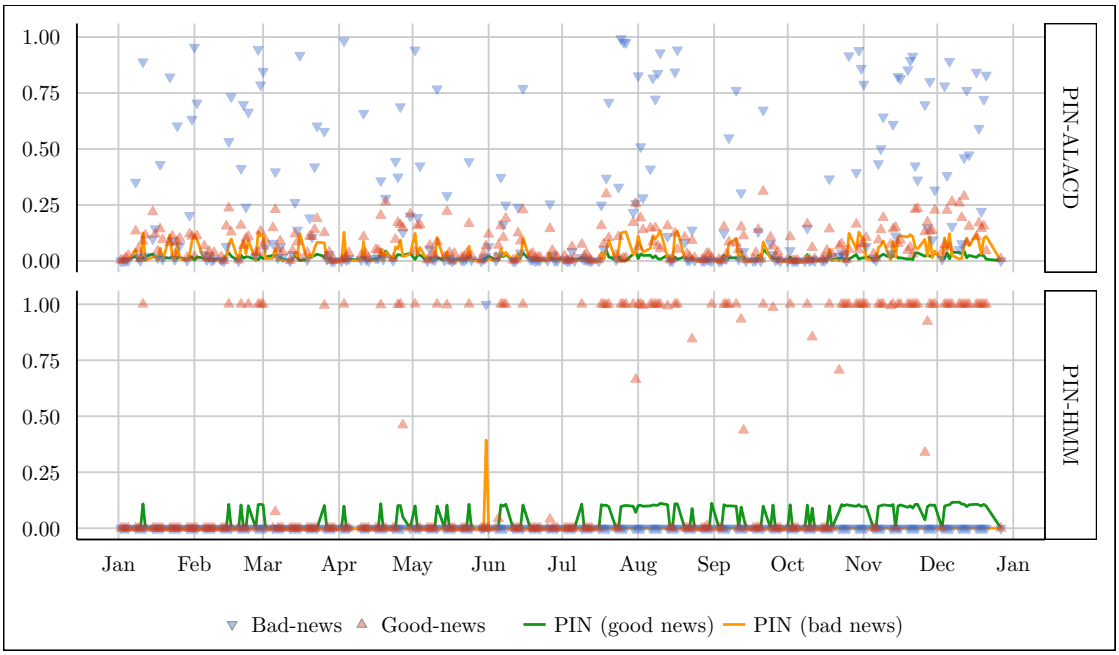

Figure 11.7: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for CON in 2007.

Figure 11.8: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for DAI in 2007.

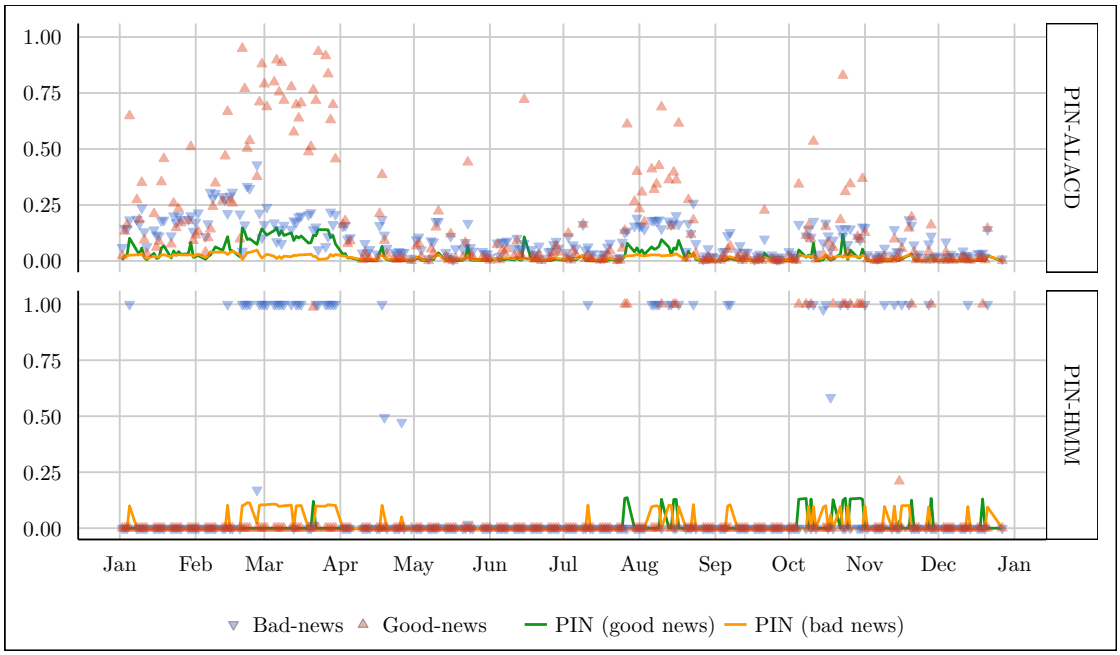

Figure 11.9: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for VOW in 2007.

Figure 11.10: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for F in 2008.

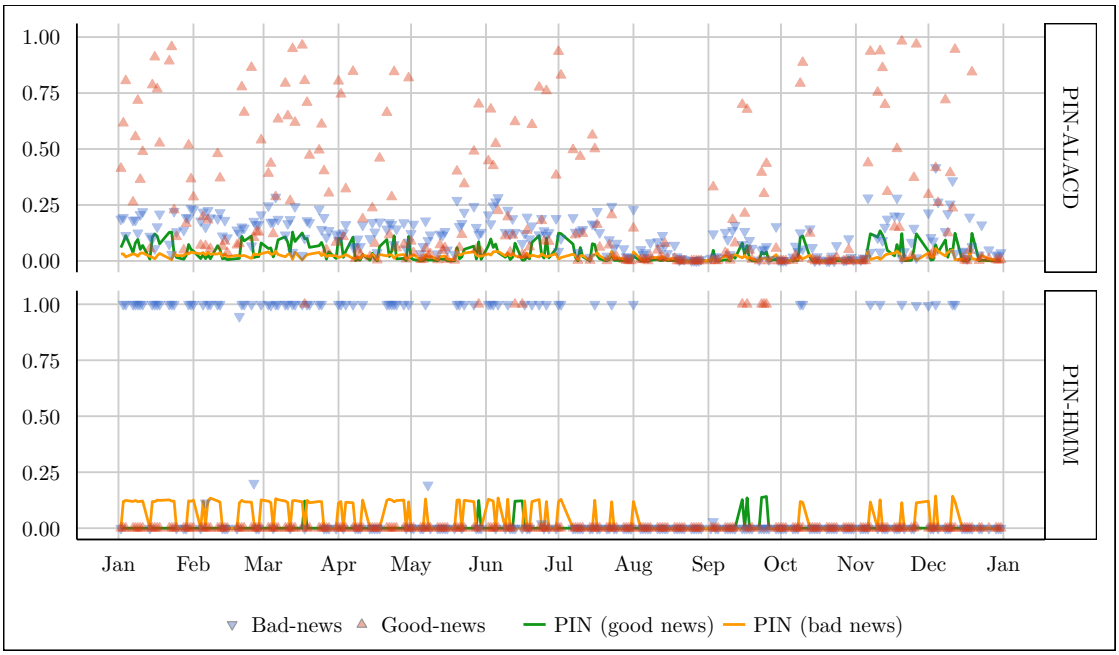

Figure 11.11: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for GM in 2008.

Figure 11.12: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for HMC in 2008.

Figure 11.13: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for JCI in 2008.

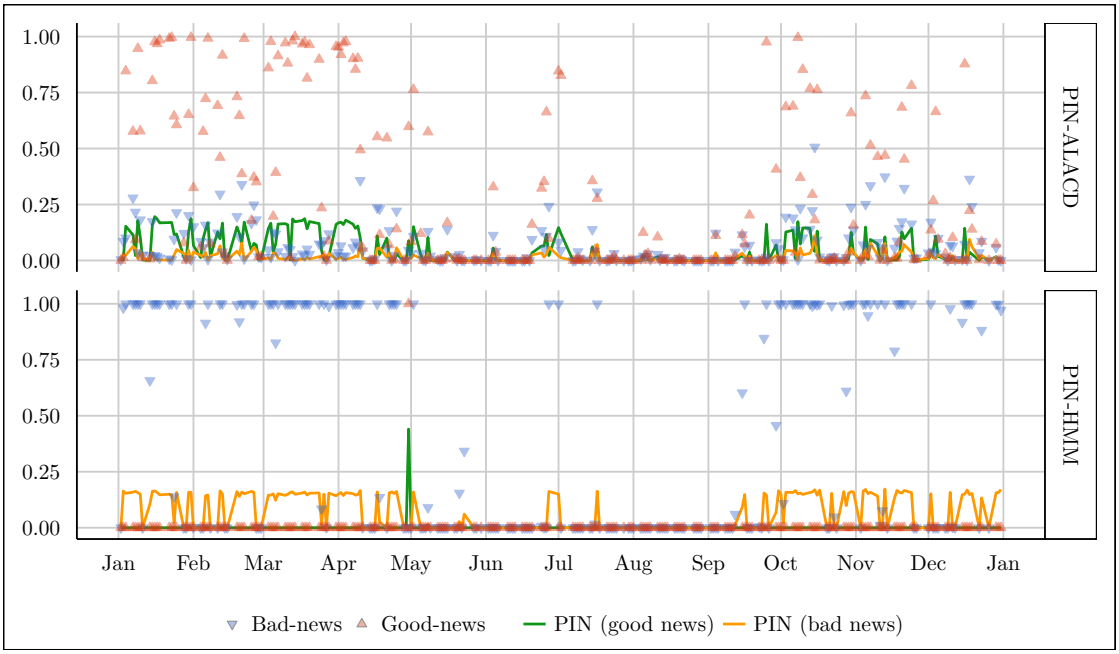

Figure 11.14: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for TM in 2008.

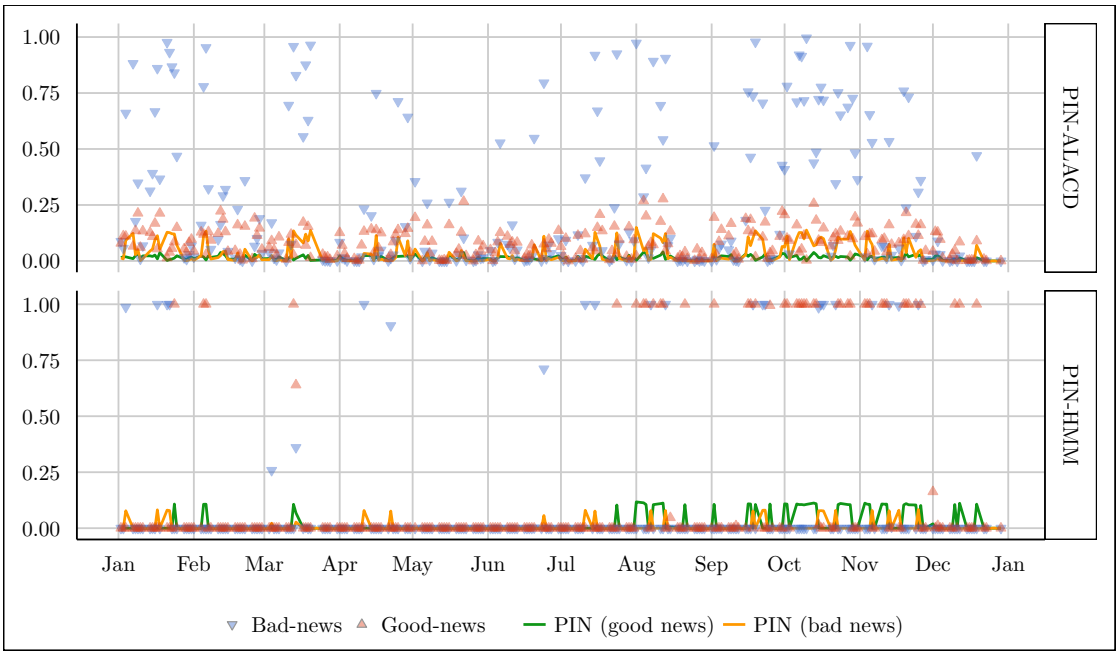

Figure 11.15: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for BMW in 2008.

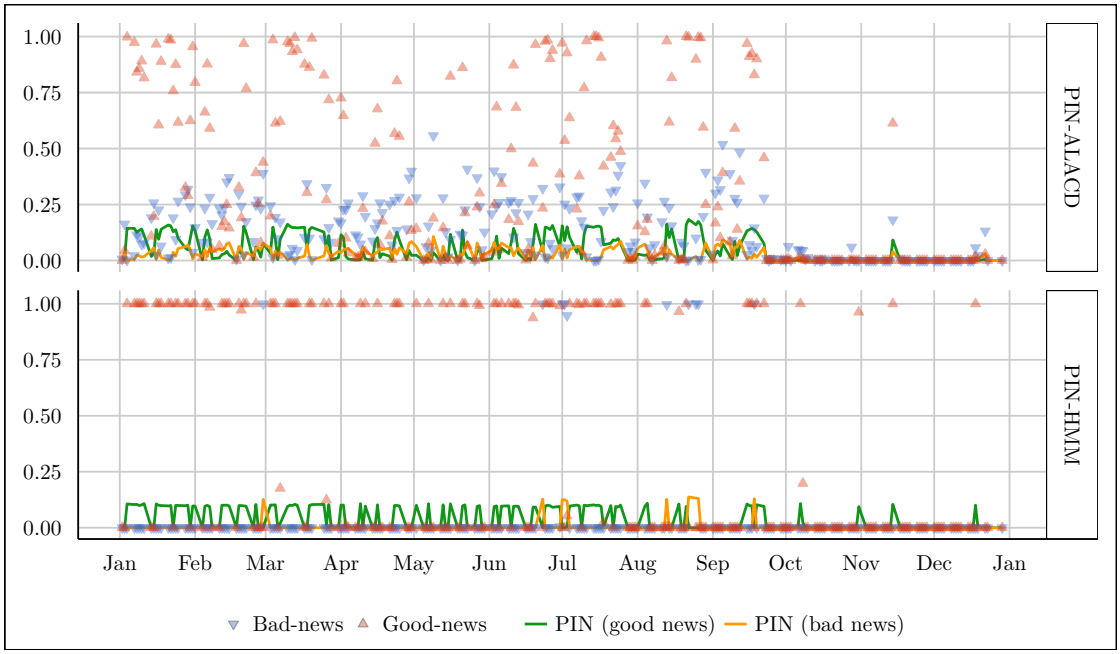

Figure 11.16: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for CON in 2008.

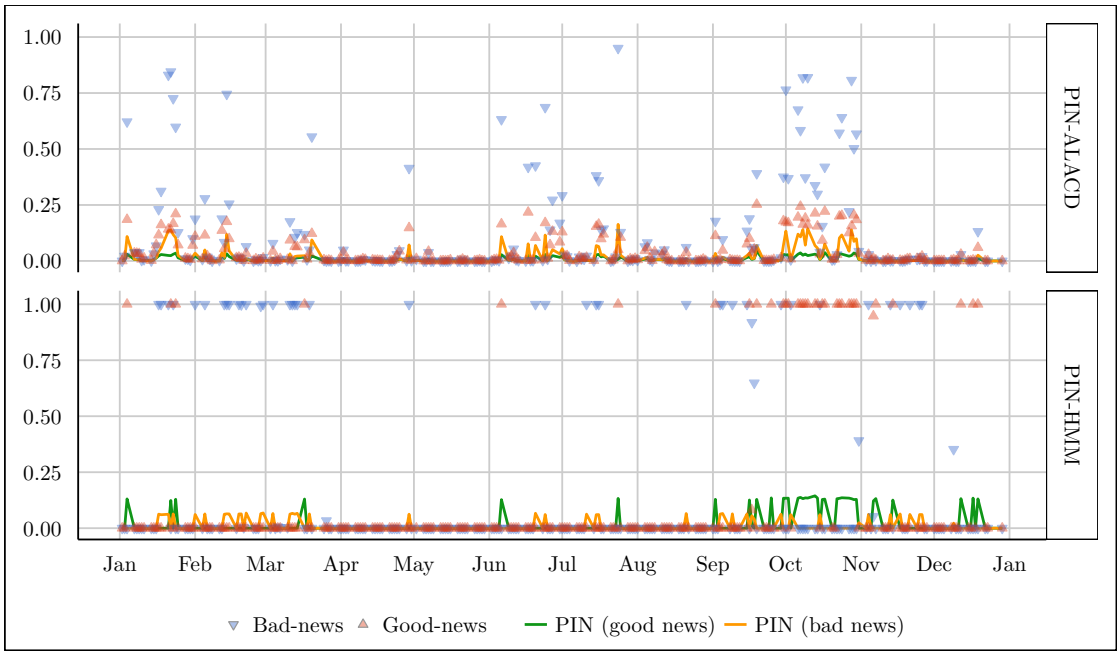

Figure 11.17: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for DAI in 2008.

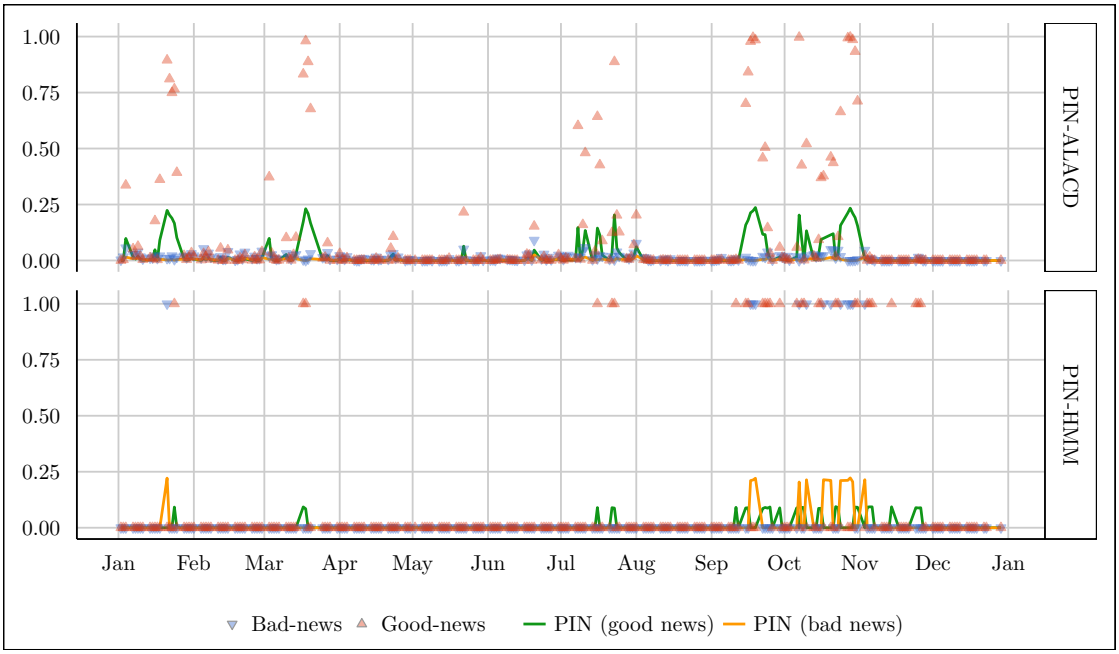

Figure 11.18: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for VOW in 2008.

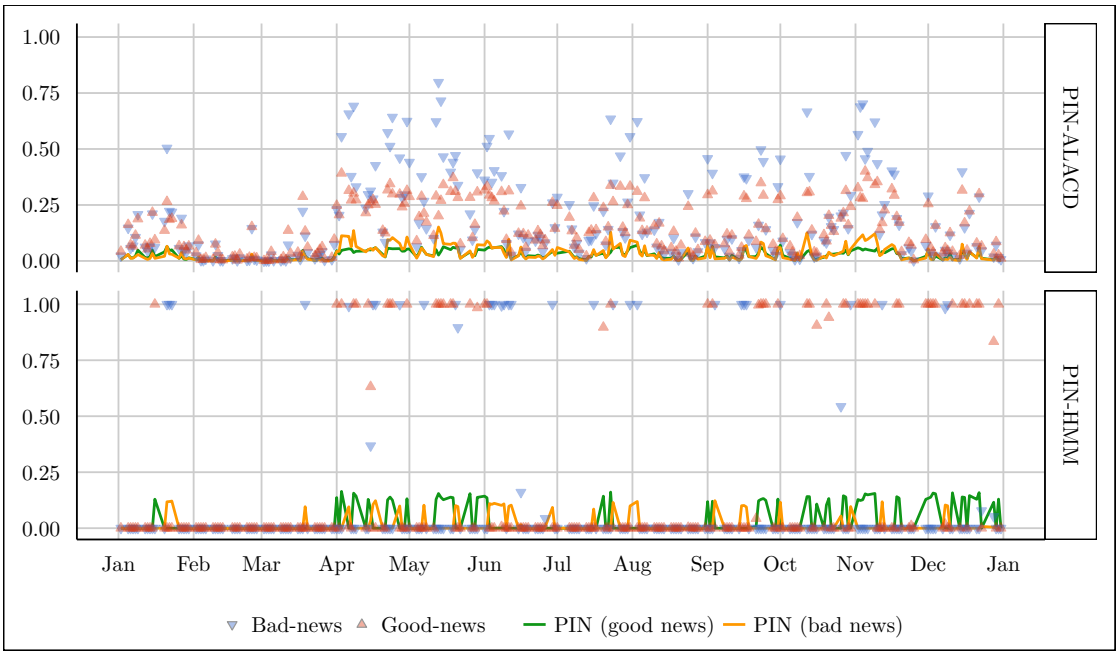

Figure 11.19: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for F in 2009.

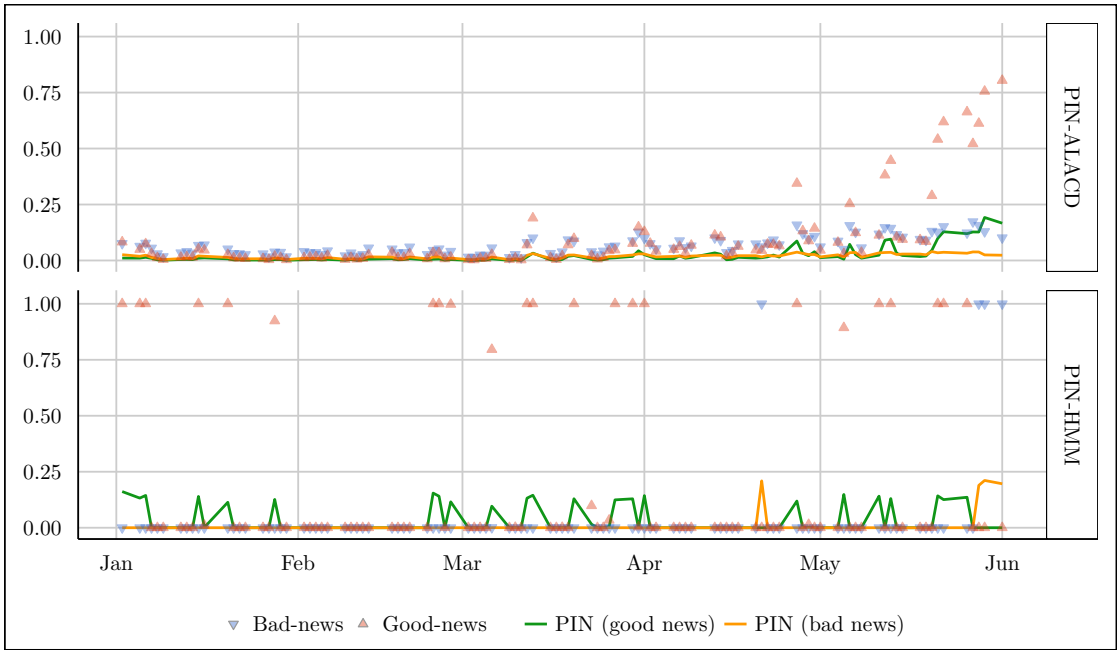

Figure 11.20: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for GM in 2009.

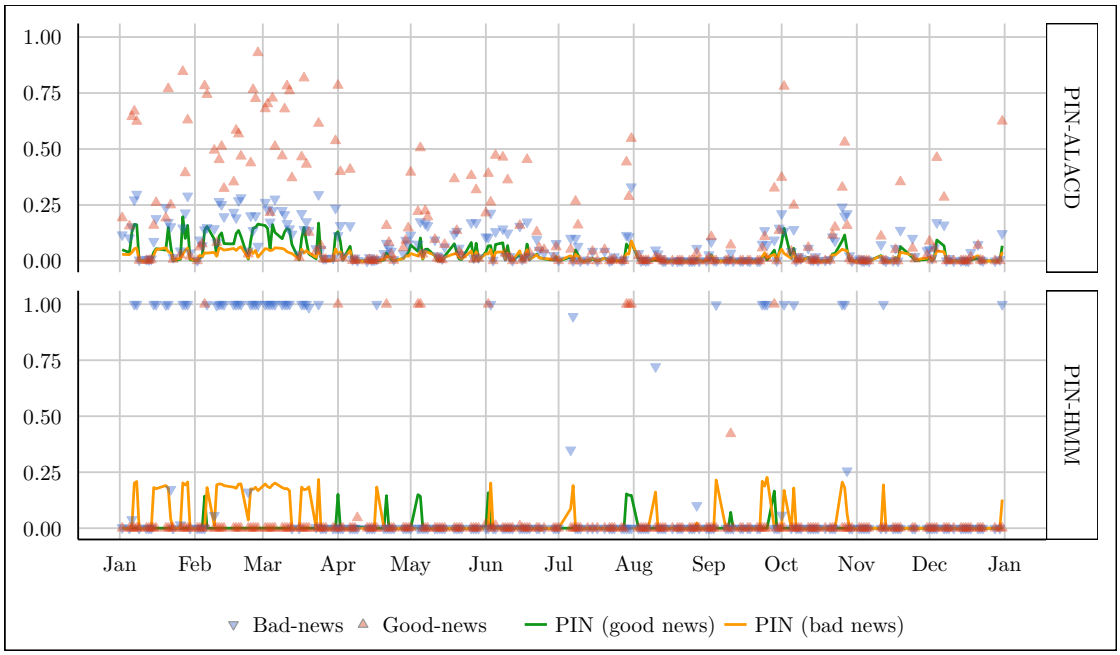

Figure 11.21: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for HMC in 2009.

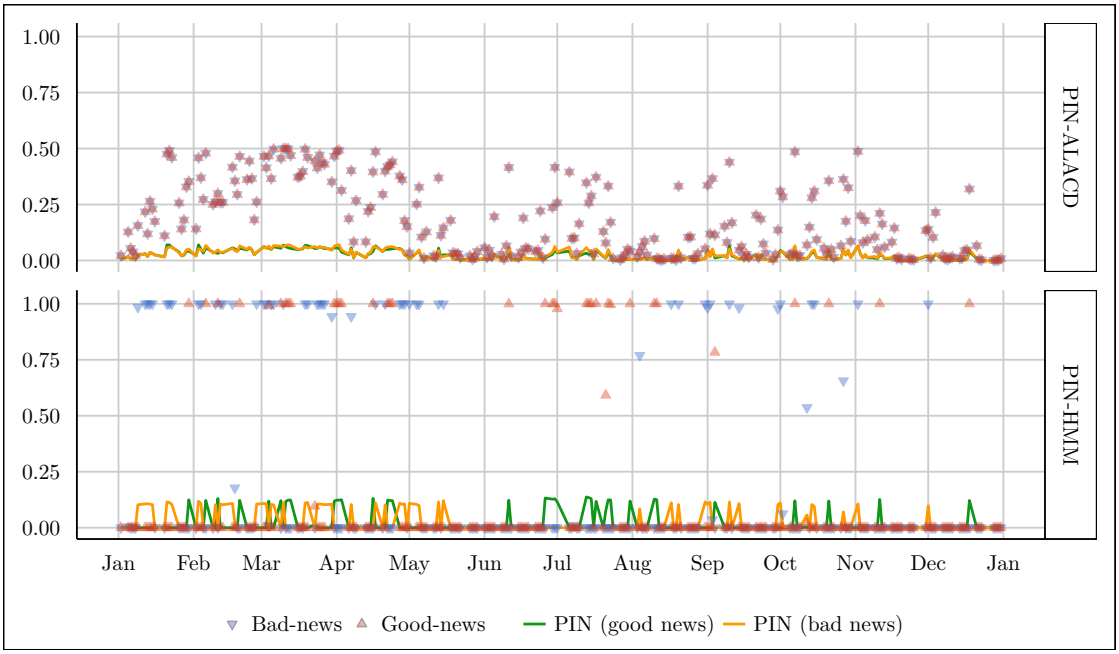

Figure 11.22: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for JCI in 2009.

Figure 11.23: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for TM in 2009.

Figure 11.24: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for BMW in 2009.

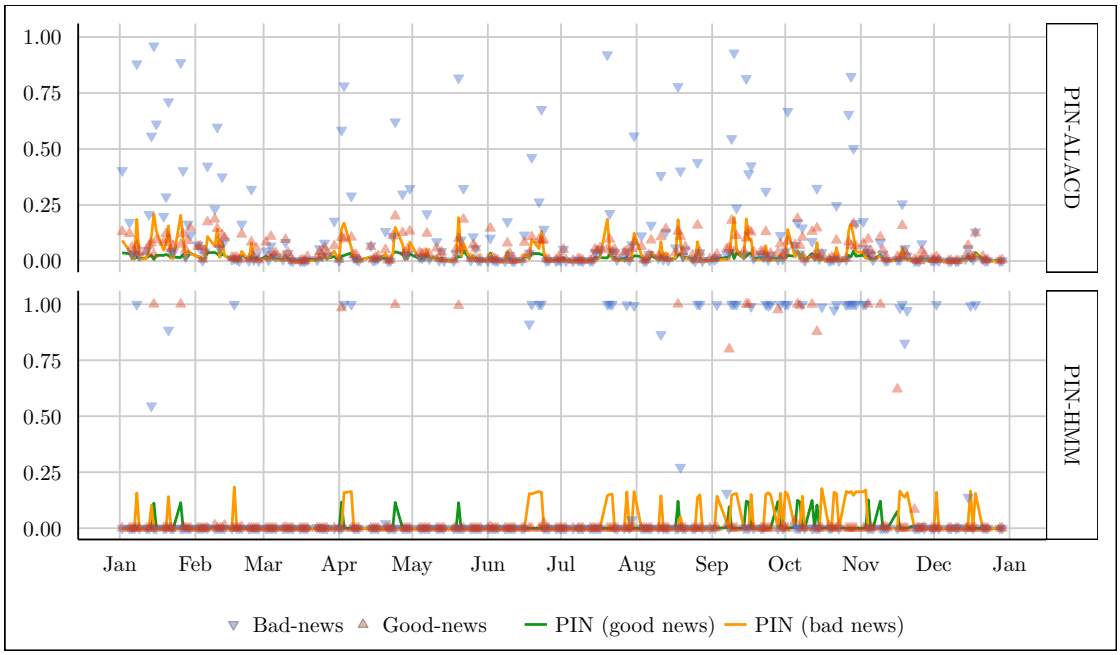

Figure 11.25: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for CON in 2009.

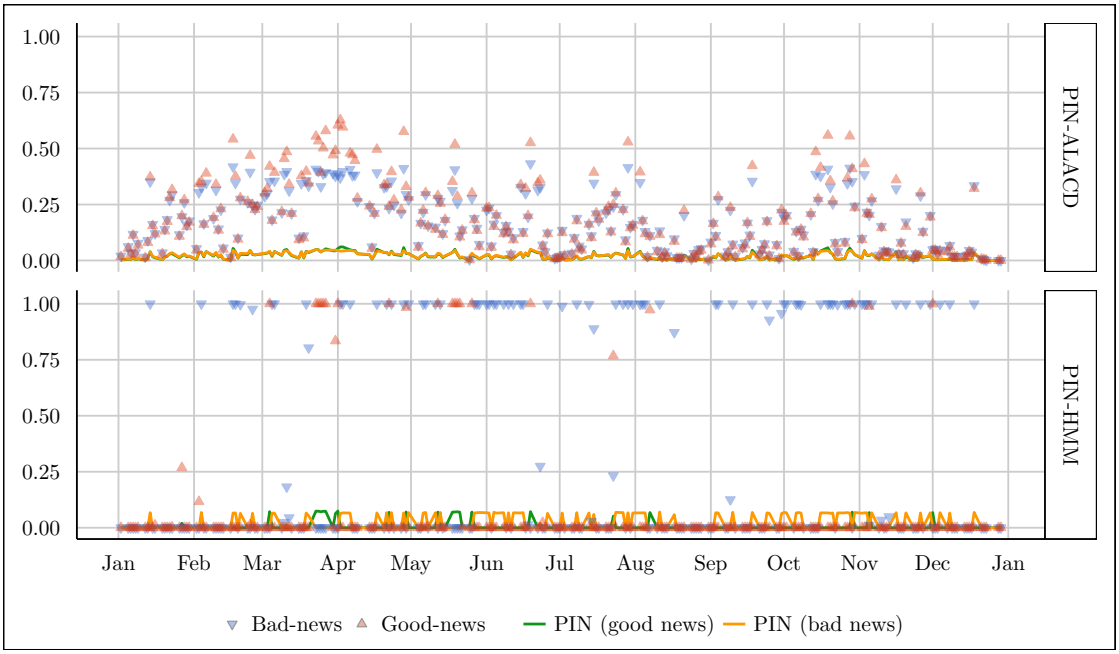

Figure 11.26: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for DAI in 2009.

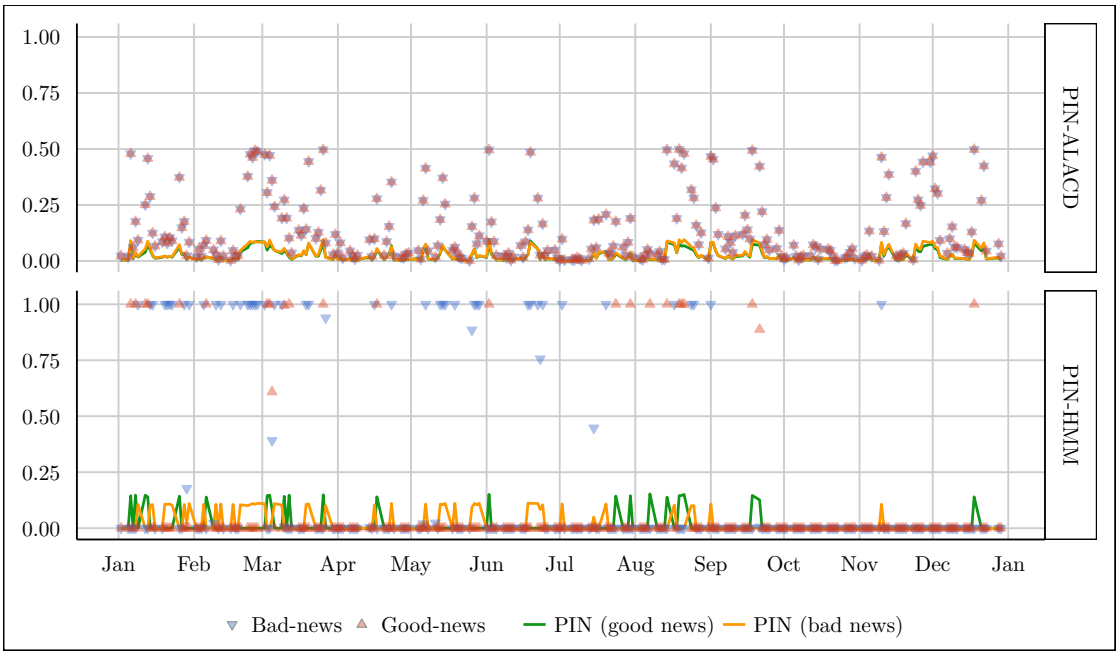

Figure 11.27: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for VOW in 2009.

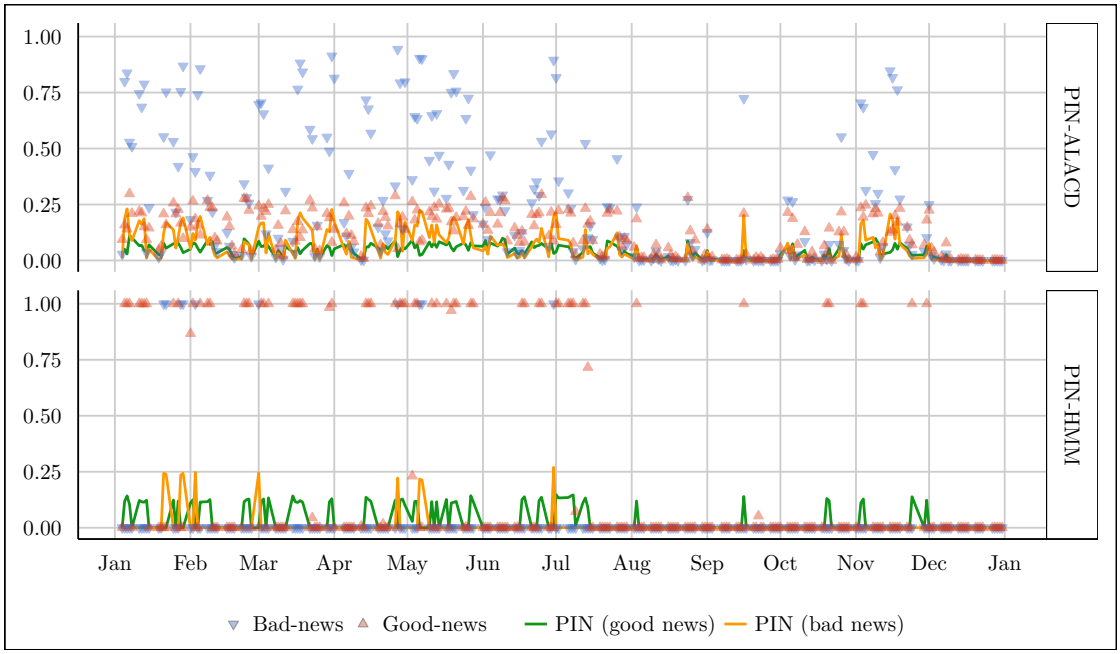

Figure 11.28: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for F in 2010.

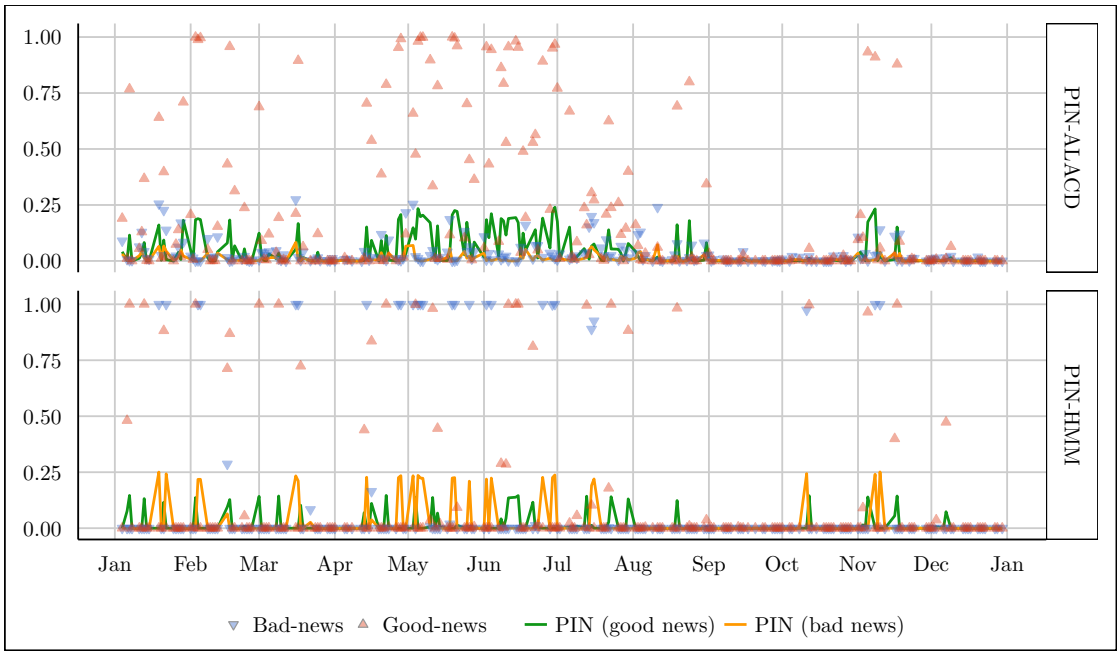

Figure 11.29: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for HMC in 2010.

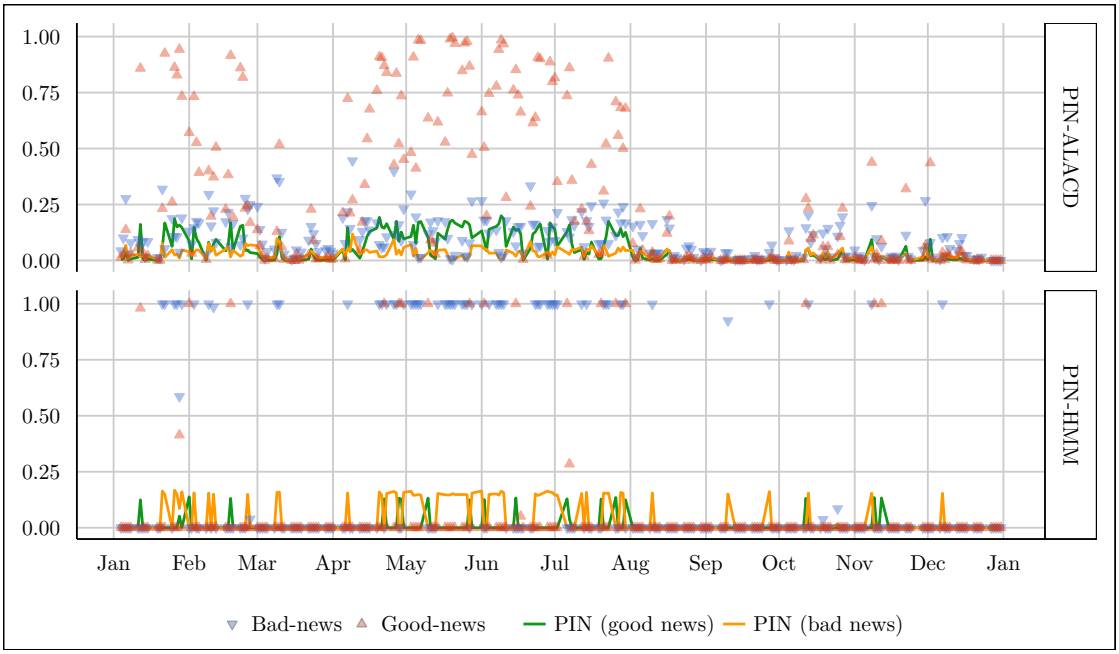

Figure 11.30: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for JCI in 2010.

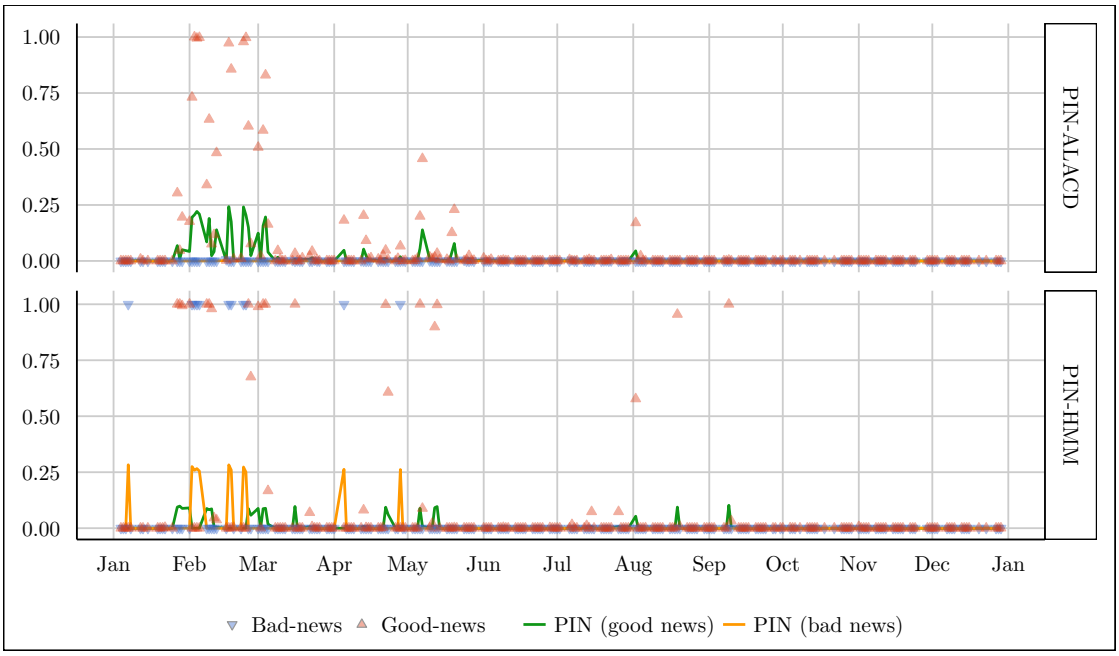

Figure 11.31: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for TM in 2010.

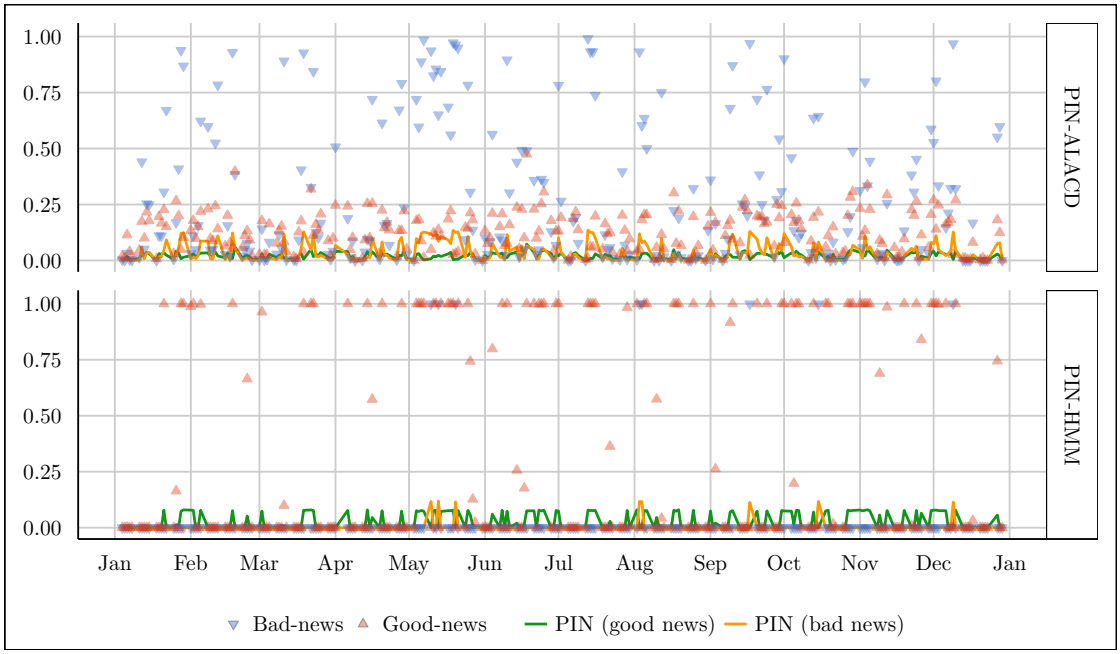

Figure 11.32: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for BMW in 2010.

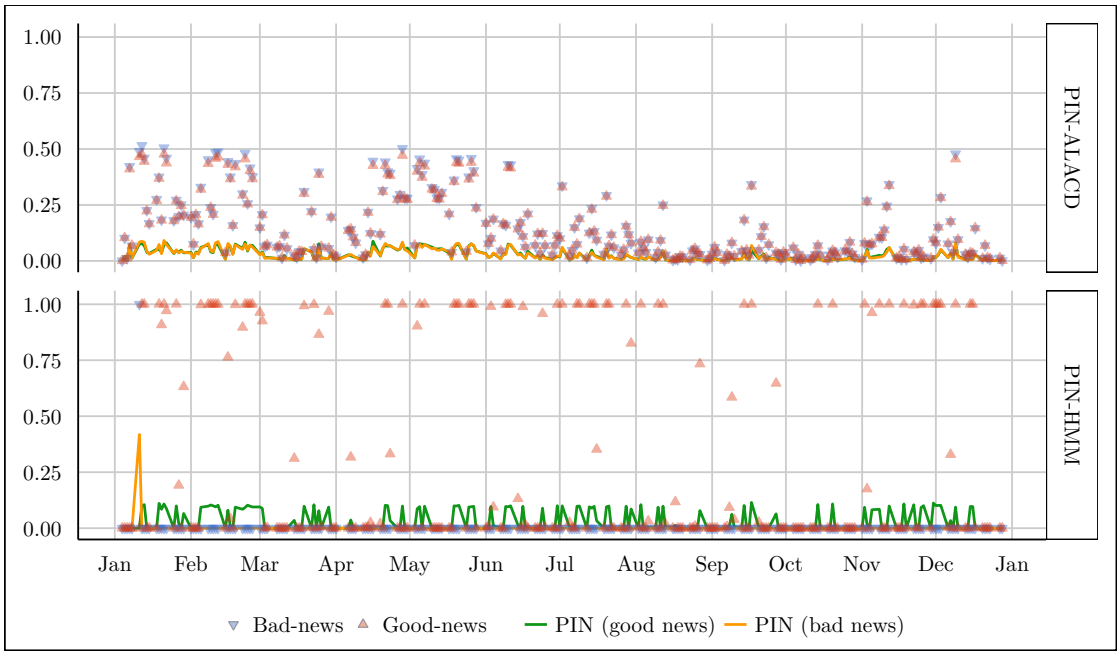

Figure 11.33: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for CON in 2010.

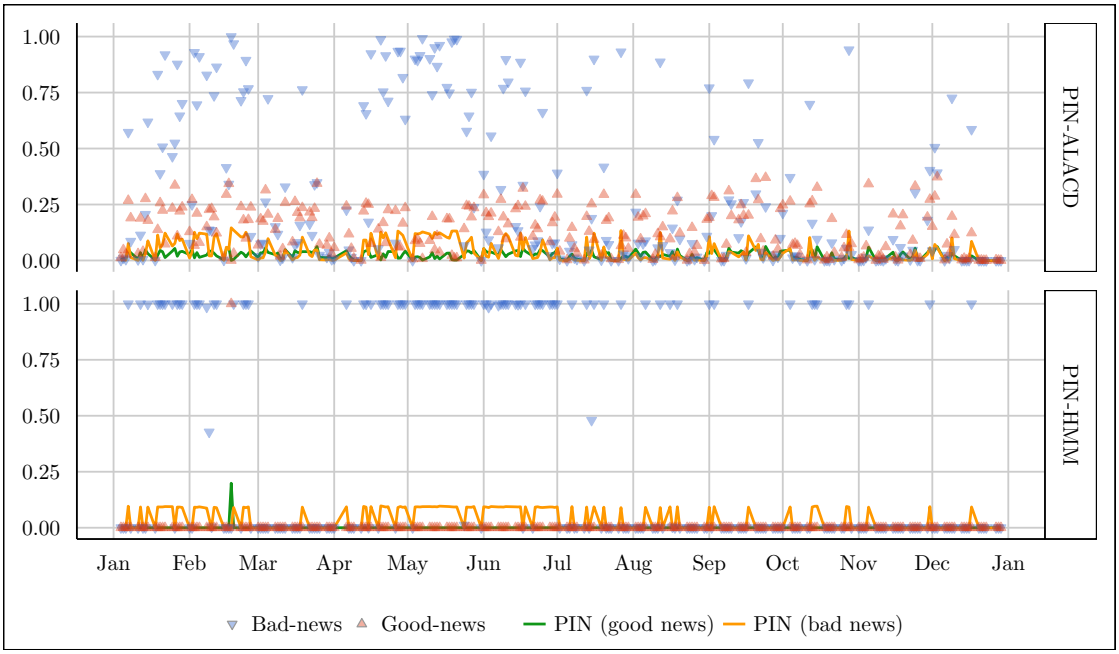

Figure 11.34: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for DAI in 2010.

Figure 11.35: Daily estimates of the probabilities of informed trading driven by good and bad news and good- and bad-news state probabilities for VOW in 2010.

The estimates of the probability of informed trading for the dynamic models are not directly comparable with the ones for the static EHO model. As explained earlier, the latter models the PIN variable as the a priori risk the market maker has to face at the beginning of each trading day. In the approach by Tay, Ting, Tse, and Warachka (2009) and our PIN-HMM model, the probability of informed trading displays the relation of the expected total number of transactions initiated by informed traders to the total number of trades.

While a comparison of PIN estimates from static and dynamic models is not reasonable, the expression $\frac{\mu}{\epsilon_b + \epsilon_s + \mu}$ from the EHO model is similar to the probability of informed trading in the dynamic setting. It incorporates the intensity of insiders on information events (μ) in its nominator and the total trading intensity ($\epsilon_b + \epsilon_s + \mu$) in its denominator (see figure 3.3).

We calculate averages of the probability of informed trading PIN on information events on a yearly basis for our PIN-HMM model. We label trading days as information events if the smoothed probability for a no-news condition is less than 1%. The results can be found in table 11.35. To achieve a better comparability in terms of the underlying time range for estimation results of the static and dynamic model, we sum up the yearly series of quarterly estimates from the EHO model and then calculate PIN. The fraction of information-based trading to total trading activities for each symbol and year are shown in table 11.36. Likewise to the comparisons of estimation results for static and dynamic models in section 11.1 and 11.2, we exclude the approach by Tay, Ting, Tse, and Warachka (2009).

It is obvious from tables 11.35 and 11.36 that the activity of insiders reported by the EHO model is substantially higher than their counterparts in the PIN-HMM approach. The results for the static model in table 11.36 range from about 18% to about 47%, with the majority of entries displaying values between 25% and 35%. The maximum yearly average of estimates of the probability of informed trading in table 11.35 equals 19.3% for HMC in 2010, while DAI exhibits the minimum of 6.7% in 2009. The difference between values in table 11.36 and 11.35 are larger than 0.1 for every symbol and year, irrespective of the marketplace under consideration.

In section 11.1 and 11.2 we explain that the static EHO model is overestimating the fraction of informed buys on good-news days and informed sells on bad-news days. As a consequence, the activity of market attendees which possess private information is also overestimated if we set our PIN-HMM model as reference.

In addition to the comparative analysis of the probability of informed trading in the EHO and PIN-HMM model, we exemplarily compare estimates of the posterior probabilities returned by both models. Since we do not want to overload this section with additional plots for each equity and year, we concentrate on previously highlighted trading periods. We investigate the first half of 2009 for GM and the year of 2008 for VOW. The corresponding graphics for the NYSE-listed symbol of GM can be found in figure 11.36 and for the German equity of VOW in figure 11.37, respectively.

A strong (pseudo)-sureness about the condition of trading days is displayed for our dynamic approach and the EHO model. For many trading days, on which both models see insider entering the market, they agree about the direction of private news for GM. There are information

events in every month with an increased amount from late February to late March. In May, the PIN-HMM model sees more trading days on which insiders are active. The estimates of the posterior probabilities for the EHO model tell us that there are only bad-news days at the end of the month. Our dynamic approach is in line with those results but returns high probabilities for information events triggered by positive direction of private news for early May.

Similar to figure 11.36 for GM it is obvious from figure 11.37 that the static EHO model and our PIN-HMM model are sure about the conditions of trading days in 2008 for the VOW symbol. However, the approach by Easley, Hvidkjaer, and O'Hara (2002) labels more trading days as information events. From April to August, the model exhibit high probabilities of information events, either with positive or negative private direction. Our dynamic approach displays only a few good-news trading days in the middle of July. Both models identify a clustering of trading days with active insiders from September to early November. While there are solely bad-news trading days in the EHO panel for this time range, we see a mixture of good-news and bad-news conditions in the panel for the PIN-HMM model.

	2007	2008	2009	2010
F	0.11596	0.13427	0.12583	0.14914
GM	0.10991	0.12290	0.12298	
HMC	0.14852	0.14639	0.18145	0.19377
JCI	0.11477	0.09730	0.11476	0.14973
TM	0.14208	0.15628	0.15267	0.16848
BMW	0.09513	0.09873	0.08600	0.08058
CON	0.10591	0.11170	0.14712	0.10437
DAI	0.08372	0.09465	0.06751	0.09407
VOW	0.10885	0.12712	0.11953	0.16802

Table 11.35: Average proportion informed transaction occupy of the total number of trades on information events in the context of the dynamic PIN-HMM model (average of the PIN_d variable on information events). Trading days are labeled as information events if the smoothed probability for a no-news condition is less than 1%.

	2007	2008	2009	2010
F	0.38066	0.45057	0.33106	0.47893
GM	0.24125	0.25642	0.44599	
HMC	0.27368	0.32122	0.38109	0.42273
JCI	0.23454	0.22500	0.26004	0.26863
TM	0.30435	0.29695	0.38374	0.47765
BMW	0.25070	0.25017	0.25876	0.21526
CON	0.23336	0.29124	0.37110	0.29784
DAI	0.26132	0.27956	0.18849	0.20760
VOW	0.28709	0.45499	0.33346	0.44416

Table 11.36: Proportion informed transaction occupy of the total number of trades on information events in the context of the static EHO model. Yearly series of quarterly estimates of ϵ_b, ϵ_s and μ are summed up to achieve a better comparability with the estimates in table 11.35.

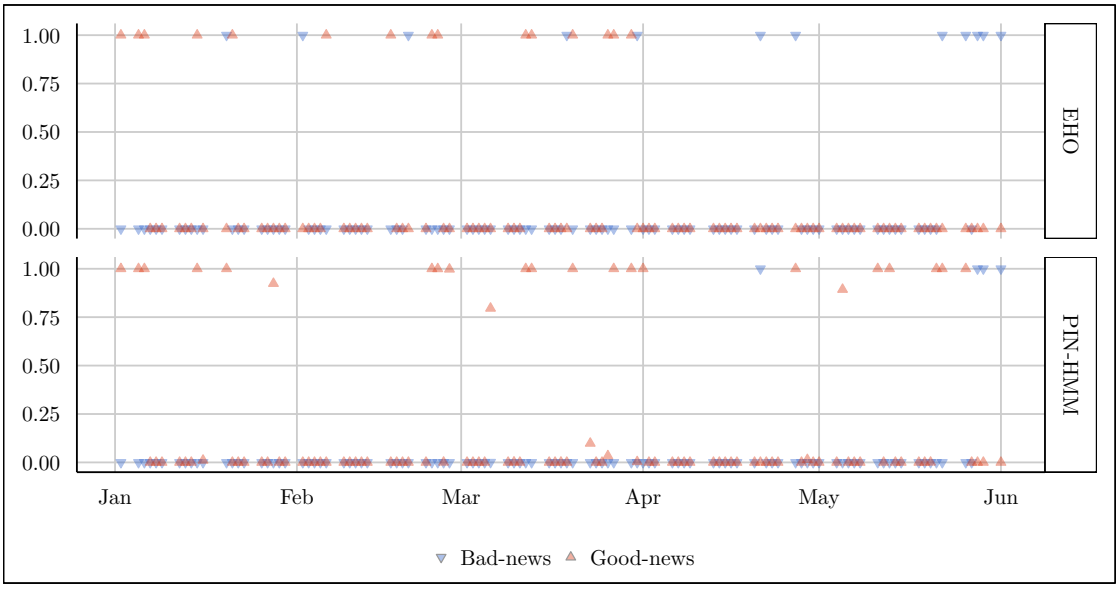

Figure 11.36: Comparison of posterior probabilities of conditions of trading days returned by the static EHO model with state probabilities reported by the dynamic PIN-HMM model for GM in the first half of 2009. The probabilities displayed in the second row of the plot can also be found in figure 11.20.

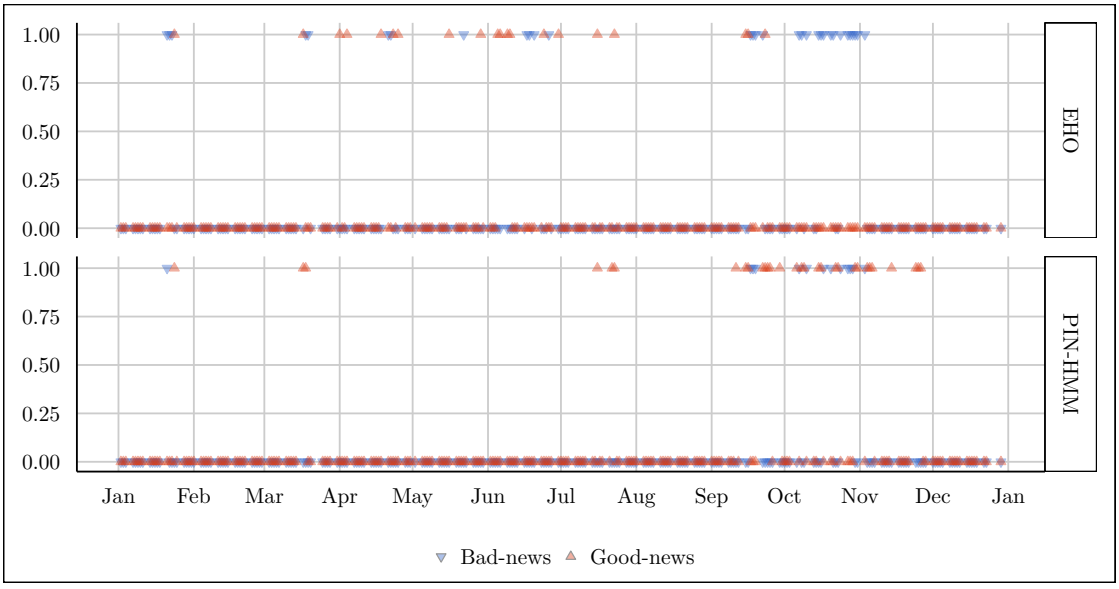

Figure 11.37: Comparison of posterior probabilities of conditions of trading days returned by the static EHO model with state probabilities reported by the dynamic PIN-HMM model for VOW in 2008. The probabilities displayed in the second row of the plot can also be found in figure 11.18.

11.4 Goodness of Fit

We can gain insights into the goodness of fit for both dynamic approaches by inspecting how well they capture the dynamics of the observed durations. Therefore we utilize probability integral transforms $z_i^q = F_{x_{i,d}}^q\left(x_{i,d} \mid \mathscr{F}_{i-1,d}\right)$ of observed durations. Under the assumption that the true distributions of $x_{i,d}$ are incorporated in the dynamic approaches, the probability integral transforms follow a standard uniform distribution (see Rosenblatt 1952). This result was extended by Diebold, Gunther, and Tay (1998) who have proven that the sequence of z_i's is *iid* standard uniform distributed if the true distribution is known.

According to equation (6.4) the observed duration for i-th transaction on trading day d is the minimum of waiting times of buys' and sells' latent point processes. Therefore we can write the cdf of the i-th duration on trading day d which resides in state q as

$$F_{x_{i,d}}^q(t) = \Pr\Big(\min\Big(T_{1,i,d}^q, T_{-1,i,d}^q\Big) \leq t \;\Big|\; \mathscr{F}_{i-1,d}\Big) \tag{11.2}$$

This implies that at least one of $T_{1,i,d}^q$ and $T_{-1,i,d}^q$ is weakly smaller than t.

$$F_{x_{i,d}}^q(t) = \Pr\Big(T_{1,i,d}^q \leq t \vee T_{-1,i,d}^q \leq t \;\Big|\; \mathscr{F}_{i-1,d}\Big) \tag{11.3}$$

The probability in equation (11.3) can be rewritten as one minus the probability that both, $T_{1,i,d}^q$ and $T_{-1,i,d}^q$, are larger than t.

$$F_{x_{i,d}}^q(t) = 1 - \Pr\Big(T_{1,i,d}^q > t \wedge T_{-1,i,d}^q > t \;\Big|\; \mathscr{F}_{i-1,d}\Big) \tag{11.4}$$

Since waiting times of buys' and sells' processes are independent given we know about the history $\mathscr{F}_{i-1,d}$, the cdf for $x_{i,d}$ on trading day d which resides in state q for PIN-ALACD model is given by

$$F_{x_{i,d}}^q\left(t \mid \mathscr{F}_{i-1,d}\right) = 1 - S_{T_{1,i,d}^q}\left(t \mid \mathscr{F}_{i-1,d}\right) S_{T_{-1,i,d}^q}\left(t \mid \mathscr{F}_{i-1,d}\right), \tag{11.5}$$

where $S_{T_{1,i,d}^q}$ and $S_{T_{-1,i,d}^q}$ represent survivor functions belonging to the interarrival times of buys' and sells' point processes, respectively. The product of both survivor functions can be interpreted as probability that neither the point process of buys nor the one for sells has arrived yet and that they do not signal the occurrence of a transaction.

Durations are only observable during official opening hours which tends to result in lower amount of large durations, especially at the end of a trading day. Therefore, we have censored datasets for which each transaction fulfills the constraint that its durations is (weakly) lower than the remaining trading time. The last duration on each trading day begins before official

closing and ends afterwards. Hence, it is not observable. Taken this censoring into account in our PIN-HMM approach, we need to incorporate the remaining trading time in equation (11.5),

$$\tilde{F}^{\,q}_{x_{i,d}}\big(t \mid \mathscr{F}_{i-1,d}\big) = \frac{1 - S_{T^q_{1,i,d}}\big(t \mid \mathscr{F}_{i-1,d}\big) S_{T^q_{-1,i,d}}\big(t \mid \mathscr{F}_{i-1,d}\big)}{1 - S_{T^q_{1,i,d}}\big(\mathrm{RT}_{i,d} \mid \mathscr{F}_{i-1,d}\big) S_{T^q_{-1,i,d}}\big(\mathrm{RT}_{i,d} \mid \mathscr{F}_{i-1,d}\big)}, \tag{11.6}$$

where $\mathrm{RT}_{i,d}$ is the upper bound for i-th transaction on trading day d (see equation (6.22) in chapter 6).

The probability integral transform for the i-th duration on trading day d is then given as

$$\begin{aligned}\tilde{F}_{x_{i,d}}\big(t \mid \mathscr{F}_{i-1,d}\big) = {} & \Pr\big(q_d = \mathscr{N} \mid \mathscr{F}_{i-1,d}\big) \cdot \tilde{F}^{\,\mathscr{N}}_{x_{i,d}}\big(t \mid \mathscr{F}_{i-1,d}\big) \\ & + \Pr\big(q_d = \mathscr{G} \mid \mathscr{F}_{i-1,d}\big) \cdot \tilde{F}^{\,\mathscr{G}}_{x_{i,d}}\big(t \mid \mathscr{F}_{i-1,d}\big) \\ & + \Pr\big(q_d = \mathscr{B} \mid \mathscr{F}_{i-1,d}\big) \cdot \tilde{F}^{\,\mathscr{B}}_{x_{i,d}}\big(t \mid \mathscr{F}_{i-1,d}\big),\end{aligned} \tag{11.7}$$

with the condition of trading day d, q_d. However, equation (11.7) is not operational, since we cannot observe the evolution of the probabilities $\Pr\big(q_d = \mathscr{N} \mid \mathscr{F}_{i-1,d}\big)$, $\Pr\big(q_d = \mathscr{G} \mid \mathscr{F}_{i-1,d}\big)$ and $\Pr\big(q_d = \mathscr{B} \mid \mathscr{F}_{i-1,d}\big)$ over a trading day d.

Therefore, we simplify the analysis by further conditioning the cdf of the durations in equation (11.7) on the state of trading day d. We then incorporate estimates of daily smoothed probabilities from section 6.2.2 for the PIN-HMM model and state probabilities of trading days introduced in section 6.1 for the PIN-ALACD approach.

Furthermore, one could think of transforming the estimates of daily state probabilities in a way that they only take on values in $\{0, 1\}$. For each trading day the highest estimated state probability is identified and set to unity, while the two remaining probabilities are set to zero. Since the HMM approach exhibits are very high (pseudo)-sureness about conditions for almost every trading day, there would only be very little difference to the results presented in this section. However, the PIN-ALACD model is very often unsure about the states of trading days and this transformation would influence the results to a huge degree. Therefore, we decided against this additional step to keep the results representative also for the model by Tay, Ting, Tse, and Warachka (2009).

Both dynamic models have problems with adequately capturing the dynamics of (very) short durations.[128] This finding was already mentioned in the works by Tay, Ting, Tse, and Warachka (2009) and Bauwens, Giot, Grammig, and Veredas (2004). Estimates of shape parameters of Weibull distributions for the PIN-HMM model are less than one for all equities and years. Densities therefore tend to infinity as durations tend to zero. Hence, there are not enough very

[128] Similar to section 11.3, we place all figures at the end of the section.

small durations compared with the amount predicted by the Weibull distributions. Nevertheless, the fit of PIN-HMM is much better than for PIN-ALACD. In addition, we inspected autocorrelations and find them to be reasonable small. However, the majority of values lie outside the confidence bands due to the tremendous sample size for all equities in our datasets.[129]

All kernel densities for probability integral transforms of PIN-ALACD have in common that they are more or less bathtub-shaped. Especially the PIN-HMM plots for the German stocks mimic the behavior of the density of a standard uniform distribution very well, extremely short durations excluded.

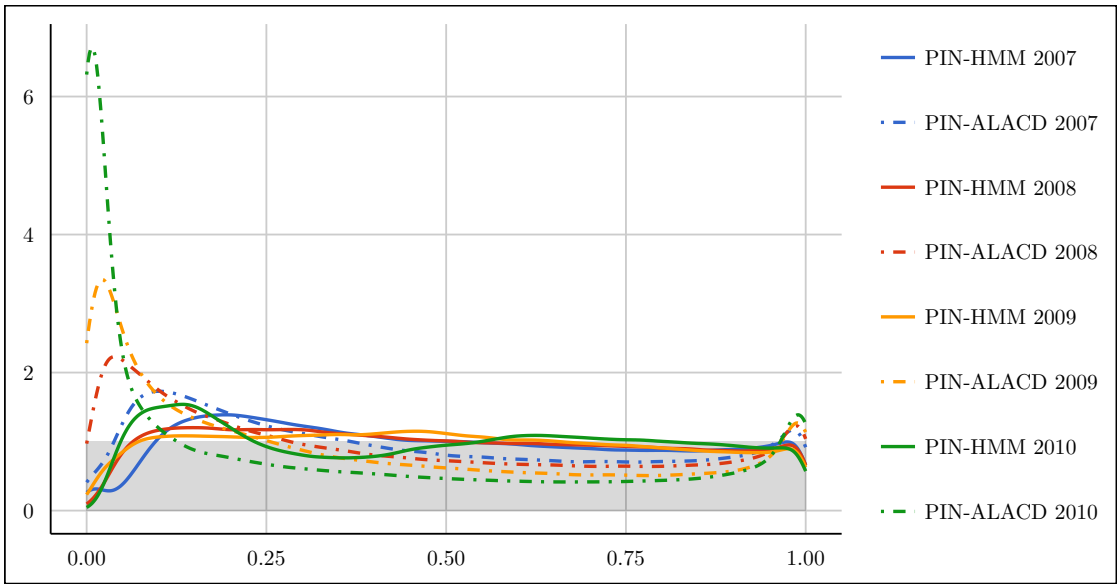

Figure 11.38: Kernel density estimations of the probability integral transforms of the transaction durations for F. Area under the standard uniform pdf is coloured in grey.

[129] To save space, we do not include results for the autocorrelation functions in this work. However, they can be found at `https://anre.shinyapps.io/AdvancedPIN/`.

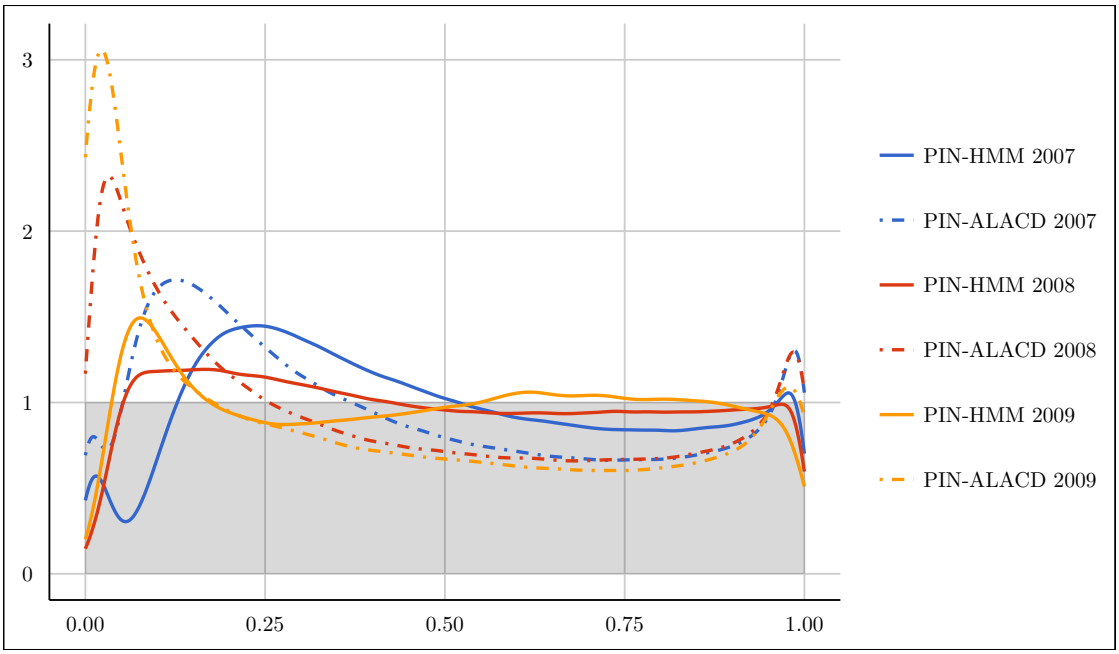

Figure 11.39: Kernel density estimations of the probability integral transforms of the transaction durations for GM. Area under the standard uniform pdf is coloured in grey.

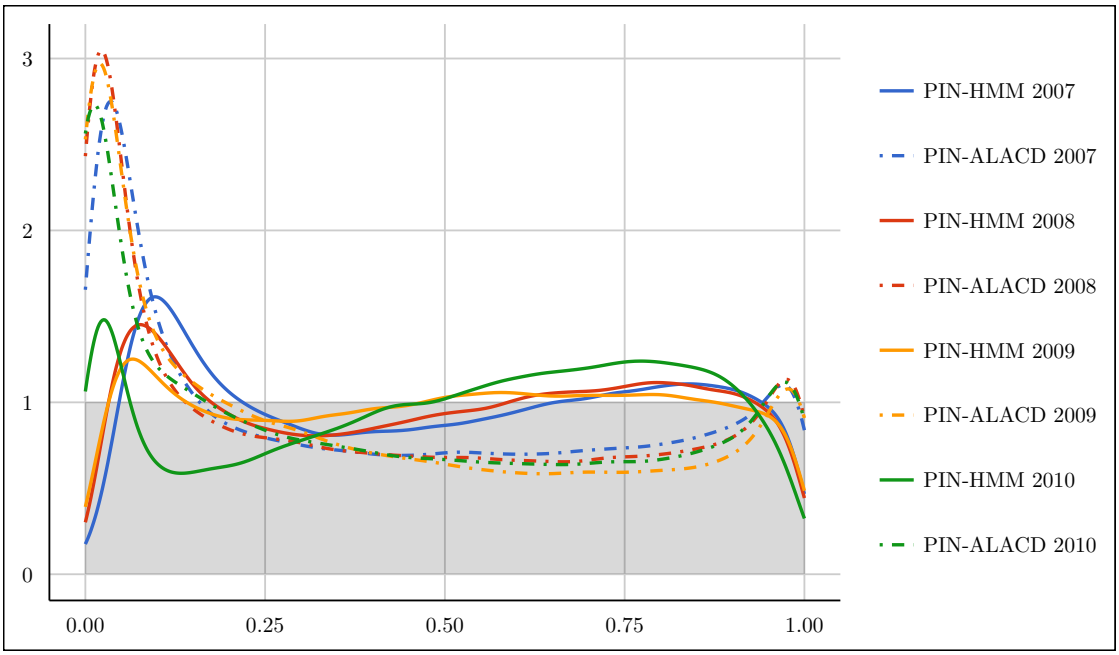

Figure 11.40: Kernel density estimations of the probability integral transforms of the transaction durations for HMC. Area under the standard uniform pdf is coloured in grey.

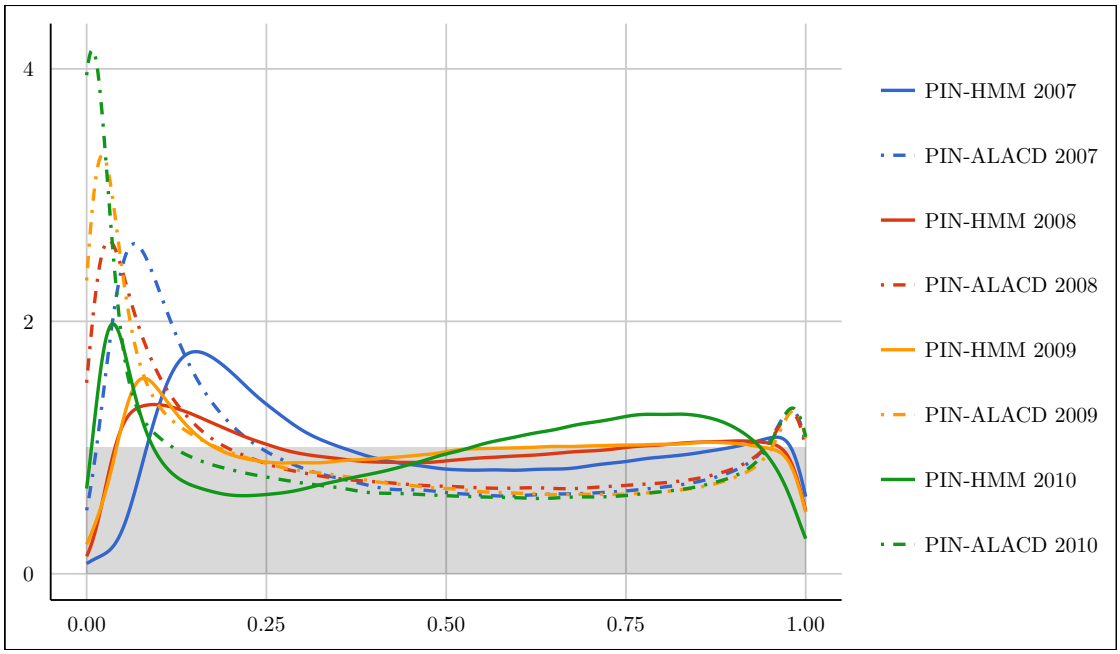

Figure 11.41: Kernel density estimations of the probability integral transforms of the transaction durations for JCI. Area under the standard uniform pdf is coloured in grey.

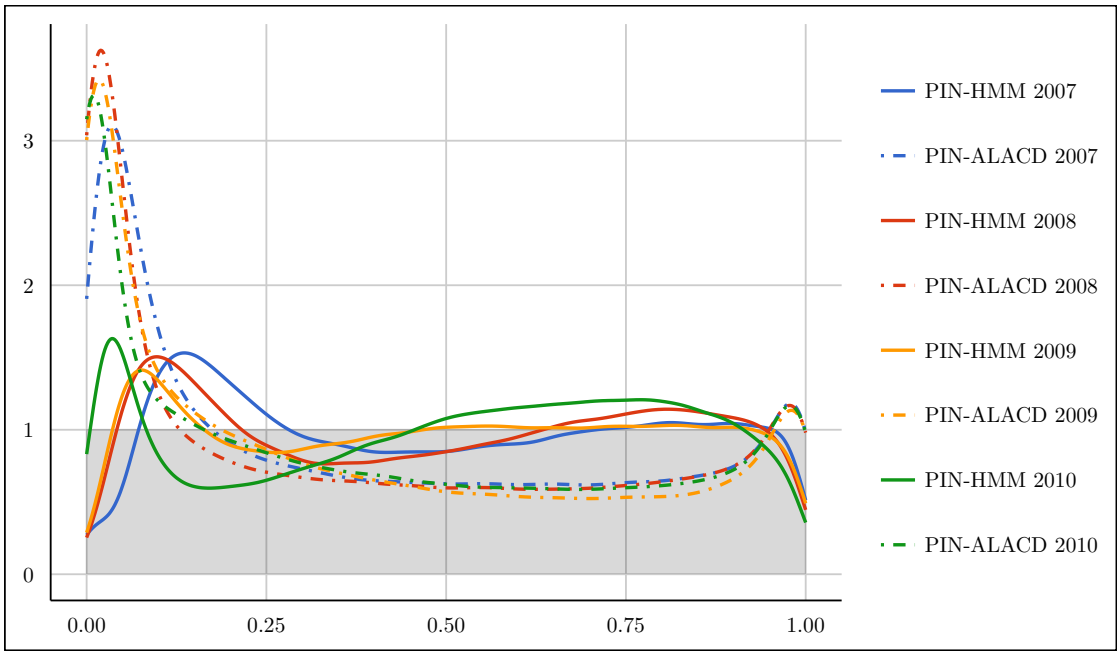

Figure 11.42: Kernel density estimations of the probability integral transforms of the transaction durations for TM. Area under the standard uniform pdf is coloured in grey.

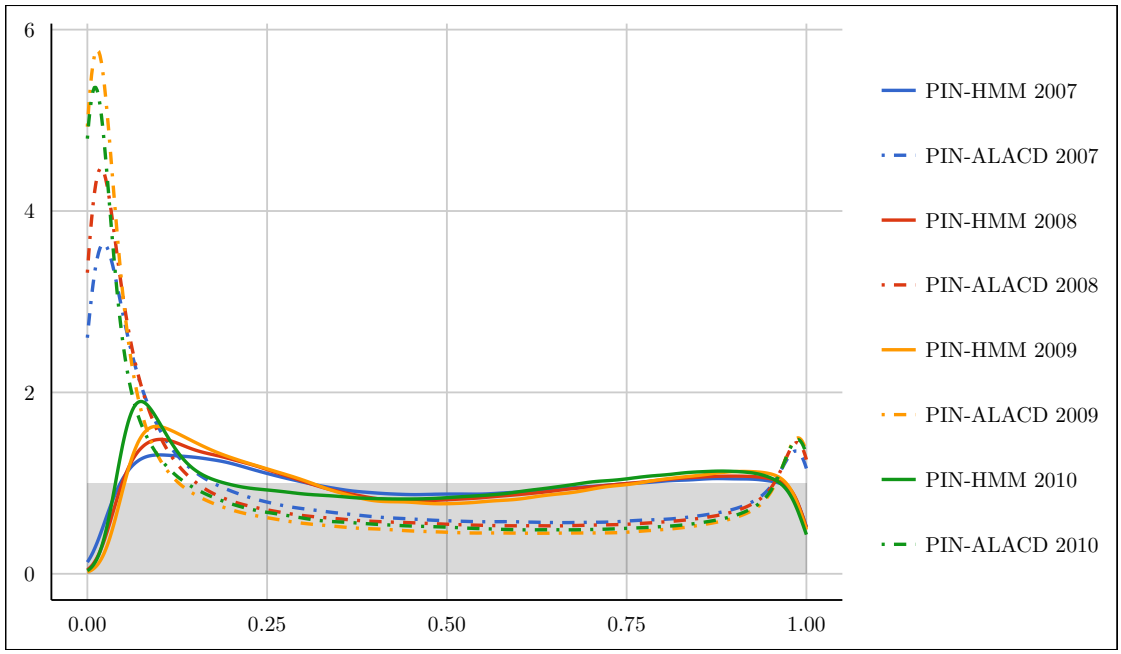

Figure 11.43: Kernel density estimations of the probability integral transforms of the transaction durations for BMW. Area under the standard uniform pdf is coloured in grey.

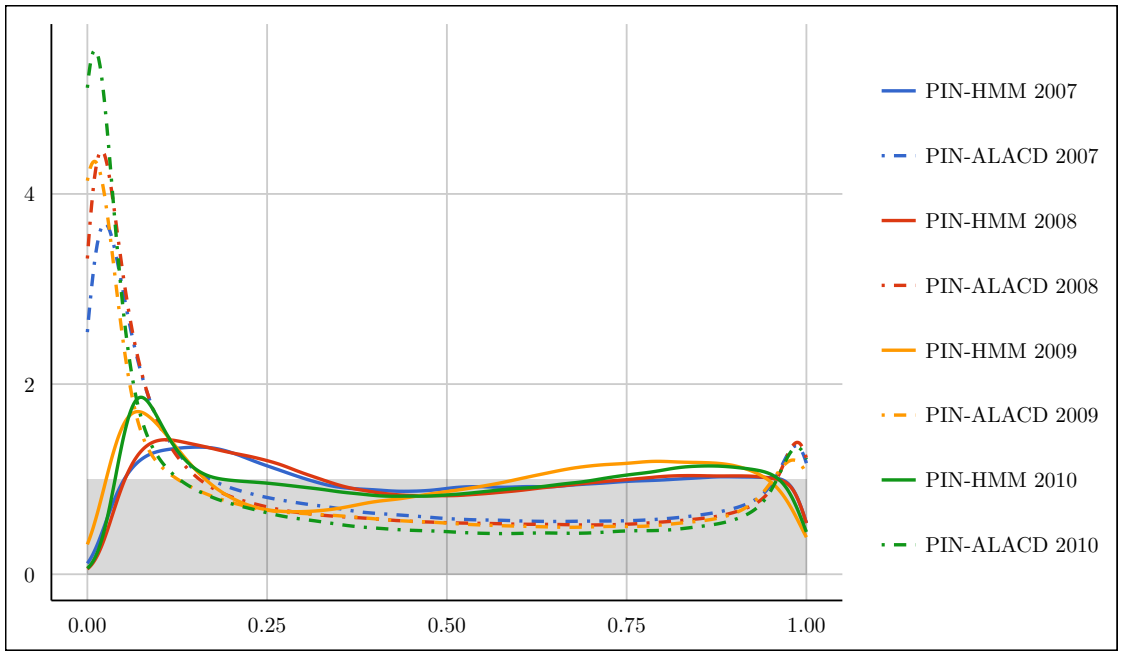

Figure 11.44: Kernel density estimations of the probability integral transforms of the transaction durations for CON. Area under the standard uniform pdf is coloured in grey.

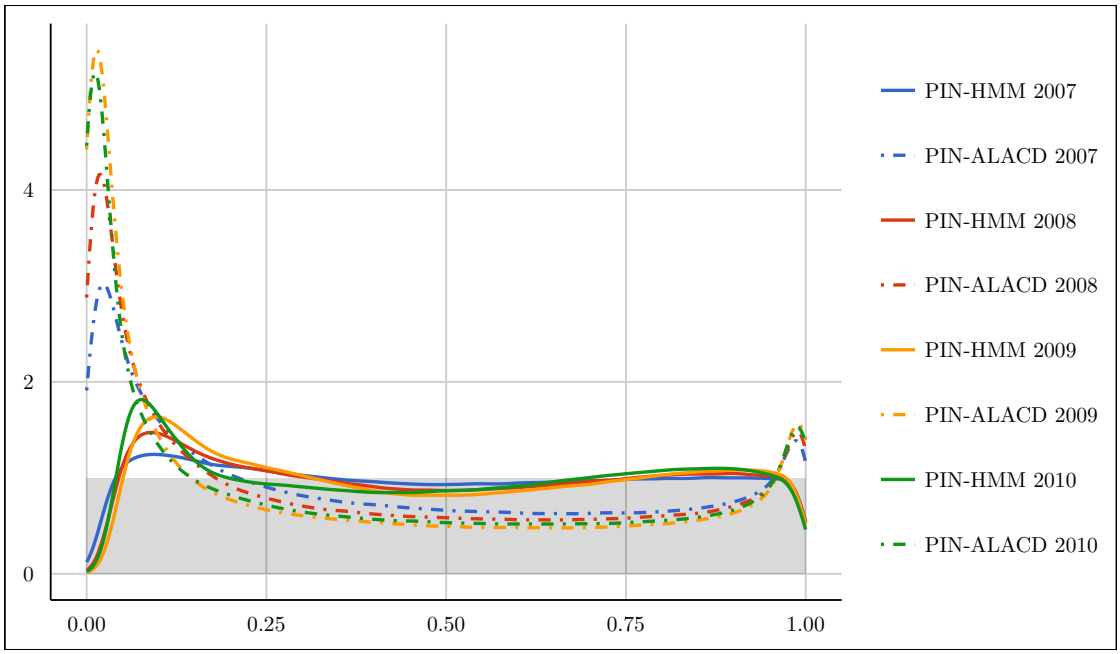

Figure 11.45: Kernel density estimations of the probability integral transforms of the transaction durations for DAI. Area under the standard uniform pdf is coloured in grey.

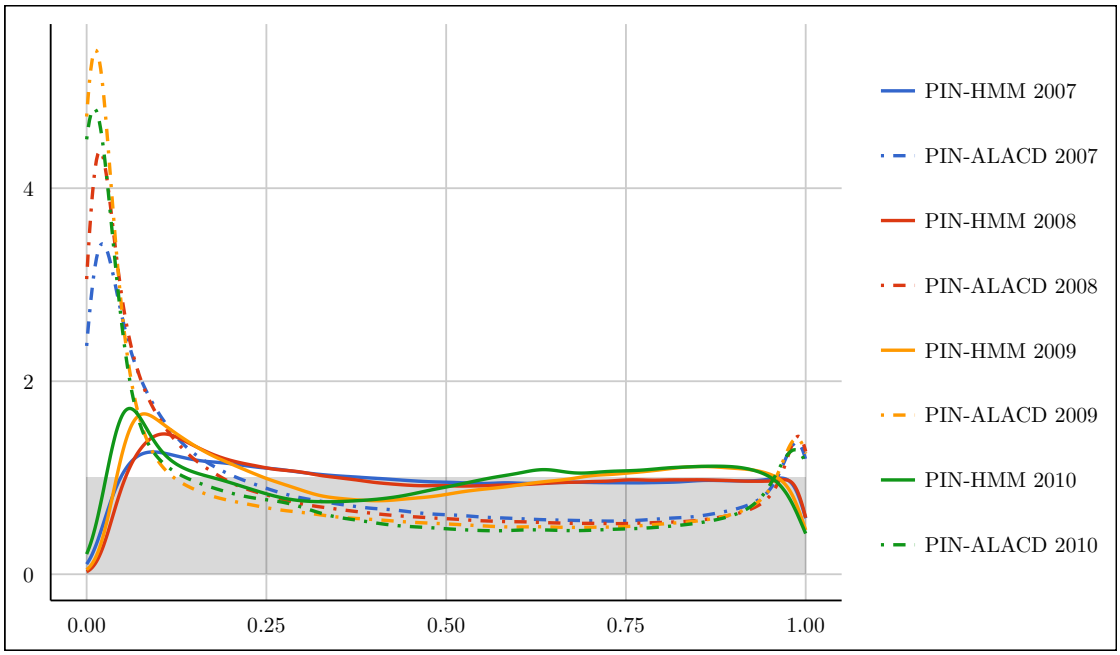

Figure 11.46: Kernel density estimations of the probability integral transforms of the transaction durations for VOW. Area under the standard uniform pdf is coloured in grey.

11.5 Intraday Estimation Results

Utilizing intraday time intervals in the context of the dynamic models for the probability of informed trading offer an alternative for the typically used unit time interval of one trading day. We apply this alternative setting to short trading periods of two of the equities under consideration in this work. This section can be seen as a *proof of concept* for intraday analyses of PIN, where in-depth analyses of this specific topic are open for further research.

We exemplarily investigate the behavior of the PIN measures for the last two trading days before the delisting of GM and trading days around the dramatic price increase for VOW on Xetra. The special circumstances for both equities on these trading days were already discussed in sections 11.1 and 11.2. In addition, we included the intraday prices for GM and VOW in figures 11.47 and 11.48, respectively.

In this section we provide intraday estimates of the probabilities of informed trading driven by good and bad news as well as estimates of good- and bad-news state probabilities. Trading periods of five minutes length are chosen as unit time interval.

As mentioned in section 6.3, the main focus in this thesis lies on the models for estimating the probability of informed trading. To the best of our knowledge, the behavior of the PIN variable was not analyzed around certain important market events (e.g., M&A's or earning announcements) on an intraday level yet. This can be a very interesting aspect for future research but goes far beyond the scope of this work.

The approach by Tay, Ting, Tse, and Warachka (2009) is included for the sake of completeness, but, similar to the estimations on a daily basis, fails to return reasonable results. As we can see in table 11.37[130] there are boundary solutions for estimates of δ_3 and δ_4 for both equities in the PIN-ALACD model. Hence, the model is not able to distinguish between good- and bad-news conditions for trading periods at all. Therefore, we concentrate on the results of our hidden Markov approach and exclude any plot for the former.[131]

Table 11.38 displays that the assumptions about the intercepts of the ALACD recursions mentioned in sections 11.1 and 11.2 are valid for buys' processes of both equities and the process of sells for GM. However, intercepts in the sells' ALACD recursion for VOW are very similar. Assumptions of coefficients belonging to signed volume are fulfilled for VOW but signs are switched in the GM case. Again, sums of $\hat{\alpha}_1$ and $\hat{\beta}_1$ as wells as $\hat{\alpha}_{-1}$ and $\hat{\beta}_{-1}$ are not close to one. Estimates of shape parameters of Weibull distributions are all smaller than unity, leading to long durations being less probable. Hence, the goodness of fit plots display similar behavior for (very) short durations as already explained in section 11.4.

On May 29, 2009 there are only a small number of information events for GM. Informed traders enter the market in the morning hours to act as sellers, while they enter the market just before the market closing to buy shares. Alternating occurrences of good- and bad-news periods can

[130] Tables of estimation results, plots of state probabilities and probabilities of informed trading as well as the probability integral transform graph are placed at the end of the section.

[131] Visualizations of the results can be found at `https://anre.shinyapps.io/AdvancedPIN/`.

be observed for the last trading day before the delisting of GM. PIN-HMM model clearly identifies the direction of private news for the majority of intervals. We see clustering of information events from 10 am to 11 am in figure 11.50 and again an increased activity of informed buyers in the afternoon.

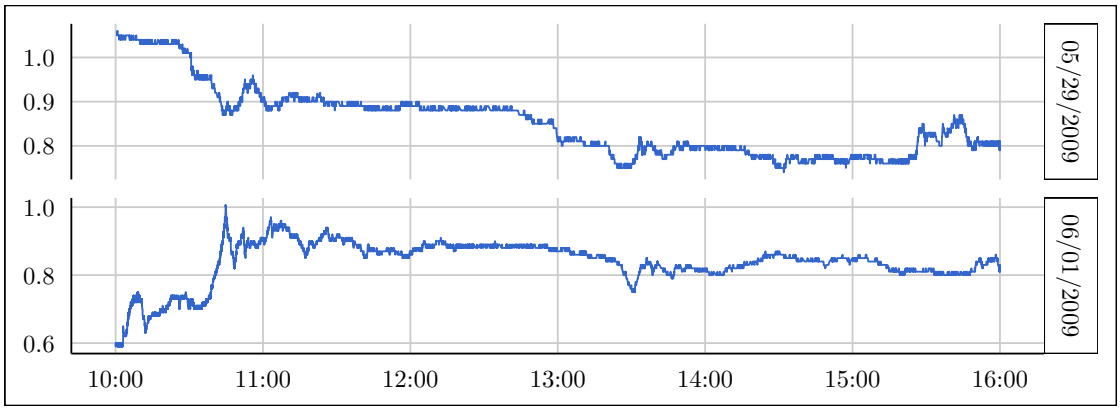

Figure 11.47: Intraday prices (in US $) of GM on May 29, 2009 and June 1, 2009. We do not display the first 30 minutes of each trading day, since they are not included in our datasets we utilize for estimation purposes (see chapter 8).

Likewise to the results for GM, the PIN-HMM model offers high (pseudo-)sureness about the condition of most trading periods for VOW. In figure 11.51, a clustering of information events can be found on each trading day under consideration. We see informed market participants buying shares for several trading periods around 5 pm on October 27, 2008 and act as sellers afterwards shortly before the official closing of the marketplace. The trading hours until lunch time are dominated by information events with negative direction of private news on October 28, 2008, with a short period of insiders driven by good news enter the market before 11 am. On October 29, 2008, we see high probabilities for bad-news conditions from 9:30 am to 10 am and high probabilities for periods with informed traders triggered by positive private news from 1 pm to 2 pm. In comparison with the results for GM, the PIN-HMM identifies substantially more information events for VOW.

On the one hand, we see from our analysis of the probability of informed trading on a intraday level that there is often a clustering of information events of the same direction. On the other hand, figures 11.50 and 11.51 display many hours of relevant trading days for which no insider is reported to be active by our dynamic model. This might be due to the fact that there were no price-relevant private information in the run-up of the trading days under consideration.

The field of intraday analyses is very new in the context of the probability of informed trading. At the time of writing, we are the first to use very short intervals of five minutes as unit time intervals. To give the PIN measure a chance to correctly capture market characteristics and the behavior of traders at the beginning of a trading day, it is necessary not to exclude any transaction data. Moreover, over-night effects of the specific marketplaces should be incorporated. This would yield completely different datasets compared to those we used for our

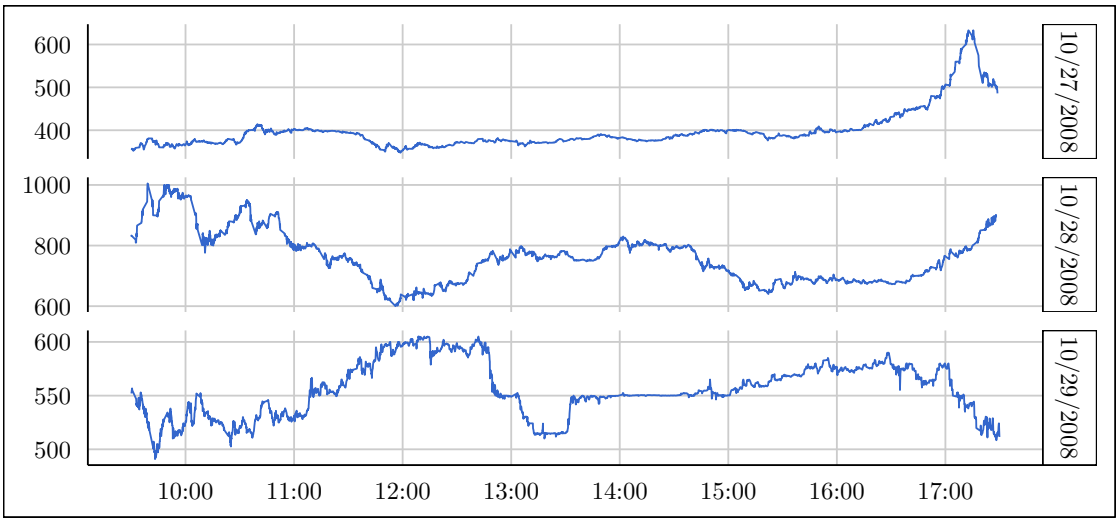

Figure 11.48: Intraday prices (in EUR) of VOW from October 27, 2008 to October 29, 2008. We do not display the first 30 minutes of each trading day, since they are not included in our datasets we utilize for estimation purposes (see chapter 8).

analyses. These steps of data preprocessing are out of scope for this work but are very interesting and important points for future research to further improve the quality of the probability of informed trading.

Nevertheless, intraday analyses seem to work well for our dynamic approach which models state probabilities by hidden Markov chains, aside from the potential problems in the very beginning of trading days mentioned above. The densities of probability integral transforms shown in figure 11.49 also struggle with very short durations as explained in section 11.4 but look overall reasonable. However, we only applied intraday estimation on two special time periods in this work. It needs further applications and testing to verify the good overall performance of the PIN-HMM model in terms of estimating the probabilities of informed trading and trading days' conditions for (very) short time ranges.

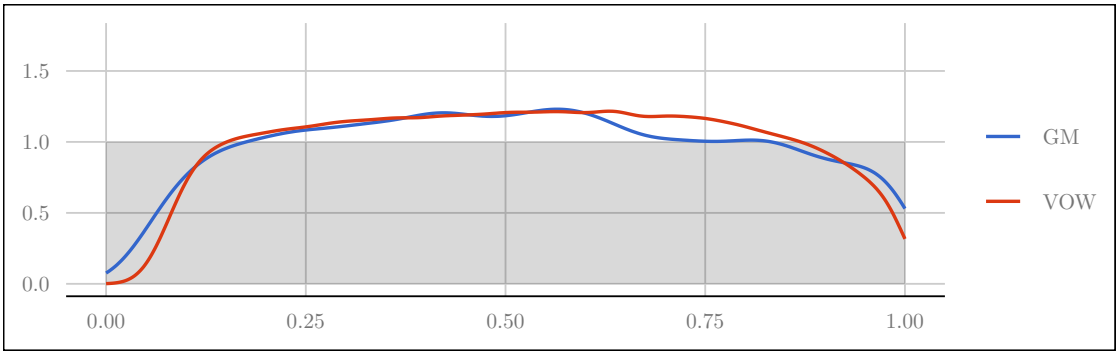

Figure 11.49: Kernel density estimations of the probability integral transforms of intraday estimation results for GM and VOW. Area under the standard uniform pdf is coloured in grey.

	Param.	GM	VOW
State Probabilities			
Intercept	δ_1	-2.01980	-1.85450
		(0.37395)	(0.24254)
Cmp. total volumes	δ_2	3.31791	2.45829
		(0.72325)	(0.38994)
Cmp. sell volumes	δ_3	**0**	**0**
		— — —	— — —
Cmp. buy volumes	δ_4	**0**	**0**
		— — —	— — —
ALACD Specifications			
Intercept (Buy after Buy)	$\nu_{1,1}$	0.97871	0.01942
		(0.04154)	(0.00796)
Intercept (Buy after Sell)	$\nu_{1,-1}$	0.10205	0.12813
		(0.03409)	(0.00764)
Cond. Duration (Buys)	α_1	0.63232	0.73766
		(0.00907)	(0.00392)
Lagged Duration (Buys)	β_1	0.23170	0.12540
		(0.00529)	(0.00155)
Signed Volume (Buys)	ζ_1	-0.02702	0.03906
		(0.00504)	(0.00164)
Intercept (Sell after Buy)	$\nu_{-1,1}$	0.04356	0.07926
		(0.04460)	(0.00903)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	1.33753	0.13705
		(0.05812)	(0.00868)
Cond. Duration (Sells)	α_{-1}	0.57200	0.72278
		(0.01369)	(0.00453)
Lagged Duration (Sells)	β_{-1}	0.19608	0.12959
		(0.00677)	(0.00189)
Signed Volume (Sells)	ζ_{-1}	0.04599	-0.02535
		(0.00630)	(0.00183)
Adj. informed trading	τ	0.75017	0.37139
		(0.02770)	(0.00932)

Table 11.37: Intraday estimation results for the PIN-ALACD model. Unit time interval is set to five minutes. Dates included in the datasource are May 29, 2009 and June 1, 2009 for GM and trading days from October 27, 2008 to October 29, 2008 for VOW. Figures in parentheses denote standard errors.

	Param.	GM	VOW
Transistion Probabilities			
$\mathscr{B} \longrightarrow \mathscr{G}$	$a_{\mathscr{B}\mathscr{G}}$	0	0.11460
		− − −	(0.05505)
$\mathscr{B} \longrightarrow \mathscr{N}$	$a_{\mathscr{B}\mathscr{N}}$	0.81119	0.10587
		(0.17304)	(0.05276)
$\mathscr{G} \longrightarrow \mathscr{B}$	$a_{\mathscr{G}\mathscr{B}}$	0.09281	0.07967
		(0.08912)	(0.05800)
$\mathscr{G} \longrightarrow \mathscr{N}$	$a_{\mathscr{G}\mathscr{N}}$	0.22483	0.33999
		(0.12968)	(0.10383)
$\mathscr{N} \longrightarrow \mathscr{B}$	$a_{\mathscr{N}\mathscr{B}}$	0.02487	0.03224
		(0.01557)	(0.01376)
$\mathscr{N} \longrightarrow \mathscr{G}$	$a_{\mathscr{N}\mathscr{G}}$	0.03886	0.04444
		(0.01815)	(0.01681)
ALACD Specifications			
Intercept (Buy after Buy)	$\nu_{1,1}$	0.33051	0.58305
		(0.06812)	(0.01413)
Intercept (Buy after Sell)	$\nu_{1,-1}$	1.86792	0.67061
		(0.09088)	(0.01372)
Cond. Duration (Buys)	α_1	0.58167	0.76856
		(0.01773)	(0.00485)
Lagged Duration (Buys)	β_1	0.20762	0.14878
		(0.01026)	(0.00281)
Signed Volume (Buys)	ζ_1	0.03853	-0.03978
		(0.00957)	(0.00275)
Intercept (Sell after Buy)	$\nu_{-1,1}$	1.12660	0.44671
		(0.05598)	(0.01022)
Intercept (Sell after Sell)	$\nu_{-1,-1}$	0.29657	0.50000
		(0.04367)	(0.00972)
Cond. Duration (Sells)	α_{-1}	0.70685	0.82824
		(0.01128)	(0.00382)
Lagged Duration (Sells)	β_{-1}	0.20698	0.11139
		(0.00756)	(0.00214)
Signed Volume (Sells)	ζ_{-1}	-0.01676	0.03667
		(0.00633)	(0.00206)
Adjustment (good news)	τ_1	0.78128	0.27592
		(0.05925)	(0.02241)
Adjustment (bad news)	τ_{-1}	0.66886	0.40297
		(0.05874)	(0.01443)
Shape Parameters			
Buys	k_1	0.64184	0.60637
		(0.00606)	(0.00137)
Sells	k_{-1}	0.69136	0.63901
		(0.00538)	(0.00126)

Table 11.38: Intraday estimation results for the PIN-HMM model. Unit time interval is set to five minutes. Dates included in the datasource are May 29, 2009 and June 1, 2009 for GM and trading days from October 27, 2008 to October 29, 2008 for VOW. Figures in parentheses denote standard errors.

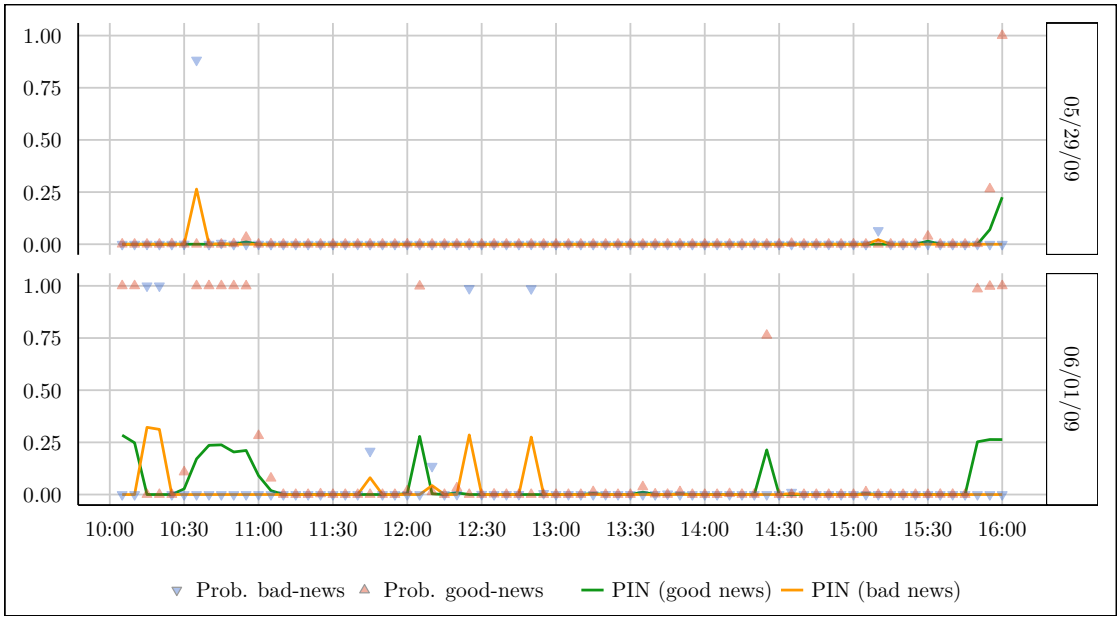

Figure 11.50: Intraday probabilities of informed trading (driven by good and bad news) and daily probabilities of good- and bad-news trading day conditions for GM at the last two trading days before its delisting. Length of intraday intervals is set to five minutes.

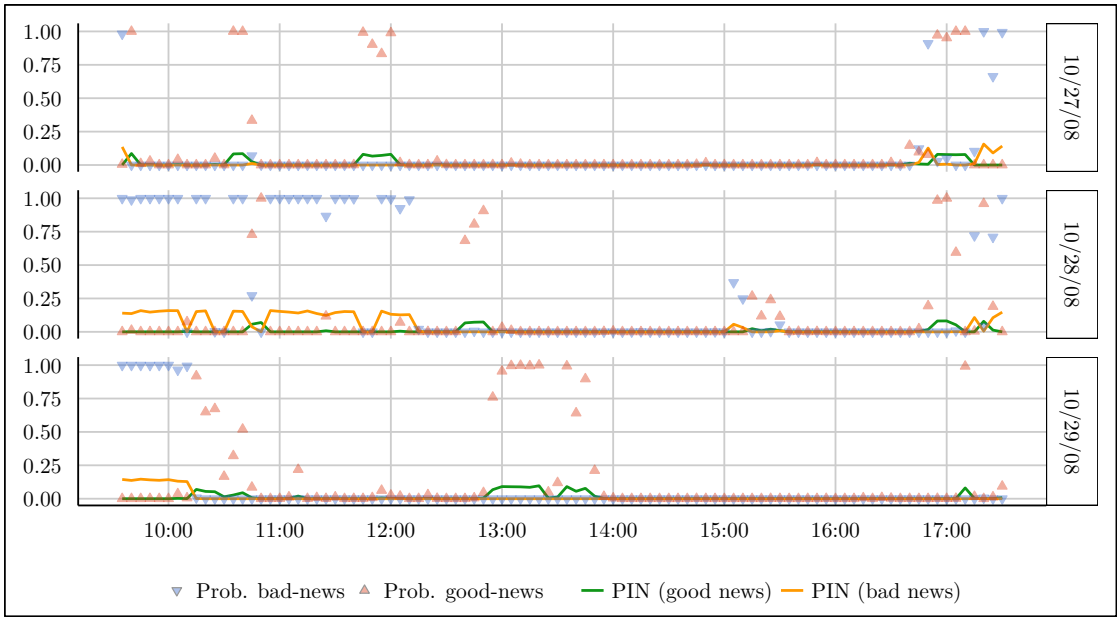

Figure 11.51: Intraday probabilities of informed trading (driven by good and bad news) and daily probabilities of good- and bad-news trading day conditions for VOW from October 27, 2008 to October 29, 2008. Length of intraday intervals is set to five minutes.

12 Conclusion

We cover the most prominent static as well as dynamic models for the probability of informed trading in this work. Chapters 3 and 4 deal with the theory of the well-known EKOP and EHO models and give in-depth explanations of various (numerical) aspects related to optimization routines in the setup of constant model parameters.

We introduce well-established statistical techniques to the context of static models for estimating the probability of informed trading. With posterior probabilities of trading days' conditions and confidence intervals for estimates of the PIN measure, we offer tools which can help to improve analyses of future empirical applications. Furthermore, posterior probabilities in the context of static PIN models shift the research focus in the direction of the dynamic approaches.

Posterior probabilities can clearly identify states of the majority of trading days for our datasets. They help to get more detailed information about activities of insiders than it is possible with only the knowledge of a priori probabilities of a good-news or bad-news information event. Only the proportion each trading day condition may occupy of the total number of trading days could be reported by past publications, while prospective works are able to display the distribution of states for each single trading day.

We also conduct a simulation study in chapter 4 to investigate the performance and quality of the initial value algorithms currently available in the literature related to static PIN models. We break with the simulation designs which were already presented in the literature of the probability of informed trading. Our procedure to create synthetic datasets of daily aggregated buys and sells is more flexible and fits better to the characteristics of the EHO model compared with the settings of simulations in the works by Gan, Chun, and Johnstone (2015) and Ersan and Alıcı (2016). While the former uses only 1000 simulation runs, the latter is more appropriate for the simpler EKOP model, which is a simplification of the EHO model. By utilizing a total number of 100,000 runs and a simulation procedure which takes into account (very) infrequently as well as heavily traded equities, we see that the HAC algorithm performs best. Although the overall results are similar for all tested algorithms, the approach utilizing hierarchical clustering is the best compromise in terms of runtime and goodness of the resulting maximum likelihood estimators of the model parameters.

We use the HAC algorithm for all optimizations in the chapter of empirical applications of the static EHO model to receive sets of initial values. As shown in section 4.2.2, the likelihood

factorization by Lin and Ke (2011) is very stable even for heavily traded equities and outperforms the alternative representation of the likelihood by Easley, Hvidkjaer, and O'Hara (2002). Hence, solely the former is included in our simulation study and the workflow of estimating the model parameters of the EHO model applied to our transaction data for NYSE- and Xetra-listed stocks.

Chapter 10 displays the empirical results of the static models as well as interpretations thereof. We apply the static EHO model to high-frequency transaction data, which covers the years from 2007 to 2010, of several firms from the automobile branch which are listed either on the New York Stock Exchange or the German electronic Xetra marketplace. Since all symbols belong to the automobile sector, potential dates of interest are the releases of scrappage programs in the United States and Germany. We briefly discuss the behavior of the probability of informed trading around those prominent events. Moreover, the PIN estimates are also analyzed around the two special trading periods of the delisting of GM on NYSE in June 2009 and the days when the price of VOW on Xetra literally exploded in October 2008.[132]

We released the **pinbasic** package for the statistical programming language R on *CRAN*. The intention of the package is to offer user-friendly functions for fast and stable estimations of the probability of informed trading.[133] Details about the functionalities and usage of functions including their complete source code can be found in chapter 5.

The very young field of dynamic models using high-frequency transaction data to estimate the probability of informed trading is introduced in chapter 6. Major advantages of these approaches are that time-varying model parameters are allowed and therefore PIN can vary over unit time intervals. Also order sizes are taken into account which is a major point of criticism of static models. The focus of research in these models is no longer the a priori probability that the market marker is confronted with market participants which possess private information, as it is in the static models. The probability of informed trading is interpreted as the unobservable fraction of insiders in the dynamic models.

We discuss the PIN-ALACD model by Tay, Ting, Tse, and Warachka (2009) and our new PIN-HMM approach utilizing hidden Markov chains for modeling conditions of trading days. Our model generalizes the PIN-ALACD method in several points. While the autoregressive specifications of conditional waiting times of buys and sells are similar, our distributional assumption for the observed durations are much more flexible. We assume the durations of buys and sells to follow independent Weibull distributions with different shape parameters, but more general distributions can be incorporated with reasonable effort. Moreover, our approach of modeling states of trading days relaxes the independence assumption and differs completely from the logistic regression in the PIN-ALACD model. In the context of the PIN-HMM approach so-called smoothed probabilities, which are conditional probabilities given the complete data, for each state and unit time interval are returned. Therefore those probabilities are closely related to

[132] We do not conduct an in-depth investigation of the estimates of the probability of informed trading around these special events. The focus of this work lies on the theory of PIN models and their fast and stable implementations. However, detailed analyses might be interesting for future research.

[133] The **pinbasic** package can be downloaded and installed from *CRAN* and runs on all popular operating systems.

posterior probabilities in the context of static models. A comparison of the outcomes of this three different techniques for two exemplary stocks is given in section 11.3.

Results of empirical applications of the dynamic models are presented in chapter 11. A major problem of the PIN-ALACD model is the occurrence of corner solutions of coefficients involved in the determination of state probabilities. Results for some equities yield probabilities of good- and bad-news conditions which are identical over the complete time range under consideration (e.g., HMC in 2007 and 2008). Contrary, empirical applications show that our PIN-HMM model is very often able to clearly assign a state to each trading period, even if we switch from days as unit time interval to the alternative of short-ranged intraday intervals. Especially for intraday analyses[134] conducted in this work, the approach by Tay, Ting, Tse, and Warachka (2009) exhibits severe problems since it cannot distinguish between the potential directions of information events at all. Hence, intraday estimation results of the PIN-ALACD model are not reasonable. Also relations of the intercepts in ALACD specifications and estimates of coefficients of lagged signed logarithmic volume are counterintuitive and do not match the hypotheses about their relations proposed in the work by Tay, Ting, Tse, and Warachka (2009). This behavior may be due to the fact that there is a time difference of more than ten years in the utilized datasets. It can also be interpreted that marketplaces evolved in a way that the PIN-ALACD model is not appropriate anymore. In contrast, our new PIN-HMM model delivers reasonable results for almost all symbols and years. It also allows to estimate the proportion of private information in intraday-analyses in a meaningful way. We can read from the relation of estimated intercepts of the ALACD specifications in the PIN-HMM model that buyer- or seller-initiated transactions induce lower conditional expected durations of subsequent buys or sells, respectively. In addition, buys or sells with large order sizes substantially reduce subsequent conditional expected durations of identical trade direction. Both findings appropriately capture characteristics of the prepared datasets we use for estimation purposes.

Despite all the improvements we gain by using the PIN-HMM model to receive estimates of the probability of informed trading, we see by inspecting the goodness-of-fit plots that very short durations still cause problems. A potential starting-point for future research may be to incorporate more general distributions for the conditional interarrival times of buys and sells, e. g. (generalized) gamma, generalized F or Burr distribution, just to name a few candidates. Likewise one could think of generalized ALACD specifications. However, the required computing time should always be kept in mind. Since no algorithm delivering *good* sets of initial values exists in literature, several maximizations with different (random) starting values should be undertaken to increase the chance of reaching the global maximum. Hence, optimization routines of PIN-HMM models with more flexible distribution and more complex ALACD specifications can readily last several days or even longer.

Another interesting point for future research could be the inclusion of overnight-effects in marketplaces. Currently, the first 30 minutes of each trading day are removed from the data to explicitly exclude such effects. However, they may be a very interesting factor, especially

[134] As mentioned in section 11.5, the intraday analyses presented in this work can be understood as *proof of concept* that it is reasonable to apply our dynamic model also to this alternative setting. This specific field of research in the context of the probability of informed trading may be very interesting for future research.

for intraday analyses. This potential sub-field of the forthcoming literature for the probability of informed trading needs investigation in two aspects. On the one side models need to be modified to be able to accurately handle such effects and on the other side the data preparation steps need to be revised.

At the time of writing, no R package with functionalities for estimating the probability of informed trading in terms of dynamic models is officially available. However, we plan to package and publicly release the source codes for model estimation and data preparation presented in chapters 7 and 9, respectively.

References

Aktas, Nihat, de Bodt, Eric, Declerck, Fany, and van Oppens, Hervé (2007). "The PIN anomaly around M&A announcements". In: *Journal of Financial Markets* 10.2, pp. 169–191. DOI: 10.1016/j.finmar.2006.09.003.

Andersen, Torben G. and Bondarenko, Oleg (2014). "VPIN and the flash crash". In: *Journal of Financial Markets* 17, pp. 1–46. DOI: 10.1016/j.finmar.2013.05.005.

Aslan, Hadiye, Easley, David, Hvidkjaer, Soeren, and O'Hara, Maureen (2011). "The characteristics of informed trading: Implications for asset pricing". In: *Journal of Empirical Finance* 18.5, pp. 782–801. DOI: 10.1016/j.jempfin.2011.08.001.

Bagehot, Walter (1971). "The Only Game in Town". In: *Financial Analysts Journal* 27.2, pp. 12–14. DOI: 10.2469/faj.v27.n2.12.

Barndorff-Nielsen, Ole E., Hansen, Peter Reinhard, Lunde, Asger, and Shephard, Neil (2009). "Realized kernels in practice: trades and quotes". In: *Econometrics Journal* 12.3, pp. C1–C32. DOI: 10.1111/j.1368-423X.2008.00275.x.

Basu, Asit P. and Rigdon, Steven E. (2001). "Ch. 2. The weibull nonhomogeneous poisson process". In: *Advances in Reliability*. Vol. 20. Handbook of Statistics. Elsevier, pp. 43–68. DOI: 10.1016/S0169-7161(01)20004-2.

Baum, Leonard E., Petrie, Ted, Soules, George, and Weiss, Norman (1970). "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains". In: *The Annals of Mathematical Statistics* 41.1, pp. 164–171. DOI: 10.1214/aoms/1177697196.

Bauwens, Luc and Giot, Pierre (2000). "The Logarithmic ACD Model: An Application to the Bid-Ask Quote Process of Three NYSE Stocks". In: *Annales d'Économie et de Statistique* 60, pp. 117–149. DOI: 10.2307/20076257.

— (2003). "Asymmetric ACD models: Introducing price information in ACD models". In: *Empirical Economics* 28.4, pp. 709–731. DOI: 10.1007/s00181-003-0155-7.

References

Bauwens, Luc, Giot, Pierre, Grammig, Joachim, and Veredas, David (2004). "A comparison of financial duration models via density forecasts". In: *International Journal of Forecasting* 20.4, pp. 589–609. DOI: 10.1016/j.ijforecast.2003.09.014.

Boehmer, Ekkehart, Grammig, Joachim, and Theissen, Erik (2007). "Estimating the probability of informed trading—does trade misclassification matter?" In: *Journal of Financial Markets* 10.1, pp. 26–47. DOI: 10.1016/j.finmar.2006.07.002.

Brockman, Paul and Chung, Dennis Y. (2008). "Investor protection, adverse selection, and the probability of informed trading". In: *Review of Quantitative Finance and Accounting* 30.2, pp. 111–131. DOI: 10.1007/s11156-007-0049-4.

Brownlees, Christian T. and Gallo, Giampiero M. (2006). "Financial econometric analysis at ultra-high frequency: Data handling concerns". In: *Computational Statistics & Data Analysis* 51.4, pp. 2232–2245. DOI: 10.1016/j.csda.2006.09.030.

Buckland, Stephe T. (1983). "Monte Carlo Methods For Confidence Interval Estimation Using The Bootstrap Technique". In: *Journal of Applied Statistics* 10.2, pp. 194–212. DOI: 10.1080/02664768300000017.

Chakrabarty, Bidisha, Li, Bingguang, Nguyen, Vanthuan, and Ness, Robert A. Van (2007). "Trade classification algorithms for electronic communications network trades". In: *Journal of Banking & Finance* 31.12, pp. 3806–3821. DOI: 10.1016/j.jbankfin.2007.03.003.

Chung, Kee H. and Li, Mingsheng (2003). "Adverse-Selection Costs and the Probability of Information-Based Trading". In: *Financial Review* 38.2, pp. 257–272. DOI: 10.1111/1540-6288.00045.

Copeland, Thomas E. and Galai, Dan (1983). "Information Effects on the Bid-Ask Spread". In: *The Journal of Finance* 38.5, pp. 1457–1469. DOI: 10.2307/2327580.

Couvreur, Christophe (1996). "Hidden Markov Models and Their Mixtures". In: *Dept. Math., Université Catholique de Louvain, Louvain, Belgium*. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.53.994&rep=rep1&type=pdf.

Daley, D. J. and Vere-Jones, D. (2003). *An Introduction to the Theory of Point Processes Volume I: Elementary Theory and Methods*. Second Edition. Springer. DOI: 10.1007/b97277.

Daley, Daryl J. and Vere-Jones, David (2010). *Encyclopedia of Quantitative Finance*. Second Edition. Vol. 3. Wiley. DOI: 10.1002/9780470061602.

Dalgaard, Peter (2008). *Introductory Statistics with R*. Second Edition. Springer. DOI: 10.1007/978-0-387-79054-1.

Diebold, Francis X., Gunther, Todd A., and Tay, Anthony S. (1998). "Evaluating Density Forecasts with Applications to Financial Risk Management". In: *International Economic Review* 39.4, pp. 863–883. DOI: 10.2307/2527342.

Dooley, Isaac and Kale, Laxmikant (2006). "Quantifying the Interference Caused by Subnormal Floating-Point Values". In: *Proceedings of the Workshop on Operating System Interference in High Performance Applications*. URL: https://pdfs.semanticscholar.org/63f5/6ca782803fb2aac01866174f32f27cd84006.pdf.

Duarte, Jefferson and Young, Lance (2009). "Why is PIN priced?" In: *Journal of Financial Economics* 91.2, pp. 119–138. DOI: 10.1016/j.jfineco.2007.10.008.

Easley, David, Engle, Robert F., O'Hara, Maureen, and Wu, Liuren (2008). "Time-Varying Arrival Rates of Informed and Uninformed Trades". In: *Journal of Financial Econometrics* 6.2, pp. 171–207. DOI: 10.1093/jjfinec/nbn003.

Easley, David, Hvidkjaer, Soeren, and O'Hara, Maureen (2002). "Is Information Risk a Determinant of Asset Returns?" In: *The Journal of Finance* 57.5, pp. 2185–2221. DOI: 10.1111/1540-6261.00493.

— (2010). "Factoring Information into Returns". In: *Journal of Financial and Quantitative Analysis* 45.2, pp. 293–309. DOI: 10.1017/S0022109010000074.

Easley, David, Kiefer, Nicholas M., O'Hara, Maureen, and Paperman, Joseph B. (1996). "Liquidity, Information, and Infrequently Traded Stocks". In: *The Journal of Finance* 51.4, pp. 1405–1436. DOI: 10.1111/j.1540-6261.1996.tb04074.x.

Easley, David, López de Prado, Marcos M., and O'Hara, Maureen (2012). "Flow Toxicity and Liquidity in a High-frequency World". In: *The Review of Financial Studies* 25.5, pp. 1457–1493. DOI: 10.1093/rfs/hhs053.

Easley, David and O'Hara, Maureen (1987). "Price, trade size, and information in securities markets". In: *Journal of Financial Economics* 19.1, pp. 69–90. DOI: 10.1016/0304-405X(87)90029-8.

Eddelbuettel, Dirk and François, Romain (2011). "Rcpp: Seamless R and C++ Integration". In: *Journal of Statistical Software* 40.8, pp. 1–18. DOI: 10.18637/jss.v040.i08.

Eddelbuettel, Dirk and Sanderson, Conrad (2014). "RcppArmadillo: Accelerating R with high-performance C++ linear algebra". In: *Computational Statistics and Data Analysis* 71, pp. 1054–1063. DOI: 10.1016/j.csda.2013.02.005.

Ellis, Katrina, Michaely, Roni, and O'Hara, Maureen (2000). "The Accuracy of Trade Classification Rules: Evidence from Nasdaq". In: *The Journal of Financial and Quantitative Analysis* 35.4, pp. 529–551. DOI: 10.2307/2676254.

Engle, Robert F. and Russell, Jeffrey R. (1998). "Autoregressive Conditional Duration: A New Model for Irregularly Spaced Transaction Data". In: *Econometrica* 66.5, pp. 1127–1162. DOI: 10.2307/2999632.

Ersan, Oguz and Alıcı, Aslı (2016). "An unbiased computation methodology for estimating the probability of informed trading (PIN)". In: *Journal of International Financial Markets, Institutions and Money* 43, pp. 74–94. DOI: 10.1016/j.intfin.2016.04.001.

Everitt, Brian S., Landau, Sabine, Leese, Morven, and Stahl, Daniel (2011). *Cluster Analysis*. Fifth Edition. Wiley Series in Probability and Statistics. Wiley. DOI: 10.1002/9780470977811.

Forsythe, George E., Malcolm, Michael A., and Moler, Cleve B. (1979). "Computer Methods for Mathematical Computations". In: *ZAMM - Journal of Applied Mathematics and Mechanics* 59.2, pp. 141–142. DOI: 10.1002/zamm.19790590235.

Fuller, Kathleen P., Van Ness, Bonnie F., and Van Ness, Robert A. (2010). "Is information risk priced for NASDAQ-listed stocks?" In: *Review of Quantitative Finance and Accounting* 34.3, pp. 301–312. DOI: 10.1007/s11156-009-0131-1.

Gan, Quan, Chun, Wei Wang, and Johnstone, David (2015). "A faster estimation method for the probability of informed trading using hierarchical agglomerative clustering". In: *Quantitative Finance* 15.11, pp. 1805–1821. DOI: 10.1080/14697688.2015.1023336.

Gay, David M. (1990). "Usage Summary For Selected Optimization Routines". In: *Computing Science Technical Report* 153. URL: https://ms.mcmaster.ca/~bolker/misc/port.pdf.

Glosten, Lawrence R. and Harris, Lawrence E. (1988). "Estimating the components of the bid/ask spread". In: *Journal of Financial Economics* 21.1, pp. 123–142. DOI: 10.1016/0304-405X(88)90034-7.

Glosten, Lawrence R. and Milgrom, Paul R. (1985). "Bid, ask and transaction prices in a specialist market with heterogeneously informed traders". In: *Journal of Financial Economics* 14.1, pp. 71–100. DOI: 10.1016/0304-405X(85)90044-3.

Goldberg, David (1991). "What Every Computer Scientist Should Know About Floating-point Arithmetic". In: *ACM Comput. Surv.* 23.1, pp. 5–48. DOI: 10.1145/103162.103163.

Grolemund, Garrett and Wickham, Hadley (2011). "Dates and Times Made Easy with lubridate". In: *Journal of Statistical Software* 40.3, pp. 1–25. DOI: 10.18637/jss.v040.i03.

Hasbrouck, Joel (1988). "Trades, quotes, inventories, and information". In: *Journal of Financial Economics* 22.2, pp. 229–252. DOI: 10.1016/0304-405X(88)90070-0.

Hautsch, Nikolaus (2004). *Modelling irregularly spaced financial data: theory and practice of dynamic duration models*. Springer. DOI: 10.1007/978-3-642-17015-7.

— (2011). *Econometrics of Financial High-Frequency Data*. Springer. DOI: 10.1007/978-3-642-21925-2.

Hautsch, Nikolaus, Kyj, Lada M., and Oomen, Roel C. A. (2012). "A blocking and regularization approach to high-dimensional realized covariance estimation". In: *Journal of Applied Econometrics* 27.4, pp. 625–645. DOI: 10.1002/jae.1218.

Henker, Thomas and Wang, Jian-Xin (2006). "On the importance of timing specifications in market microstructure research". In: *Journal of Financial Markets* 9.2, pp. 162–179. DOI: 10.1016/j.finmar.2006.01.001.

Henry, Tyler R. (2006). "Short selling, informed trading, and stock returns". Working Paper, University of Georgia. URL: http://media.terry.uga.edu/documents/finance/henry.pdf.

Holthausen, Robert W., Leftwich, Richard W., and Mayers, David (1987). "The effect of large block transactions on security prices: A cross-sectional analysis". In: *Journal of Financial Economics* 19.2, pp. 237–267. DOI: 10.1016/0304-405X(87)90004-3.

IEEE (1985). "IEEE Standard for Binary Floating-Point Arithmetic". In: *ANSI/IEEE Std 754-1985*. DOI: 10.1109/IEEESTD.1985.82928.

Isaacson, Dean L. and Madsen, Richard W. (1976). *Markov Chains: Theory and Applications*. Wiley series in Probability and Mathematical Statistics. Wiley. ISBN: 0471428620.

Johnson, Norman L., Kemp, Adrienne W., and Kotz, Samuel (2005). *Univariate Discrete Distributions*. Third Edition. Wiley Series in Probability and Statistics. Wiley. DOI: 10.1002/0471715816.

Johnson, Norman L., Kotz, Samuel, and Balakrishnan, Narayanaswamy (1994). *Continuous Univariate Distributions*. Second Edition. Wiley series in Probability and Mathematical Statistics Volume 1. Wiley. ISBN: 0471584959.

Kang, Moonsoo (2010). "Probability of information-based trading and the January effect". In: *Journal of Banking & Finance* 34.12, pp. 2985–2994. DOI: 10.1016/j.jbankfin.2010.07.007.

Kiefer, Nicholas M. (1988). "Economic Duration Data and Hazard Functions". In: *Journal of Economic Literature* 26.2, pp. 646–679. DOI: 10.2307/2726365.

Konstantopoulos, Takis (2009). "Introductory Lecture Notes on Markov Chains and Random Walks". In: *Department of Mathematics, Uppsala University* 200.9. URL: http://www.bioinfo.org.cn/~wangchao/maa/mcrw.pdf.

Kwok, Simon Sai Man, Li, Wai Keung, and Yu, Philip Leung Ho (2009). "The Autoregressive Conditional Marked Duration Model: Statistical Inference to Market Microstructure". In: *Journal of Data Science* 7.2, pp. 1127–1162. URL: http://www.jds-online.com/file_download/205/JDS-438.pdf.

Lee, Charles M. C. and Ready, Mark J. (1991). "Inferring Trade Direction from Intraday Data". In: *The Journal of Finance* 46.2, pp. 733–746. DOI: 10.1111/j.1540-6261.1991.tb02683.x.

Lee, Charles M.C. and Radhakrishna, Balkrishna (2000). "Inferring investor behavior: Evidence from {TORQ} data". In: *Journal of Financial Markets* 3.2, pp. 83–111. DOI: 10.1016/S1386-4181(00)00002-1.

Lee, Peter M. (2012). *Bayesian Statistics: An Introduction*. Fourth Edition. Wiley. ISBN: 1118332571.

Lei, Qin and Wu, Guojun (2005). "Time-varying informed and uninformed trading activities". In: *Journal of Financial Markets* 8.2, pp. 153–181. DOI: 10.1016/j.finmar.2004.09.002.

Li, Haitao, Wang, Junbo, Wu, Chunchi, and He, Yan (2009). "Are Liquidity and Information Risks Priced in the Treasury Bond Market?" In: *The Journal of Finance* 64.1, pp. 467–503. DOI: 10.1111/j.1540-6261.2008.01439.x.

Lin, Hsiou-Wei William and Ke, Wen-Chyan (2011). "A computing bias in estimating the probability of informed trading". In: *Journal of Financial Markets* 14.4, pp. 625–640. DOI: 10.1016/j.finmar.2011.03.001.

Madhavan, Ananth (2000). "Market microstructure: A survey". In: *Journal of Financial Markets* 3.3, pp. 205–258. DOI: 10.1016/S1386-4181(00)00007-0.

Maechler, Martin (2015). *Rmpfr: R MPFR - Multiple Precision Floating-Point Reliable*. R package version 0.6-0. URL: https://CRAN.R-project.org/package=Rmpfr.

Manaster, Steven and Mann, Steven C. (1996). "Life in the Pits: Competitive Market Making and Inventory Control". In: *The Review of Financial Studies* 9.3, pp. 953–975. DOI: 10.1093/rfs/9.3.953.

Matsumoto, Makoto and Nishimura, Takuji (1998). "Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-random Number Generator". In: *ACM Trans. Model. Comput. Simul.* 8.1, pp. 3–30. DOI: 10.1145/272991.272995.

Meitz, Mika and Teräsvirta, Timo (2006). "Evaluating Models of Autoregressive Conditional Duration". In: *Journal of Business & Economic Statistics* 24.1, pp. 104–124. DOI: 10.1198/073500105000000081.

Mohanram, Partha and Rajgopal, Shiva (2009). "Is PIN priced risk?" In: *Journal of Accounting and Economics* 47.3, pp. 226–243. DOI: 10.1016/j.jacceco.2008.10.001.

Müllner, Daniel (2013). "fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python". In: *Journal of Statistical Software* 53.9, pp. 1–18. DOI: 10.18637/jss.v053.i09.

Nash, John (2014). "On Best Practice Optimization Methods in R". In: *Journal of Statistical Software, Articles* 60.2, pp. 1–14. DOI: 10.18637/jss.v060.i02.

Nexa Technologies Inc. (2007). *Deutsche Börse Equity Trade and Quote Data File Format Document*. Version 1.2; newer versions available online. URL: https://s3-us-west-2.amazonaws.com/tick-data-s3/pdf/TickData_File_Format_Overview_Deutsche_Borse.pdf.

— (2011). *US Equity Trade, Quote and One-Minute Data File Format Document*. Version 2.87; newer versions available online. URL: https://s3-us-west-2.amazonaws.com/tick-data-s3/pdf/TickData_File_Format_Overview_US_Equities.pdf.

Odders-White, Elizabeth R. (2000). "On the occurrence and consequences of inaccurate trade classification". In: *Journal of Financial Markets* 3.3, pp. 259–286. DOI: 10.1016/S1386-4181(00)00006-9.

Øksendal, Bernt (2010). *Stochastic Differential Equations: An Introduction with Applications*. Sixth Edition. Springer. DOI: 10.1007/978-3-642-14394-6.

Pacurar, Maria (2008). "Autoregressive Conditional Duration Models in Finance: A Survey of the Theoretical and Empirical Literature". In: *Journal of Economic Surveys* 22.4, pp. 711–751. DOI: 10.1111/j.1467-6419.2007.00547.x.

Pöppe, Thomas, Aitken, Michael, Schiereck, Dirk, and Wiegand, Ingo (2016). "A PIN per day shows what news convey: the intraday probability of informed trading". In: *Review of Quantitative Finance and Accounting* 47.4, pp. 1187–1220. DOI: 10.1007/s11156-015-0535-z.

Preve, Daniel and Tse, Yiu-Kuen (2013). "Estimation of Time-Varying Adjusted Probability of Informed Trading and Probability of Symmetric Order-Flow Shock". In: *Journal of Applied Econometrics* 28.7, pp. 1138–1152. DOI: 10.1002/jae.2302.

Quarteroni, Alfio, Sacco, Riccardo, and Saleri, Fausto (2007). *Numerical Mathematics*. Second Edition. Texts in applied mathematics. Springer. DOI: 10.1007/b98885.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. URL: https://www.R-project.org/.

Rabiner, Lawrence R. (1989). "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2, pp. 257–286. DOI: 10.1109/5.18626.

Rabiner, Lawrence R. and Juang, Biing-Hwang (1986). "An introduction to hidden Markov models". In: *IEEE ASSP Magazine* 3.1, pp. 4–16. DOI: 10.1109/MASSP.1986.1165342.

Roll, Richard (1984). "A Simple Implicit Measure of the Effective Bid-Ask Spread in an Efficient Market". In: *The Journal of Finance* 39.4, pp. 1127–1139. DOI: 10.1111/j.1540-6261.1984.tb03897.x.

Rosenblatt, Murray (1952). "Remarks on a Multivariate Transformation". In: *The Annals of Mathematical Statistics* 23.3, pp. 470–472. DOI: 10.1214/aoms/1177729394.

Rosenthal, Dale W. R. (2012). "Modeling Trade Direction". In: *Journal of Financial Econometrics* 10.2, pp. 390–415. DOI: 10.1093/jjfinec/nbr014.

Ross, Sheldon M. (1996). *Stochastic processes*. Wiley series in Probability and Statistics. Wiley. ISBN: 0471120626.

Snyder, Donald L. and Miller, Michael I. (1991). *Random Point Processes in Time and Space*. Springer. DOI: 10.1007/978-1-4612-3166-0.

Tay, Anthony, Ting, Christopher, Tse, Yiu Kuen, and Warachka, Mitch (2007). "Modeling Transaction Data of Trade Direction and Estimation of Probability of Informed Trading". In: *SMU Economics & Statistics Working Paper Series*. URL: http://www.eaber.org/sites/default/files/documents/smu_tay_2007_02.pdf.

— (2009). "Using High-Frequency Transaction Data to Estimate the Probability of Informed Trading". In: *Journal of Financial Econometrics* 7.3, pp. 288–311. DOI: 10.1093/jjfinec/nbp005.

Theissen, Erik (2000). "A Test of the Accuracy of the Lee/Ready Trade Classification Algorithm". In: *Journal of International Financial Markets, Institutions and Money* 11.2, pp. 147–165. DOI: 10.1016/S1042-4431(00)00048-2.

Tick Data, LLC. (2011). *TickWrite 7.1 User Instructions*. Database Management Software For High-Frequency Global Market Data; version 20110701; newer versions available online. URL: https://tick-data-s3.s3.amazonaws.com/pdf/TickWrite7_Manual.pdf.

— (2019). *Research-Quality Historical Market Data Solutions*. URL: www.tickdata.com (visited on 02/27/2019).

Tijms, Henk C. (2003). *A first course in stochastic models*. Wiley. DOI: 10.1002/047001363X.

Tsay, Ruey S (2010). *Analysis of financial time series*. Third Edition. Wiley Series in Probability and Statistics. Wiley. ISBN: 0470414359.

Vergote, Olivier (2005). "How to Match Trades and Quotes for NYSE Stocks?" In: *SSRN working papers*. DOI: 10.2139/ssrn.808984.

Wald, Abraham and Wolfowitz, Jacob (1940). "On a Test Whether Two Samples are from the Same Population". In: 11.2, pp. 147–162. DOI: 10.1214/aoms/1177731909.

Wickham, Hadley (2007). "Reshaping Data with the reshape Package". In: *Journal of Statistical Software* 21.12, pp. 1–20. DOI: 10.18637/jss.v021.i12.

— (2009). *ggplot2: Elegant Graphics for Data Analysis*. Springer. DOI: 10.1007/978-0-387-98141-3.

— (2016). *scales: Scale Functions for Visualization*. R package version 0.4.0. URL: https://CRAN.R-project.org/package=scales.

Wickham, Hadley, James, David A., and Falcon, Seth (2014). *RSQLite: SQLite Interface for R*. R package version 1.0.0. URL: https://CRAN.R-project.org/package=RSQLite.

Yan, Yuxing and Zhang, Shaojun (2012). "An improved estimation method and empirical properties of the probability of informed trading". In: *Journal of Banking & Finance* 36.2, pp. 454–467. DOI: 10.1016/j.jbankfin.2011.08.003.

Yang, Kyung-won and Lee, Tai-yong (2010). "Heuristic scaling method for efficient parameter estimation". In: *Chemical Engineering Research and Design* 88.5, pp. 520–528. DOI: 10.1016/j.cherd.2009.09.017.

Yin, Xiangkang and Zhao, Jing (2015). "A Hidden Markov Model Approach to Information-Based Trading: Theory and Applications". In: *Journal of Applied Econometrics* 30.7, pp. 1210–1234. DOI: 10.1002/jae.2412.

Zhou, Rhea Tingyu and Lai, Rose Neng (2009). "Herding and information based trading". In: *Journal of Empirical Finance* 16.3, pp. 388–393. DOI: 10.1016/j.jempfin.2009.01.004.

Zucchini, Walter and MacDonald, Iain L. (2009). *Hidden Markov Models for Time Series: An Introduction using R*. Second Edition. Vol. 22. Chapman & Hall/CRC Monographs on Statistics and Applied Probability. CRC Press. DOI: 10.1201/b20790.

Zygmund, Antoni and Fefferman, Robert (2003). *Trigonometric Series*. Third Edition. Cambridge Mathematical Library. Cambridge University Press. DOI: 10.1017/CBO9781316036587.