Statistical methods for conditional volatility modeling with applications to finance

Samir Orujov

COLLEGE MATHSTIC
DOCTORAL BRETAGNE
BRETAGNE OCEANE

Université Bretagne Sud

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE BRETAGNE SUD

ÉCOLE DOCTORALE N° 644
Mathématiques et Sciences et Technologies de l'Information et de la Communication en Bretagne Océane
Spécialité : *Mathématiques et leurs interactions*

Par
Samir ORUJOV

Statistical methods for conditional volatility modeling with applications to finance

Thèse présentée et soutenue à Vannes, le 19 décembre 2024
Unité de recherche : Laboratoire de Mathématiques de Bretagne Atlantique
Thèse N° : 709

Rapporteurs avant soutenance :

Sophie Dabo	Professeure des universités, Université de Lille
Badih Ghattas	Professeur des universités, Université Aix-Marseille

Composition du Jury :

Président :	Sylvain Le Corff	Professeur des universités, Université Sorbonne
Examinateurs :	Adil Bagirov	Professor, Federal University, Australia
	Ion Grama	Professeur des universités, Université Bretagne Sud
Dir. de thèse :	François Septier	Professeur des universités, Université Bretagne Sud
Co-enc. de thèse :	Audrey Poterie	Maîtresse de conférences, Université Bretagne Sud
	Victor Elvira	Professor, University of Edinburgh, UK

Acknowledgement

Je tiens à exprimer ma profonde gratitude envers mes directeurs de thèse pour leur encadrement et leurs précieux conseils tout au long de mon parcours doctoral. Je reconnais que toutes les erreurs et imperfections de cette thèse sont entièrement les miennes.

Je souhaite également remercier le Professeur Alain Coulon pour son soutien indéfectible durant ma thèse, ainsi que le Docteur Florence Paturel pour son aide précieuse lors de mes problèmes de santé. Un remerciement particulier au Professeur Ion Grama à l'UBS, qui a généreusement consacré son temps à discuter et résoudre des problèmes mathématiques avec moi.

Un immense merci à mes parents pour leur soutien constant et inconditionnel.

Je suis également reconnaissant envers l'Université ADA en Azerbaïdjan, en particulier le Recteur Professeur Dr. Hafiz Pashayev, le Vice-Recteur Elkin Nurmammadov, le Doyen Huseyn Ismayilov et tous les autres membres du personnel, pour leur soutien financier et moral qui a été essentiel à la réalisation de mon doctorat.

Je tiens à remercier le Professeur Pini Davidov du Collège Azrieli pour son aide dans la recherche d'un article important à la bibliothèque de l'Université Technion à Haïfa et pour m'avoir envoyé une copie numérique.

Je souhaite également exprimer ma reconnaissance envers la société Total et l'Université ADA pour leur soutien financier, ainsi qu'à leurs employés pour leur aide tout au long de mon doctorat.

Enfin, je remercie tous ceux qui ont contribué, directement ou indirectement, à l'achèvement de cette thèse.

I would like to express my deepest gratitude to my thesis supervisors for their guidance and invaluable advice throughout my doctoral journey. I acknowledge that all mistakes and shortcomings in this thesis are solely my own.

I also wish to thank Professor Alain Coulon for his unwavering support during my PhD, and Doctor Florence Paturel for her assistance with my physical challenges when needed. A special thanks to Professor Ion Grama at UBS, who generously dedicated his time to discuss and solve mathematical problems with me.

A heartfelt thanks to my parents for their constant and unconditional support.

I am grateful to ADA University in Azerbaijan, especially Rector Professor Dr. Hafiz Pashayev, Vice Rector Elkin Nurmammadov, Dean Huseyn Ismayilov, and all other staff

members, for their financial and moral support which was essential for the completion of my PhD.

I would like to thank Professor Pini Davidov at Azrieli College for his assistance in locating an important article from the Technion University library in Haifa and for providing me with a soft copy.

I also acknowledge the financial support from Total company and ADA University, and extend my thanks to their employees for their support throughout my doctoral studies.

Finally, I would like to thank everyone who contributed, directly or indirectly, to the completion of this thesis.

Table of Contents

Chapter 1

Introduction

It is a well-established fact that returns of a financial asset is directly related to the risk of holding that asset (Sharpe, 1964; Lintner, 1965). Therefore, modeling and/or estimating the risk as measured by volatility is very important in terms of financial risk management, option pricing and portfolio optimization. Moreover, all financial instruments regardless of the financial market they belong to and the period of analysis, share common statistical properties which are called stylized empirical facts (Cont, 2001). Interestingly, several of the stylized facts considered in Cont (2001) are related to volatility of financial instruments. One of them is the so called *volatility clustering* phenomenon which is reflected in persistence of volatility magnitudes. More precisely, volatility clustering means that large volatility periods are likely to be followed by large, and small volatility periods are likely to be followed by small volatility periods. The second stylized fact is called the *leverage effect* in finance which implies the negative correlation between returns and volatility. The third stylized fact that are interesting for the purposes of this thesis is *asymmetry in time scales* which means that low frequency measures of volatility predict the high frequency volatility better than other way around (for a detailed exposition of the stylized facts see (Cont, 2001)). Moreover, volatility is regarded as an important determinant of the efficiency of the financial market with low volatility markets to me more efficient compared to high volatility ones (Park and Linton, 2012). All the latter points emphasize the importance of volatility modeling. However, it turns out that volatility of a financial asset is not an easy quantity to measure in the first place (Park and Linton, 2012) which further complicates its modeling. The major reason for that is the latent nature of volatility of a financial instrument. To formalize, let us follow the notation of Bauwens et al. (2012) and suppose that price and return of a financial instrument follows a continuous-time diffusion process given below:

$$dp(t) = \mu(t)dt + \sigma(t)dW(t),\ t \geq 0, \tag{1.1}$$

where $dp(t)$ is the change in natural logarithm of prices at an infinitesimal time interval dt, $\mu(t)$ is a continuous locally bounded variation process, $\sigma(t)$ is strictly positive stochastic volatility process which is right continous with left limit and $W(t)$ is a standard Brownian motion. From (1.1) it follows that one period return, r_t, of the financial instrument defined as the logarithmic difference in its prices between two successive periods is

$$r_t = p(t) - p(t-1) = \int_{t-1}^{t} \mu(s)ds + \int_{t-1}^{t} \sigma(s)dW(s). \tag{1.2}$$

It was shown that daily returns, r_t, obey the following conditional normal probability law

$$r_t \mid \mathcal{I}_t \sim N\Bigg(\int_{t-1}^{t} \mu(s)ds, \underbrace{\int_{t-1}^{t} \sigma(s)^2 ds}_{IV_t} \Bigg), \tag{1.3}$$

where $\mathcal{I}_t$ denotes the information set and IV_t is the integrated volatility at time t (Barndorff-Nielsen and Shephard, 2002). More precisely, IV_t is latent because it depends on unobservable spot volatility, $\sigma(s)$ (Bauwens et al., 2012).

In this thesis, we deal with two different classes of discrete time conditional volatility modeling which address the latency of volatility in different ways. The first type of conditional volatility models exposed in this work is logarithmic generalized conditional heteroscedasticity (log-GARCH) class of models that was first proposed by Pantula (1986). It is the extension of the (generalized) autoregressive conditional heteroskedasticity ((G)ARCH) class of models developed by Engle (1982) and Bollerslev (1986), respectively. GARCH class of models and its extensions infer the latent volatility from a link function. The link function models the conditional volatility at time t as a linear function of the past values of returns $r_{t-1}, r_{t-2} \ldots$ (Bauwens et al., 2012). In this thesis, we contribute to the existing literature on log-GARCH models. A brief description of our contribution is provided in Section 1.2.1.

The second class of models entertained in this thesis is the *heterogenous autoregressive model of realized volatility* (HAR) model proposed by Corsi (2009). The HAR class of models adopts an autoregressive structure for volatility. Additionally, instead of the latent integrated volatility defined in (1.3), HAR class focuses on modeling the consistent estimator of the integrated volatility. In the seminal work of Andersen and Bollerslev (1998), it has been shown that the sum of the intraday squared returns (referred to as the *daily realized volatility*) provide a consistent and less noisy estimator for the integrated volatility compared to daily squared returns (the square of the logarithmic difference between

close and open price of an asset during a day). In Section 1.1.2, we elaborate more on the daily realized volatility concept. Indeed, till the publication of this important paper daily squared returns which are unbiased but noisy estimators of the integrated volatility were used as a proxy (estimator) of the true underlying volatility in the related literature (Bauwens et al., 2012).

Futhermore, as Park and Linton (2012) put it, suppose an investor facing returns r_t and information set $\mathcal{I}_s$ at time $s < t$ is interested in the conditional variance of returns $Var(r_t \mid \mathcal{I}_s)$. Measuring the latter might be complicated due to the fact that the time increment $t-s$ might be unobservable or stochastic or the information set $\mathcal{I}_t$ may be high dimensional, and the distribution $f(r_t \mid \mathcal{I}_s)$ may be unknown (Park and Linton, 2012). Indeed, in Chapter 2 of the thesis we are addressing the concern of the high dimensional information set by incorporating the variable selection layer into the estimation process of a specific class of volatility models.

Additionally, we use daily realized volatility to evaluate the performance of one-step-ahead forecasts generated from log-GARCH type models in Chapter 2. This is in accordance with the stylized fact - asymmetry in time scales touched upon previously and also with the current literature (Bergsli et al., 2022).

In addition, in this present thesis, we pay attention to the other two stylized facts, namely the leverage effect and the volatility clustering (Cont, 2001). Indeed, in Chapter 2 of the thesis, the conditioned information set for volatility modeling includes asymmetry terms to capture the leverage effect which is in line with Sucarrat (2019). In Chapter 3, we estimate the clusterwise HAR model which is important to capture the leverage effect and to estimate the state (equivalently, cluster or regime) specific volatility persistence parameters. In Chapter 4, we propose a new algorithm identify the states (regimes) and to estimate parameters of Open Loop Threshold Autoregressive (OLTAR) models (Chen, 1995) which is a special case of the Threshold Autoregressive (TAR) models (Tong, 1983) popular in non-linear time series modeling. Moreover, in the latter chapter, we propose a useful extension of the generic HAR model by blending it with OLTAR model. We call this new extension as OLTAR-HAR model and show that the model outperforms the benchmark HAR model when applied to Bitcoin realized daily volatility data.

To establish the context for our contributions to the related literature, the next section reviews the GARCH class of volatility models, with an emphasis on log-GARCH type models. Additionally, we provide an overview of the HAR class of volatility models and some of its extensions.

1.1 Research Motivations and Key Objectives

Volatility is an important quantity but its measurement and modeling poses challenges as mentioned previously. Some class of discrete time series models of conditional volatility such as GARCH class, assume that the underlying volatility is latent and try to infer it from data. On the other hand, the other class of discrete time series models for conditional volatility such as HAR models replace the underlying volatility by its proxy, namely, with the realized volatility. In this thesis, we propose some extensions of the latter models and apply them to Bitcoin returns. Bitcoin market has drawn attention of academicians of different fields since its inception (Sapovadia, 2015; Guo et al., 2018; Bergsli et al., 2022). Due to specific underlying technology, unprecedented volatility of its price and public interest we want to test the discrete time series models proposed in this thesis in Bitcoin markets.

It is worth to mention that, the proposed models in this thesis are estimated using novel estimation techniques. The presented estimation techniques are heavily based on the well-known *cross entropy* method for continuous and discrete optimization (Rubinstein, 1999). Chapter 3 learns some of the properties of the cross entropy method and modifies the generic method to estimate the proposed extensions of HAR models. By doing so, we contribute to optimization and clustering literature because the method designed in Chapter 3 can be applied to the wide range of clustering problems. In Chapter 4, we propose yet another novel estimation technique, but this time for estimating OLTAR models and the new hybrid OLTAR-HAR models. The latter chapter adds to the non-linear time series modeling and volatility literature.

In the next section, we briefly review the literature on GARCH and HAR class of volatility models as they are essential for this thesis.

1.1.1 GARCH Class of Volatility Models

GARCH class of models is one of the most popular class to model volatility by practitioners (Franses and Dijk, 2000). There are many extensions of GARCH models such as Integrated GARCH (IGARCH), Fractionally Integrated GARCH (FIGARCH), Stochastic Volatility (SV), Exponential GARCH (EGARCH), logarithmic GARCH (log-GARCH), Glosten, Jagannathan and Runkle GARCH (GJR-GARCH), Volatility Switching GARCH etc (for a detailed survey of GARCH and extensions see (Teräsvirta, 2009; Franses and Dijk, 2000; Francq and Zakoian, 2011)). This class of models assumes that the volatility

is latent and seeks to infer the latter through the conditional volatility equation. More specifically, the GARCH class of models assumes that a covariance stationary process is given by

$$y_t = \mu_t + \epsilon_t, \quad \epsilon_t = \sigma_t \eta_t, \quad \sigma_t > 0, \quad \eta_t \sim iid\,(0,1)\,, \tag{1.4}$$

where $\{y_t\}_{t\geq 0}$ is the time series process (e.g. financial return), $\mu_t = \mathbb{E}(y_t \mid \mathcal{I}_t)$ is the conditional mean with a functional form known *a priori* but whose parameters need to be estimated, $\sigma_t^2 = Var(y_t \mid \mathcal{I}_t)$ is the conditional variance of y_t, $\mathcal{I}_t$ is the conditioning information set at time t and the error term ϵ_t is a random variable with mean zero and variance σ_t^2. The term η_t is an independently and identically distributed innovation process with zero mean and unit variance, as indicated by $iid\,(0,1)$.

In (1.4), the conditional variance of y_t can evolve over time, while the unconditional variance is assumed to be fixed (which is implied by the stationarity of y_t). The GARCH model of Bollerslev (1986) assumes linear dynamics for the conditional variance of the ϵ_t process in (1.4). Formally, a GARCH(p_1,p_2) model is defined by

$$\begin{aligned} &\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p_1} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p_2} \alpha_{p_1+j} \sigma_{t-j}^2, \\ &\alpha_0 > 0, \quad \alpha_i \geq 0, \quad \forall i \in \{1,\ldots,p_1+p_2\}, \end{aligned} \tag{1.5}$$

where α_0 is the volatility intercept and $p_1 \in \mathbb{N}$ and $p_2 \in \mathbb{N}$ are ARCH (squared lags of ϵ_t) and GARCH (squared lags of σ_t) orders respectively. Note that if $p_2 = 0$, the GARCH(p_1,p_2) model is the ARCH(p_1) model.

Furthermore, the natural extension of the GARCH model in which exogenous variables can be added is called the GARCHX model (Sucarrat, 2020). The inclusion of exogenous covariates into the GARCH volatility equation comes with the additional cost of non-negativity restrictions on both the exogenous variables $\{x_{k,t}\}_{k=1}^{p_3}$ and their associated parameters $\{\beta_k\}_{k=1}^{p_3}$ in order to guarantee the positivity of the conditional variance. Furthermore, inference procedures under nullity are non-standard (Francq and Thieu, 2019; Sucarrat, 2021) and there is a need for alternative asymptotics, as in (Pedersen and Rahbek, 2019). This restrictive nature of most members of the GARCH family models is relaxed in the *exponential GARCH model with exogenous variables* (EGARCHX) (Nelson, 1991) and *logarithmic GARCH model with exogenous variables* (log-GARCHX). The log-GARCHX model is indeed the restricted form of *logarithmic GARCH model with exogenous covariates and threshold variables* (log-TGARCHX), as we emphasize herein

further. Therefore, the terms log-GARCHX and log-TGARCHX are sometimes used interchangeably throughout this work. In the last two members of the GARCH family (EGARCHX and log-TGARCHX), predicted volatility is guaranteed to be positive, and non-negativity constraints on parameters and covariates are lifted. However, the advantage of log-GARCHX over EGARCHX is that unconditional variance in the EGARCH formulation may not exist if the errors are too fat-tailed (Sucarrat, 2019). Moreover, probabilistic properties of EGARCH and log-GARCH models have been studied, and it was shown that the consistency and asymptotic normality of the quasi-maximum likelihood estimation (QMLE) of log-GARCH models hold under conditions milder than those of EGARCH. Furthermore, log-GARCH models may capture richer asymmetry dynamics and its shock transmission mechanism may fit the reality better compared to EGARCH and GARCH (Francq et al., 2013).

Additionally, sometimes the log-GARCHX model with asymmetry terms is referred to as the asymmetric log-GARCHX model in the literature (Sucarrat, 2019). However, we suggest calling the latter log-TGARCHX because it is equivalent to the TGARCHX model of Glosten et al. (1993) except for the logarithmic transformation.

Arguably, one of the most attractive features of the log-TGARCHX model is that it accepts the ARMA representation and therefore the parameter estimation can be carried out in any standard statistical software using least squares and quasi-maximal likelihood (Francq and Thieu, 2019; Sucarrat, 2019). The possibility of the ARMAX representation paves the way for a wide range of extensions of the log-GARCH-type models. In this thesis, we propose to enhance log-TGARCHX models by including a variable selection procedure during model estimation. Indeed, this is very crucial as the information set $\mathcal{I}_t$ available at time t may be enourmously large as mentioned previously (see also Park and Linton (2012)). Also, the exogenous determinants of the traditional financial markets have been well documented in the literature (Corradi et al., 2013; Asgharian et al., 2023; Ma et al., 2024). However, we believe that due to the specific nature of cryptocurrency markets selecting important exogenous covariates from a large set of variables may be essential. For example, variables related to the blockchain technology and their link to Bitcoin volatility have not been entertained enough in literature.

1.1.2 HAR Class of Volatility Modeling

Volatility models proposed in the literature seek to model volatility dynamics in a way to internalize its well-known properties such as volatility clustering, slow decaying

autocorrelation structure and nonlinear response to different type of market information. Fractional integration and hyperbolic decay of autocorrelation function are used in some members of GARCH class models to generate long-memory volatility predictions to be in line with the mentioned stylized facts of financial makets. However, the estimation of this models may be challenging. Heterogenous autoregressive (HAR) models proposed by Corsi (2009) turn out to be very simple yet very powerful. They are able to capture the volatility dynamics of financial markets quite well, provide an excellent fit to data and possess an economic interpretation (Corsi et al., 2012).

The HAR model in its simplest form is specified as follows:

$$RV_{t+1}^{(d)} = \alpha_0 + \alpha_1 RV_t^{(d)} + \alpha_2 RV_t^{(w)} + \alpha_3 RV_t^{(m)} + \eta_{t+1}, \tag{1.6}$$

where $\{RV_t^{(d)}\}_{t\in\mathbb{N}}$ is the time series of daily realized volatility and t is the time (day) index, $RV_t^{(w)}$ and $RV_t^{(m)}$ are weekly and monthly realized volatilities, respectively, measured at daily frequency. Formally, let R_t, the return at the t-th day, be defined as

$$R_t = \ln P_t - \ln P_{t-1}, \tag{1.7}$$

where P_t is the close price of a financial asset on day t. Then, we define m equally spaced intraday returns between each consecutive two days t and $t+1$. More precisely, the j-th intraday return for the the t-th day is denoted by $R_{t+\frac{j}{m}}$ and defined as:

$$R_{t+\frac{j}{m}} = \ln\left(P_{t+\frac{j}{m}}\right) - \ln\left(P_{t+\frac{j-1}{m}}\right), \quad \text{for } j = 0, \ldots, m-1, \tag{1.8}$$

where $\ln\left(P_{t+\frac{j}{m}}\right)$ (resp. $\ln\left(P_{t+\frac{j-1}{m}}\right)$ is the logarithm of the intraday price of the financial asset at time $t+\frac{j}{m}$ (resp. $t+\frac{j-1}{m}$). Thus, the daily realized variance (volatility), for the t-th day can be defined as the sum of squared intraday returns following the previous reference:

$$RV_t^{(d)} = \sum_{j=0}^{m-1} R_{t+\frac{j}{m}}^2. \tag{1.9}$$

where the subscript d is used to emphasize the daily frequency of the realized variance. Corsi (2009) defines weekly and monthly realized volatilities as 5 days and 22 days moving average of the daily realized variance, respectively, which correspond to number of working days in a week and month. However, for the purposes of this thesis weekly realized volatility is 7 days and monthly realized volatility is 30 days moving average of the daily

realized volatility. The reason for this is that cryptocurrency markets are active non-stop.

The core idea behind the HAR models as Corsi (2009) points out, lies in the fact that financial markets consist of heterogenous agents with different trading frequency and horizons. Due to this fact, different markent participants are perceving, reacting and creating different types of risks (volatilities) in financial markets. Therefore, HAR models are including measures of realized volatility at different horizons. However, as confirmed by Corsi et al. (2012), the generic HAR model may fail to properly capture the stylized facts about volatility mentioned previously in this thesis and structural breaks and other nonlinearities. Therefore, the authors suggest extensions of the generic HAR models which account for structural breaks (regime switches) to better reflect the stylized facts. Such a flexibility increases the ability of HAR models to capture the known volatility dynamics. Interestingly, we have not encountered the clusterwise HAR model in the related literature which is a simple form to account for difference in states (regimes) of time series. Clusterwise linear regression model first introduced by Späth (1979) considers fitting more than one regression surfaces to data at hand simultaneously, to account for nonlinearities. The application of the clusterwise linear regression to time series data is referred to as *digression* in (Chen, 1995).

In Chapter 3, we propose a novel and very simple extension of the HAR model, called clusterwise HAR model with K clusters corresponding to different volatility periods to better capture the volatility dynamics inherent in financial markets and documented in finance literature (see (Cont, 2007; 2001)). The simple clusterwise HAR model with two clusters is defined as follows:

$$\begin{aligned} RV_{t+1}^{(d)} =& \mathbb{1}_{\{\boldsymbol{x}_t^\star=1\}}(\alpha_{10} + \alpha_{11}RV_t^{(d)} + \alpha_{12}RV_t^{(w)} + \alpha_{13}RV_t^{(m)}) \\ &+ \mathbb{1}_{\{\boldsymbol{x}_t^\star=2\}}(\alpha_{20} + \alpha_{21}RV_t^{(d)} + \alpha_{22}RV_t^{(w)} + \alpha_{23}RV_t^{(m)}) + \eta_{t+1}, \end{aligned} \tag{1.10}$$

where η_t is the error term with zero mean at time t, $\boldsymbol{x}_k^*$ denotes the optimal volatility cluster and a_{kj} for $k = 1, 2$ and $j = 0, 1, 2, 3$ are coefficients to be estimated.

The mathematical structure of (1.10) is exactly the same as the digression model in Chen (1995), however, we are first to apply it to the financial context of the generic HAR models. Moreover, our main contribution in that chapter is the development of a novel methodology to estimate the parameters and cluster labels ($x_t^\star$ in (1.10)) of the clusterwise HAR model. Although clusterwise HAR in Chapter 3 has shown an enourmous improvement over the generic HAR in terms of the fit to data it is of limited use for prediction (forecasting) purposes. It is because the movement of time series from one

cluster to another is assumed to be exogenous in clusterwise HAR model.

In Chapter 4, we endogenize the cluster assignment. More precisely, we assume that the assignment of observations to clusters (states or regimes in time series vocabulary) are latent, but driven by the so-called *threshold variables*. In addition, the potential threshold variables are a subset of a larger and observable set of variables. Such kind of time series model is referred to as *open loop threshold autoregressive model (OLTAR)* in (Chen, 1995). The estimation of such models poses a challenge because both the states and the threshold mehanism determining the states are latent. Similar to the algorithms developed by (Chen, 1995), we propose a new algorithm to estimate the OLTAR model parameters, and learn the hidden state dynamics using the well known machine learning algorithm CART (Breiman et al., 1984). The validity of the method and consistency of the parameter estimation is asserted by simulations. However, the mathematical proof of the consistency is still an open question.

1.1.3 Summary of Objectives

This thesis addresses the challenges in modeling and estimating volatility in financial time series, with a specific focus on enhancing existing models and proposing new algorithms. The key objectives of this work are:

— To improve variable selection and estimation when dealing with log-TGARCHX models.

— To extend both HAR and OLTAR model so that the models can better capture volatility dynamics pinned down in financial literature.

— To propose novel techniques to estimate the parameters of the discrete time series models of conditional volatility proposed in this thesis.

These objectives aim to contribute both to theoretical advancements in time series modeling and to practical applications, particularly in the context of Bitcoin volatility.

1.2 Outline of the Thesis and Contributions

This thesis is structured into three main chapters, each addressing a significant aspect of volatility modeling and estimation. In Chapter 2, we introduce a novel variable selection algorithm for log-GARCHX models, applying it to Bitcoin market data to demonstrate its improved forecasting performance. Chapter 3 focuses on clustering methods applied

to HAR model, proposing a novel hybrid algorithm that combines Cross Entropy and K-means techniques to enhance the estimation accuracy.. Chapter 4 extends this work by developing a new OLTAR-HAR model, which better captures regime changes in volatility, and applying it to real-world Bitcoin data.

The contributions of this work span both methodological advancements and empirical applications, providing new tools for researchers and practitioners dealing with volatility in financial markets.

1.2.1 Chapter 2: A Flexible Variable Selection Algorithm for log-GARCHX Models

The contributions of this work are twofold. First, we propose an algorithm called VS-LTGARCHX to incorporate the variable selection procedure into the the univariate log-TGARCHX estimation process. Incidentally, variable selection within the context of ARCH and log-ARCHX models has been extensively studied in the literature. For instance, a recent paper by Nielsen and Rahbek (2024) employs a penalized quasi-likelihood approach to estimate and perform model selection for ARCH models. The authors derive the asymptotic properties of these penalized quasi-likelihood estimators, demonstrating that the use of SCAD and hard threshold penalty functions enables these estimators to satisfy both sparsity and oracle properties. In contrast to the approach by Nielsen and Rahbek (2024), our proposed algorithm imposes fewer restrictions on model parameters. Additionally, the general-to-specific (GETS) approach has been utilized to select mean and volatility specifications for log-ARCHX models in several studies (Bauwens et al., 2012; Sucarrat and Escribano, 2012; Pretis et al., 2018). However, the GETS modeling employed in these references is iterative and computationally more demanding compared to our algorithm. Our algorithm shares similarities with the method described by Zhang (2003) in terms of its stepwise structure. It is non-iterative and applicable to log-TGARCHX models and their nested variants, including log-ARCHX. Nevertheless, it is worth noting that we do not claim that our algorithm is superior or inferior in performance compared to other methodologies present in the literature, and we do not analyze its comparative power with other variable selection methods. Instead, our algorithm offers a novel approach to variable selection in the context of log-TGARCHX models. Interestingly, the models selected by our algorithm have been shown to perform better than benchmark models, namely the unrestricted model with complete set of exogenous variables and

log-GARCH(1,1) model, in predicting one-step-ahead volatility forecasting in the Bitcoin market. Moreover, our algorithm is very flexible in the sense that any variable selection procedure can be employed within its framework. However, in this paper, we use three widely used variable selection methods, namely, LASSO, ABESS, and Boruta procedures separately within the VS-LTGARCHX algorithm to select the best subset of regressors while estimating log-TGARCHX models. We show that enhancing log-TGARCHX models with suitable variable selection procedures generates useful results for conditional volatility modeling in terms of prediction and interpretation. Moreover, the findings of Francq and Sucarrat (2017) suggest that the proposed algorithm can be extended to the multivariate case using an equation-by-equation approach. However, a thorough treatment of multivariate log-TGARCHX modeling is beyond the scope of this paper and will be explored in future research. Second, we apply the proposed VS-LTGARCHX algorithm to time series of Bitcoin, one of the most popular decentralized cryptocurrencies operating on the blockchain (distributed ledger) platform. In fact, Bitcoin has drawn public attention since its inception due to the novel underlying technology that enables the fast transfer of funds internationally without third-party involvement. However, the log-TGARCHX model has not yet been used to study the volatility of Bitcoin, despite the former's higher flexibility compared to many other GARCH family models. We contribute to the existing body of literature by filling the latter gap.

This work has been recently accepted by the Journal of Time Series Econometrics.

1.2.2 Chapter 3: Cross Entropy With Symmetry Breaking and K-Means - A Hybrid Algorithm for Clustering

In this part of the thesis, we focus on HAR class of models and extend them to clusterwise HAR models. We have several contributions to the related literature. First, we contribute to optimization literature by analyzing the deterministic version of the well-known cross-entropy (CE) method for optimization. Specifically, we formally show that, under certain circumstances, the deterministic version of the algorithm does not converge to the global optimum in the presence of multiple optima. To fix this problem, we suggest first suggest to introduce a symmetry breaking condition. Then, using simple examples we demonstrate that even in the presence of the symmetry-breaking constraints the deterministic CE algorithm may not converge to the global optimum if its key parameter - quantile rank parameter denoted by $\rho \in (0, 1)$ is not correctly initialized. Unfortunately,

the correct value of the parameter leading to convergence is unknown *a priori*. To reduce the dependence of the algorithm on ρ we propose to initialize that to a value close to 1, and then, decrease its value geometrically over successive iterations. This modification is denoted as CESB($\rho^{(t)}$). In simulations, we demonstrate that such a modification guarantees the convergence of the deterministic CESB($\rho^{(t)}$) to the global optimum, however, it may increase the number of iterations needed for convergence. As a consequence, we propose to incorporate the popular and computationally fast heuristic k-means clustering algorithm into CESB. This novel hybrid algorithm called CESBKM offers improved convergence and accuracy in solving clustering and clusterwise linear regression problems. We show formally that the deterministic CESBKM algorithm is guaranteed to converge to one of the global optima of k-means clustering and clusterwise linear regression problem in finite number of steps. More importantly, in simulations (finite sample setting) we show that CESBKM algorithm outperforms the generic CE in terms of reaching the global optimum, average number of iterations, and average relative error. Finally, we contribute to literature on HAR models and Bitcoin by extending the HAR model to clusterwise HAR and applying it to Bitcoin returns. Specifically, we how that: i) clusterwise HAR model improves the model fit significantly as compared to the generic HAR when applied to Bitcoin returns; ii) Estimating the clusterwise HAR model using CESBKM algorithm provides much better model fit compared to estimation by the generic cross entropy method, more precisely, the model R^2 increases by 14 percentage points when estimated by the CESBKM algorithm as compared to the generic CE. The problem with the clusterwise HAR model is that the assignment of the returns to clusters (regimes) is completely exogenous. In Chapter 4 of this thesis, we solve this problem by learning the cluster assignment dynamics by non-parametric decision tree algorithm.

1.2.3 Chapter 4: CEDT Algorithm for Estimating Open Loop Threshold Autoregressive Models

Here, we introduce a novel approach to OLTAR model estimation which combines both the Cross-Entropy (CE) method (Rubinstein, 1999) and the Classification And Regression Tree algorithm (CART) introduced by Breiman (1996). We call this new method the CEDT algorithm (for Cross Entropy & Desicion Tree). Our approach addresses two key challenges in OLTAR modeling:

— the estimation of state variables in Open-Loop Threshold Autoregressive (OLTAR)

models, where the threshold function and its parameters are unknown,

— the selection of relevant threshold variables from a potentially large feature space.

The CEDT algorithm leverages the global optimization capabilities of the CE method for state estimation while the decision tree algorithm allows to capture complex variables relationship thanks to its complete non-parametric nature. Moreover, our method can improve interpretability of the threshold mechanism, espacially when the decison trees are simple. Therefore, this combination of CE and CART allows for a more comprehensive exploration of the model space, potentially leading to more accurate and interpretable learning of OLTAR models.

Furthermore, as a second contribution, we propose a novel OLTAR-HAR model for volatility modeling which is the blend of the TAR model and the popular Heterogenous Autoregressive (HAR) models. HAR model is a very simple model which is also very powerful model popular in financial literature to model and forecast the realized volatility (Corsi, 2009). However, simple HAR models may not take into account regime-switching properties of financial markets well. Since the publication of the seminal paper of Corsi (2009), many extensions of the generic HAR model have been proposed in literature. For example, tree-HAR model (Audrino and Corsi, 2010; Corsi et al., 2012) uses Classification and Regression Trees (CART) of Breiman et al. (1984) with the generic HAR to capture long-memory and possible non-linear effects, HAR model of realized volatility with jumps (HAR-RV-J) and with continuous jumps (HAR-RV-CJ) of Andersen et al. (2007) to capture the abrupt price variations in prices of financial assets which have long-term effects on their volatility by decomposing volatility into continuos sample path and jumps, HAR with threshold jumps (HAR-TCJ) of Corsi et al. (2010) proposes threshold jumps, and finally, HAR with signed jumps (HAR-SJ) of Sheppard and Patton (2011) seek to capture the leverage effect by introducing signs of the intra-daily returns into the generic HAR model (for a more detailed survey of HAR models see Corsi et al. (2012) and Wang et al. (2016)). The OLTAR-HAR model we propose here is similar to the tree-HAR model in the sense that both have a HAR model in each regime. The major difference between the tree-HAR model and the OLTAR-HAR model is that OLTAR-HAR model do not need to assume any a priori assumption about the threshold function driving the regime switching mechanism. In that sense, the OLTAR-HAR model is more general than the tree-HAR model.

Chapter 2

A Flexible Variable Selection Algorithm for log-GARCHX Models

2.1 Introduction

The generalized autoregressive conditional heteroskedasticity model (GARCH), introduced by Bollerslev (1986) is a popular class of econometric models to capture the dynamics of conditional variance (volatility) of financial time series. In financial management, volatility of assets is modeled and forecasted through conditional volatility models with the goal of pricing assets, optimizing portfolios, dealing with risk management, etc. In order to allow for more flexible models in this direction, a natural extension of the GARCH model, called GARCHX, allows for the inclusion of exogenous variables to model the conditional variance of time series. The advantage is that additional conditioning variables can significantly improve the model and, therefore, its predictive capabilities (see, for instance, Sucarrat (2021)). However, GARCHX models present some limitations. More precisely, they impose a positivity constraint on the intercept parameter and a non-negativity constraint on all other parameters and all exogenous variables. The impossibility of using relevant covariates in the model clearly limits the widespread applicability of the GARCHX models (Francq and Thieu, 2019; Sucarrat, 2021).

Furthermore, if we are interested in invertibility and stationarity, then additional conditions should be imposed on the autoregressive and moving average parameters of the conditional variance equation (Hamilton, 1994). This restrictive nature of most members of the GARCH family models is relaxed in the EGARCH (Nelson, 1991) and log-GARCH models with exogenous variables (EGARCHX and log-GARCHX, respectively). The log-GARCHX model is indeed the restricted form of log-GARCH with exogenous covariates and threshold variables (log-TGARCHX), as we emphasize herein further. Therefore, the

terms log-GARCHX and log-TGARCHX are sometimes used interchangeably throughout the paper. In the last two members of the GARCH family, fitted volatility is guaranteed to be positive, and non-negativity constraints on parameters and covariates are lifted. However, the advantage of log-GARCHX over EGARCHX is that unconditional variance in the EGARCH formulation may not exist if the errors are too fat-tailed (Sucarrat, 2019). Moreover, probabilistic properties of EGARCH and log-GARCH models have been studied, and it was shown that the consistency and asymptotic normality of the quasi-maximum likelihood estimation (QMLE) of log-GARCH models holds under conditions milder than those of EGARCH. Furthermore, log-GARCH models may capture richer asymmetry dynamics and its shock transmission mechanism may fit the reality better compared to EGARCH and GARCH (Francq et al., 2013).

Arguably, one of the most attractive features of the log-TGARCHX model is that it admits an ARMA representation, as first pointed out by Pantula (1986). This allows for parameter estimation to be carried out using widely available statistical software by employing least squares and quasi-maximum likelihood methods (Francq and Thieu, 2019; Sucarrat, 2019). Moreover, the latter makes the standard asymptotical theory about parameter estimates valid. Under these circumstances, hypothesis testing about the statistical significance of exogenous covariates and ARCH/GARCH parameters is standard. Conversely, because of the respective positivity and non-negativity constraints imposed on parameters in generic GARCH models, the conventional asymptotic theory fails, and non-standard methods should be applied. These questions have been dealt with in detail in Francq and Thieu (2019). Furthermore, Francq and Sucarrat (2017) show that the multivariate log-TGARCHX model can be estimated equation-by-equation using the ARMA representation which is another advantage of log-TGARCHX over the generic GARCHX model.

The contributions of this work are twofold. First, we propose an algorithm called VS-LTGARCHX to incorporate the variable selection procedure into the the univariate log-TGARCHX estimation process. Incidentally, variable selection within the context of ARCH and log-ARCHX models has been extensively studied in literature. For instance, a recent paper by Nielsen and Rahbek (2024) employs a penalized quasi-likelihood approach to estimate and perform model selection for ARCH models. The authors derive the asymptotic properties of these penalized quasi-likelihood estimators, demonstrating that the use of SCAD and hard threshold penalty functions enables these estimators to satisfy both sparsity and oracle properties. In contrast to the approach by Nielsen and Rahbek (2024),

our proposed algorithm imposes fewer restrictions on model parameters. Additionally, the general-to-specific (GETS) approach has been utilized to select mean and volatility specifications for log-ARCHX models in several studies (Bauwens and Sucarrat, 2010; Sucarrat and Escribano, 2012; Pretis et al., 2018). However, the GETS modeling employed in these references is iterative and computationally more demanding compared to our algorithm. Our algorithm shares similarities with the method described by Zhang (2003) in terms of its stepwise structure. It is non-iterative and applicable to log-TGARCHX models and their nested variants, including log-ARCHX. Nevertheless, it is worth noting that we do not claim that our algorithm is superior or inferior in performance compared to other methodologies present in literature, and we do not analyze its comparative power with other variable selection methods. Instead, our algorithm offers a novel approach to variable selection in the context of log-TGARCHX models. Interestingly, the models selected by our algorithm have been shown to perform better than benchmark models, namely the unrestricted model with complete set of exogenous variables and log-GARCH(1,1) model, in predicting one-step-ahead volatility in the Bitcoin market. Moreover, our algorithm is very flexible in the sense that any variable selection procedure can be employed within its framework. However, in this paper, we use three widely used variable selection methods, namely, LASSO, ABESS, and Boruta procedures separately within the VS-LTGARCHX algorithm to select the best subset of regressors while estimating log-TGARCHX models. We show that enhancing log-TGARCHX models with suitable variable selection procedures generates useful results for conditional volatility modeling in terms of prediction and interpretation. Moreover, the findings of Francq and Sucarrat (2017) suggest that the proposed algorithm can be extended to the multivariate case using an equation-by-equation approach. However, a thorough treatment of multivariate log-TGARCHX modeling is beyond the scope of this paper and will be explored in future research [1]. Second, we apply the proposed VS-LTGARCHX algorithm to time series of Bitcoin (BTC) prices, one of the most popular decentralized cryptocurrencies operating on the blockchain (distributed ledger) platform. In fact, BTC has drawn public attention since its inception due to the novel underlying technology that enables the fast transfer of funds internationally without third-party involvement. However, the log-TGARCHX model has not yet been used to study the volatility of BTC prices, despite the former's higher flexibility compared to many other GARCH family models. We contribute to the existing body of literature by filling the latter gap.

1. We thank an anonymous referee for this important observation.

The paper is structured as follows. In Section 2.2, we describe and contrast the GARCHX and log-TGARCHX models and emphasize the advantages of the latter over the former in terms of the inclusion of exogenous variables and model estimation. In Section 2.3, we present a novel procedure for selecting the subset of variables from a much larger set of variables to be used within the log-TGARCHX model. In Section 2.4, we apply our procedure to the BTC market and show that, indeed, the one-step-ahead volatility predictions can be improved using the suggested procedure, and in Section 2.5, we conclude.

2.2 On Generalized Autoregressive Conditional Heteroskedasticity based Models

This section introduces briefly the *generalized autoregressive conditional heteroskedasticity with exogenous variables model* (GARCHX) and the *logarithmic threshold generalized autoregressive conditional heteroskedasticity with exogenous variables model* (log-TGARCHX), and we discuss why the latter may be superior to the former, particularly in the presence of multiple exogenous variables. The last subsection provides the ARMA representation of log-TGARCHX models, which helps to simplify the parameter estimation and inference procedures.

2.2.1 The GARCHX Model

Assume a covariance stationary process given by

$$y_t = \mu_t + \epsilon_t, \quad \epsilon_t = \sigma_t \eta_t, \quad \sigma_t > 0, \quad \eta_t \sim iid\,(0,1)\,, \tag{2.1}$$

where $\{y_t\}_{t\geq 0}$ is the time series process (e.g. financial return), $\mu_t = \mathbb{E}(y_t \mid \mathcal{I}_t)$ is the conditional mean with a functional form known *a priori* but whose parameters need to be estimated, $\sigma_t^2 = Var(y_t \mid \mathcal{I}_t)$ is the conditional variance of y_t, $\mathcal{I}_t$ is the conditioning information set at time t and the error term ϵ_t is a random variable with mean zero and variance σ_t^2. The term η_t is an independently and identically distributed innovation process with zero mean and unit variance, as indicated by $iid\,(0,1)$. Finally, the variables η_t and σ_t are assumed to be independent.

The conditional variance equation of GARCHX(p_1,p_2,p_3) model is defined by

$$\begin{aligned}\sigma_t^2 =&\alpha_0 + \sum_{i=1}^{p_1} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p_2} \alpha_{p_1+j} \sigma_{t-j}^2 + \sum_{k=1}^{p_3} \beta_k x_{k,t}, \\ &\alpha_0 > 0, \quad \alpha_i \geq 0 \quad \forall i \in \{1, \dots, p_1 + p_2\}, \\ &\beta_k \geq 0 \text{ and } x_{k,t} \geq 0 \quad \forall k = \{1, \dots, p_3\},\end{aligned} \tag{2.2}$$

where α_0 is the volatility intercept and $p_1 \in \mathbb{N}$ and $p_2 \in \mathbb{N}$ are ARCH (squared lags of ϵ_t) and GARCH (squared lags of σ_t) orders respectively. Note that if $p_2 = 0$, the GARCH(p_1,p_2) model is the ARCH(p_1) model. Moreover, $\{x_{k,t}\}_{k=1}^{p_3}$ are p_3 stationary exogenous covariates and $\{\beta_k\}_{k=1}^{p_3}$ are the unknown associated parameters to estimate. Note that σ_t is independent of η_t, although it depends on past values $\eta_{t-1}, ..., \eta_{t-p_1}$. The inclusion of exogenous covariates into the GARCH equation comes with the additional cost of non-negativity restrictions on both the exogenous variables $\{x_{k,t}\}_{k=1}^{p_3}$ and their associated parameters $\{\beta_k\}_{k=1}^{p_3}$ in order to guarantee the positivity of the conditional variance. Furthermore, inference procedures under nullity are non-standard (Francq and Thieu, 2019; Sucarrat, 2021) and there is a need for alternative asymptotics, as in Pedersen and Rahbek (2019). This restrictive nature of GARCHX models is relaxed a great deal by log-TGARCHX models, which we demonstrate in the next subsection.

2.2.2 The Log-TGARCHX model

The log-TGARCHX model (Sucarrat, 2019) assumes logarithmic specification of the conditional variance process and includes both asymmetry terms and stationary covariates. Formally, the conditional variance equation for the log-TGARCHX(p_1,p_2,p_3,p_4) is defined as follows:

$$\begin{aligned}\ln \sigma_t^2 = \alpha_0 + \sum_{i=1}^{p_1} \alpha_i \ln \epsilon_{t-i}^2 + \sum_{j=1}^{p_2} \alpha_{p_1+j} \ln \sigma_{t-j}^2 + \sum_{k=1}^{p_3} \beta_k x_{k,t} + \\ + \sum_{l=1}^{p_4} \gamma_l \mathbb{I}_{\{\epsilon_{t-l}<0\}} \ln \epsilon_{t-l}^2, \quad \alpha_i, \beta_k, \gamma_l \in \mathbb{R}, \\ \forall i \in \{0, \dots, p_1 + p_2\}, \quad \forall k \in \{1, \dots, p_3\}, \quad \forall l \in \{1, \dots, p_4\},\end{aligned} \tag{2.3}$$

where α_0 is log-volatility intercept which controls the level of volatility in a multiplicative way, $(\alpha_i)_{i=\{1,...,p_1\}}$ and $(\alpha_{p_1+j})_{j=\{1,...,p_2\}}$ are real scalar sequences called ARCH-parameters and GARCH-parameters, respectively (Sucarrat, 2019). Note that, when $p_3 = p_4 = 0$, the

log-GARCHX(p_1,p_2,p_3,p_4) model boils down to the log-GARCH(p_1, p_2) model introduced by Pantula (1986), Geweke (1986) and Milhoj (1987), independently.

The log-GARCHX model is referred to as the asymmetric log-GARCHX model in the literature (Sucarrat, 2019). However, we suggest calling the latter log-TGARCHX because it is equivalent to the TGARCHX model of Glosten et al. (1993) except for the logarithmic transformation.

2.2.3 ARMA Representation of Log-TGARCHX Model

The log-GARCH family of volatility models has several attractive features. One of them is the representability via non-linear ARMA process, which makes those models amenable to estimation using ARMA inference methods and almost with any statistical software. Due to the latter fact, the standard inference procedures remain applicable. Furthermore, the addition of exogenous, deterministic and / or predetermined conditioning variables does not change the relationship between the ARMA coefficients and the log-GARCH coefficients under suitable conditions (Sucarrat et al., 2016; Sucarrat, 2019). More precisely, if $\mathbb{P}(\eta_t = 0) = 0$ and $\mathbb{E}|\ln \eta_t^2| < \infty$ for η_t defined in (2.1), then log-TGARCHX accepts the ARMAX representation (Sucarrat, 2019) as follows:

$$\begin{aligned}\ln {\epsilon_t}^2 = \phi_0 + \sum_{i=1}^{p_1} \phi_i \ln \epsilon_{t-i}^2 + \sum_{j=1}^{p_2} \phi_{p_1+j} u_{t-j} + \sum_{k=1}^{p_3} \beta_k x_{k,t-1} \\ + \sum_{l=1}^{p_4} \gamma_l \mathbb{I}_{\{\epsilon_{t-l}<0\}} \ln \epsilon_{t-l}^2 + u_t,\ u_t \sim iid(0, \sigma_u^2).\end{aligned} \tag{2.4}$$

After estimating the parameters of the ARMAX representation, we employ the following mappings (for a detailed exposition, refer to Sucarrat (2019)) to obtain parameter estimates for the log-TGARCHX model stated in (2.3):

$$\begin{aligned} u_t &= \ln \eta_t^2 - \mathbb{E}\left(\ln \eta_t^2\right) \\ \phi_i &= \begin{cases} \alpha_0 + \left(1 - \sum_{j=1}^{p_2} \alpha_{p_1+j}\right) \times \mathbb{E}\left(\ln \eta_t^2\right), & i = 0 \\ \alpha_i + \alpha_{p_1+i}, & 1 \le i \le p_1 \\ -\alpha_i, & p_1 + 1 \le i \le p_2. \end{cases} \end{aligned} \tag{2.5}$$

The values of parameters $(\beta_l)_{l=\{1,\ldots,p_3\}}$ for the exogenous variables are the same both in log-GARCHX specification and in the corresponding ARMAX representation.

Although ARCH and GARCH models can also be estimated using AR and ARMA representation, respectively, with least squares, this approach is less attractive compared to log-TGARCHX models due to the reasons mentioned in Sucarrat and Escribano (2012, p. 2) and in Francq and Zakoïan (2010, p.162).

The possibility of the ARMAX representation paves the way for a wide range of extensions of the log-GARCH-family models. In this paper, we propose enhancing log-GARCHX models by including a variable selection procedure during model estimation. Nowadays, there exist plenty of methods for performing variable selection. In this work, we focus on three methods but other ones can be easily adapted to the proposed framework. The following section describes our original methodology and the three variable selection methods.

2.3 Enhancing log-TGARCHX Model by Adding Variable Selection Algorithms

2.3.1 Variable Selection Procedure for log-TGARCHX model (VS-LTGARCHX)

As described by Sucarrat (2019), the log-TGARCHX model is estimated in two steps. The first step is to form the ARMAX (p_1, p_2, p_3, p_4) representation of the model as given in (2.4) and estimate the coefficients using the ordinary least squares (OLS) method. The second step consists of mapping the estimated coefficients to the coefficients of the model of interest, namely, log-TGARCHX specified in (2.3) using the maps in (2.5).

As an important contribution of this paper, we show that the log-TGARCHX model can be improved by selecting a relevant subset of exogenous variables and asymmetry lags, denoted by $K_\star \subseteq \{1, \ldots, p_3\}$ and $L_\star \subseteq \{1, \ldots, p_4\}$, respectively. To that end, we propose a heuristic strategy that embeds variable selection within the log-TGARCHX estimation procedure. More precisely, this approach describes i) how to select both the subset of exogenous variables, through the $L_\star$ index set, and that of the asymmetry terms through $K_\star$ index set to be used in the log-TGARCHX model, and then ii) how to estimate parameters of the model. This new method that is made of three steps is described below.

1. *Step 1. Fit an ARMA representation and get the residuals*
   First, we fit the following ARMA(p_1,p_2) model:

$$\ln {\epsilon_t}^2 = \phi_0 + \sum_{i=1}^{p_1} \phi_i \ln \epsilon_{t-i}^2 + \sum_{j=1}^{p_2} \phi_{p_1+j} z_{t-j} + z_t. \tag{2.6}$$

Then, we compute the model residuals which are defined as:

$$\hat{z}_t = \ln {\epsilon_t}^2 - \widehat{\ln {\epsilon_t}^2}, \tag{2.7}$$

where $\widehat{\ln {\epsilon_t}^2}$ denotes the fitted (predicted) value of $\ln {\epsilon_t}^2$ from the regression model in (2.6). We use the `arima` function in base R (R Core Team, 2023) to estimate the parameters in (2.6). According to the documentation of the package, the default estimation process is carried out based on the procedure depicted in Gardner et al. (1980).

2. *Step 2. Perform the variable selection procedure for both exogenous variables and asymmetry orders*
   Next, variable selection is applied on the following unrestricted model:

$$\hat{z}_t = a_0 + \sum_{k=1}^{p_3} b_k \mathbb{I}_{\{\epsilon_{t-k}<0\}} \ln \epsilon_{t-k}^2 + \sum_{l=1}^{p_4} c_l x_{l,t-1} + e_t, \tag{2.8}$$

   where $a_0, \{b_k\}_{k=1}^{p_3}, \{c_l\}_{l=1}^{p_4}$ are the model parameters to be estimated, $\mathbb{I}_{\{\epsilon_{t-k}<0\}} \ln \epsilon_{t-k}^2$ is the k-th lagged asymmetry term evaluated at time t, $x_{l,t-1}$ is the l-th stationary exogenous covariate evaluated at time t and e_t is the model error at time t. In this paper, we made use of three different variable selection algorithms shown in subsection 2.3.2 just for illustration purposes. However, any other variable selection method can be legitimately used in this step (*Step 2*) of our algorithm. As a result of the variable selection process, we obtain the index sets $L_\star$ and $K_\star$ of size $p_{3\star}$ and $p_{4\star}$, respectively. To perform variable selection in practice for the purposes of *Step 2* we make use of `glmnet` (Friedman et al., 2010), `abess` (Zhu et al., 2022) and `Boruta` (Kursa and Rudnicki, 2010) libraries, respectively.

3. *Step 3. Fit the final log-TGARCHX Model*
   Finally, we can estimate the log-TGARCHX(p_1, p_2, $p_{3\star}$, $p_{4\star}$) model:

$$\begin{aligned} \ln \sigma_t^2 = \alpha_0 + \sum_{i=1}^{p_1} \alpha_i \ln \epsilon_{t-i}^2 + \sum_{j=1}^{p_2} \alpha_{p_1+j} \ln \sigma_{t-j}^2 + \sum_{k \in K^*} \beta_k x_{k,t} \\ + \sum_{l \in L^*} \gamma_l \mathbb{I}_{\{\epsilon_{t-l}<0\}} \ln \epsilon_{t-l}^2. \end{aligned} \tag{2.9}$$

The parameters $\{\alpha_i\}_{i=1}^{p_1+p_2}, \{\beta_k\}_{k\in K_\star}, \{\gamma_l\}_{l\in L_\star}$ of the final model can be estimated using the quasi-maximum likelihood approach (Francq et al., 2013) or alternatively, using ARMA-based methods (Sucarrat and Escribano, 2018).

ARMA-based estimation of log-TGARCHX models means setting up and estimating the parameters of the ARMAX representation of the corresponding log-TGARCHX model using either least squares method (Sucarrat et al., 2016) or quasi-maximum likelihood method with exponential χ^2 as an instrumental density (Francq and Sucarrat, 2013). To carry out parameter estimations for *Step 3* we use `lgarch` package (Sucarrat, 2015) throughout our analysis and use ARMA-based least squares estimator (although ARMA-based quasi maximum likelihood is also available in the package). The procedure sets up an ARMAX representation of the corresponding log-TGARCHX model as given in (2.4), estimates the parameters of the corresponding ARMAX model using least squares method and then, maps parameter estimates of ARMAX model to those of the underlying log-TGARCHX model using (2.5).

The motivation behind the three-step procedure described above is rooted in the selective regularization of the parameters of the ARMAX representation in (2.4). The goal is to regularize all parameters except the ARMA coefficients. This approach is crucial because if the ARMA terms are also regularized, there is a risk that they might all be simultaneously reduced to zero. In such a scenario, the regularized ARMAX representation of the log-TGARCHX model would become meaningless. Therefore, to ensure the meaningful estimation of the log-TGARCHX model coefficients, it is essential to keep the ARMA parameters non-regularized. Mathematically, the objective is to solve the minimization problem:

$$\begin{aligned}\min \sum_{t=1}^{T} \Bigg(& \ln {\epsilon_t}^2 - \phi_0 - \sum_{i=1}^{p_1} \phi_i \ln {\epsilon_{t-i}}^2 - \sum_{j=1}^{p_2} \phi_{p_1+j} u_{t-j} \\ & - \sum_{k=1}^{p_3} \beta_k x_{k,t-1} - \sum_{l=1}^{p_4} \gamma_l \mathbb{I}_{\{\epsilon_{t-l}<0\}} \ln {\epsilon_{t-l}}^2 \Bigg)^2 + \mathcal{P}(\beta, \gamma; \lambda),\end{aligned} \tag{2.10}$$

where $\mathcal{P}(\beta, \gamma; \lambda)$ represents a penalty function applied to the non-ARMA parameters (β, γ), with λ as the regularization parameter.

A complex issue in the latter minimization problem is the inclusion of latent moving average (MA) terms (u_{t-j}). Typically, least squares or maximum likelihood alongside iterative methods such as the innovation algorithm are used to estimate the parameters

of the ARMA model and the latent MA series simultaneously (for a complete discussion on the estimation of the ARMA model, see Gardner et al. (1980); Brockwell and Davis (1991); Hamilton (1994)). The latent nature of MA terms adds complexity to the optimization process in (2.10). Notably, for estimating log-ARCHX models, the MA component of the ARMAX representation is omitted, leaving an ARX representation. In such instances, regularizing (penalizing) the ARX representation can be efficiently achieved in one step which corresponds to solving (2.10) with MA terms omitted. This means that our novel methodology can be easily adapted to penalized estimation of log-ARCHX and log-ARCH models (for penalized likelihood estimation of ARCH models see Nielsen and Rahbek (2024)). [2] Comparing the performance of our novel methodology to that of the penalized likelihood method in the context of ARCH models would be intriguing, though it is beyond the scope of this study. We demonstrate in the following Proposition 2.1 that the estimators obtained from penalizing the regression model in (2.8) (refer to *Step 2* of the three-step method) are asymptotically equivalent to those obtained from minimizing the objective function in (2.10), given certain regularity conditions.

Proposition 2.1. *Let us assume the following orthogonality conditions hold:*

1. $Cov(\ln \epsilon_{t-i}^2, x_{k,t-1}) = 0, \quad \forall i = 1, \dots, p_1, \quad \forall k = 1, \dots, p_3,$
2. $Cov(\ln \epsilon_{t-i}^2, \sum_{l=1}^{p_4} \gamma_l \mathbb{I}_{\{\epsilon_{t-l}<0\}} \ln \epsilon_{t-l}^2) = 0, \quad \forall i = 1, \dots, p_1, \quad \forall l = 1, \dots, p_4,$
3. $Cov(u_{t-j}, x_{k,t-1}) = 0, \quad \forall j = 1, \dots, p_2, \quad \forall k = 1, \dots, p_3,$
4. $Cov(u_{t-j}, \mathbb{I}_{\{\epsilon_{t-l}<0\}} \ln \epsilon_{t-l}^2) = 0, \quad \forall j = 1, \dots, p_2, \quad \forall l = 1, \dots, p_4.$

Suppose further that the (true) ARMA process has p_1 AR lags and p_1+p_2 MA lags, and all variables of ARMA model have finite variance. Then, the estimators obtained from solving (2.10) and from running penalized regression on (2.8) are asymptotically equivalent i.e., they converge to the same value in probability.

Proposition 2.1, whose proof can be found in Proof 6.1.1, asserts that the penalized regression problem in (2.10) to select variables in ARMAX model can be performed in an equivalent but simpler way under suitable conditions. After variable selection is done (*Step 2*) the final log-TGARCHX model is estimated. The consistency and asymptotic normality of the ARMA-based least squares estimators of the log-TGARCHX model are demonstrated in Sucarrat et al. (2016).

Ultimately, Proposition 2.1 sets the statistical foundation of the three-step methodology in the context of penalization-based variable selection methods, provided that certain

2. We thank an anonymous referee for the latter observation and reference.

regularity conditions are met. Nonetheless, other types of variable selection methods may also be incorporated into the three-step procedure on a heuristic basis. For instance, Boruta, one of the three variable selection methods we use in this paper, is not based on penalization.

In the next subsection, we shed light on the second step of the proposed methodology.

2.3.2 Variable selection algorithms

In this section, three variable selection methods are introduced to select $L_\star$ and $K_\star$ (see *Step 2* in Section 2.3.1). However, any other variable selection method can be incorporated into the VS-LTGARCHX algorithm, which makes it very flexible. Moreover, as described in the previous section, the variable selection procedure is performed on the linear model (2.8). The subset selection procedure is important, especially when the set of potential regressors is large or when the regressors are highly correlated. For instance, when the number of regressors is larger than the number of observations, it is well-known that linear models may suffer from high variance, which is referred to as over-fitting in literature. Furthermore, parsimonious models are more easily interpretable than complex models with a lot of regressors. Therefore, the selection of variables to identify the most influential variables can improve both predictive accuracy and interpretability (see Miller (2002) for more details). In fact, there exist many variable selection methods in the related literature, and any of them can be incorporated into our novel algorithm. Here, we have chosen three of them just for illustration purposes: the least absolute shrinkage and selection operator (LASSO), the adaptive best subset selection (ABESS), and Boruta.

Variable selection using LASSO

LASSO (*least absolute shrinkage and selection operator*) is a regression method introduced by Tibshirani (1996). The method is still widely used to perform both variable selection and regularization. This is achieved by adding a penalty function of the model parameters. Consider the linear regression setting in (2.8). Then, the LASSO regression problem consists of estimating the model parameter vector $\Theta = \{a_0, \{b_k\}_{k=1}^{p_3}, \{c_l\}_{l=1}^{p_4}\}$ that

solves the following constrained OLS problem:

$$\widehat{\Theta}^{\text{lasso}} = \underset{\Theta\in\mathbb{R}^{(p_3+p_4+1)}}{\arg\min} \Big(\sum_{t=1}^{T} \Big(z_t - a_0 - \sum_{k=1}^{p_3} b_k \mathbb{I}_{\{\epsilon_{t-k}<0\}} \ln \epsilon_{t-k}^2 - \sum_{l=1}^{p_4} c_l x_{l,t-1} \Big)^2 + \\ + \lambda \Big(\sum_{k=1}^{p_3} |b_k| + \sum_{l=1}^{p_4} |c_l| \Big) \Big), \tag{2.11}$$

where the objective function is nothing else than the sum of mean squared errors and $\lambda > 0$ is a regularization parameter. LASSO regression not only shrinks the magnitude of all the model coefficients, but also sets some of them to zero. Then, in *Step 2* we use LASSO to perform the variable selection in (2.8). The subsets of indices $L_\star$ and $K_\star$ obtained using the LASSO are denoted by

$$\begin{aligned} K_\star^{\text{lasso}} &= \{k | \hat{b}_k^{\text{lasso}} \neq 0, \quad \forall k = 1, \dots, p_3\} \\ L_\star^{\text{lasso}} &= \{l | \hat{c}_l^{\text{lasso}} \neq 0, \quad \forall l = 1, \dots, p_4\} \end{aligned} \tag{2.12}$$

corresponds to the sets of indices of non-zero parameters. Note that the value of the regularization parameter λ is usually unknown and therefore needs to be tuned. Several approaches have been suggested to adjust λ. Here, we adopt the algorithm outlined in Section 2.5 of Friedman et al. (2010). The algorithm finds the minimum Lagrange multiplier value in the Lagrangian representation of the constrained optimization problem in the (2.11) which shrinks all slope coefficients to zero simultaneously, and that value is denoted by λ_{max}. Then λ_{min} is set to $0.001\lambda_{max}$ and the grid is formed as a decreasing sequence of 100 values from λ_{max} to λ_{min} on the log scale.

Adaptive best subset selection (ABESS) algorithm

ABESS is a recently developed polynomial algorithm due to Zhu et al. (2020) for the best subset selection. In simple linear regression setting (e.g. (2.8)), this algorithm chooses the subset of covariates by solving the following minimization problem:

$$\widehat{\Theta}^{\text{abess}} = \underset{\Theta\in\mathbb{R}^{(p_3+p_4+1)}}{\arg\min} \sum_{t=1}^{T} \Big(z_t - a_0 - \sum_{k=1}^{p_3} b_k \mathbb{I}_{\{\epsilon_{t-k}<0\}} \ln \epsilon_{t-k}^2 - \sum_{l=1}^{p_4} c_l x_{l,t-1} \Big)^2 \\ \text{subject to} \quad \Big(\sum_{k=1}^{p_3} ||b_k||_0 + \sum_{l=1}^{p_4} ||c_l||_0 \Big) \leq s, \tag{2.13}$$

where $s \in \{1, \dots, p_3 + p4\}$ is a regularization parameter and $||\cdot||_0$ is the ℓ_0 norm.

Here, the Lagrangian of the problem is not continuous, and its solution may be computationally infeasible with large p. However, a recently developed splicing algorithm and special information criterion (SIC) have been proven to solve the minimization problem above in polynomial times and for an unknown sparsity parameter (Zhu et al., 2020).

Although the ABESS algorithm is extremely fast compared to its competitors as shown in the latter reference, it seems to be more aggressive compared to LASSO in penalization as the form of the constraint function suggests. Therefore, when the set of the covariates upon which variable selection needs to be performed is relatively small, LASSO should be preferred. However, a comparison of the two is outside the scope of our study and needs further research.

Finally, one of the main advantages of ABESS is that the potential values of the tuning parameter s are finite, as the l_0 norm penalty suggests. The selected variables by ABESS algorithm are the ones which receive non-zero coefficients after the minimization of the latter equation. More precisely,

$$\begin{aligned} K_\star^{\text{abess}} &= \{k|\hat{b}_k^{\text{abess}} \neq 0, \quad k = 1, \dots, p_3\} \\ L_\star^{\text{abess}} &= \{l|\hat{c}_l^{\text{abess}} \neq 0, \quad l = 1, \dots, p_4\}. \end{aligned} \tag{2.14}$$

Boruta Algorithm

Boruta is an iterative feature selection algorithm based on the random forest algorithm (Kursa and Rudnicki, 2010). At each iteration, the algorithm creates random shuffle copies of the original features. Then, a random forest is trained on the extended data that included both the original features and their respective shuffled copies. To measure the importance of each feature, the algorithm computes a Z score on each feature and compares it to a threshold defined as the maximum Z score of all shuffle copies. Then, only variables for which importance is significantly higher than the threshold are kept in the model while the others are removed. This process is repeated until either all features are labeled as important/non-important or until the maximum number of iterations is reached (Kursa and Rudnicki, 2010).

Ultimately, applying Boruta algorithm on (2.8) results in two sets, $K_\star^{boruta}$ and $L_\star^{boruta}$, which correspond to the sets of indices of the retained exogeneous variables and asymmetry lags.

2.4 Application to BTC market

In this section, we apply the VS-LTGARCHX algorithm to BTC market [3] and study its performances in comparison with the log-GARCH(1,1) model and the full-blown log-TGARCHX model. The log-GARCH(1,1) model is a log-GARCH model with no exogenous variable while the full-blown log-TGARCHX model is a log-GARCH model with the complete set of exogenous variables and asymmetry terms. Those two *extremes* form the benchmark models to compete with. The next subsection describes the BTC data used.

2.4.1 Data: data sources and data pre-processing

The sample period runs from December 18, 2017 to June 17, 2022 and includes 1643 observations in total collected at daily frequency. It is worth mentioning that the BTC price data were collected in 5-minute and daily frequencies, which were taken from Binance exchange using a Python library due to McHardy (2023). The BTC price data at the daily frequency were used to obtain the BTC daily returns. Moreover, BTC prices at 5-minute frequency were used to obtain the daily realized variance as described in (Bergsli et al., 2022). Formally, let R_t, the return at the t-th day, be defined as

$$R_t = \ln P_t - \ln P_{t-1}, \tag{2.15}$$

where P_t is the close price of BTC on day t. Then, we define m equally spaced intraday returns between each consecutive two days t and $t+1$. More precisely, the j-th intraday return for the the t-th day is denoted by $R_{t+\frac{j}{m}}$ and defined as:

$$R_{t+\frac{j}{m}} = \ln\left(P_{t+\frac{j}{m}}\right) - \ln\left(P_{t+\frac{j-1}{m}}\right), \quad \text{for } j = 0, \ldots, m-1, \tag{2.16}$$

where $\ln\left(P_{t+\frac{j}{m}}\right)$ (resp. $\ln\left(P_{t+\frac{j-1}{m}}\right)$ is the logarithm of the intraday price of BTC at time $t+\frac{j}{m}$ (resp. $t+\frac{j-1}{m}$). Thus, the daily realized variance also called volatility, for the t-th day can be defined as the sum of squared intraday returns following the previous reference:

$$V_t^{(d)} = \sum_{j=0}^{m-1} R^2_{t+\frac{j}{m}}. \tag{2.17}$$

3. Supplementary material, code, and data can be found in https://github.com/salahaddiniayyubi/Log-TGARCHX-Subset-Selection

where the subscript d is used to emphasize the daily frequency of the realized variance.

For the purposes of this paper, we used BTC prices in 5-minute frequency to estimate the daily realized variance (volatility), which implies that m in (2.16) and (2.17) is set to 288 throughout our analysis.

We classify exogenous variables into groups using a classification similar to Chen et al. (2021), namely, Blockchain info, Public Opinion, Risks and Uncertainties, Financials and Macroeconomic development. However, note that the number of variables we use in this work is much more than in Chen et al. (2021). To be more precise, we consider 39 exogenous variables, 3 variables which are transformations of high-frequency returns (daily, weekly and monthly realized volatilites) and a threshold variable, which constitute 42 variables in total. The threshold variable is nothing but a binary variable which receives the value of unity if the previous day return is negative and the value of zero if otherwise. The complete list of exogenous variables is given in Appendix 6.1.2. All variables were collected at a daily frequency. However, unlike BTC prices, measurements on most exogenous variables are missing on weekends and public holidays. Then, to overcome inconsistency in frequency, the last observation carried forward (LOCF) method was used to fill the missing values.

In addition, we used only stationary exogenous variables in our study as shown in Sucarrat (2019). Therefore, Augmented Dickey-Fuller (ADF) test due to Dickey and Fuller (1979) was applied to each exogenous variable to identify non-stationary ones. The non-stationary variables that take only positive values were exposed to the logarithmic difference transformation ($d\ln(X_t) = \ln X_t - \ln X_{t-1}$) and the prefix $dl\_$ was added to their names. In contrast, non-stationary variables with non-positive and positive values were subject to first differencing ($X_t \mapsto X_t - X_{t-1}$), and the prefix $d\_$ was added to their names. Furthermore, we applied stationarity test (Kwiatkowski et al., 1992) to the data to confirm the results of the ADF test which is a standard practice (Brooks, 2008, p. 331). If the stationarity hypothesis is rejected by KPSS those series were subject to the first differencing and the prefix $d\_$ was added to their names. In addition, all covariates were standardized to have zero mean and unit variance.

Incidentally, data downloading and pre-processing has been undertaken using Python (Van Rossum and Drake, 2009), while the rest of the statistical analysis was performed using R (R Core Team, 2023).

Finally, in the next subsection, we will describe in detail how the VS-LTGARCHX algorithm is applied to BTC market data and how the performance of different variants

of the algorithm is evaluated.

2.4.2 Model Performance and Comparison to Benchmark

Once the data preprocessing performed, we divide the dataset into train, test, and validation set. Next, the hyperparameters of the variable selection algorithms are tuned. Then, the final models are built and their performances are measured. The following subsections describe the methodology we performed to achieve these steps.

Data Split

First, the time series data set was divided into train, validation, and test sets. More precisely, the data partitioning consisted of dividing the sample period (from December 18, 2017 to June 17, 2022) into three sub-periods corresponding, respectively, to the train, validation and test sets.

The train set is used to build a prediction model (learner) with a fixed set of hyperparameters. The validation set is used to select the best combination of hyperparameters for the respective variable selection method based on some performance metrics. Ultimately, the test set is used to evaluate the performance of the final model, the one fitted with the best combination of hyperparameters (Hastie et al., 2001).

Moreover, the benchmark models considered in this paper do not have any hyperparameters to tune. Therefore, the validation set is not needed, so it was merged with the train set when dealing with the benchmark models.

Finally, to respect the dynamic structure of the data, we use the recursive and rolling window scheme while partitioning the data set for all models considered (Elliott and Timmermann, 2008).

Hyper-Parameter Tuning

As has already been mentioned, the three variable selection methods, namely LASSO, ABESS and Boruta, have hyperparameters that need to be tuned. To do that, we used two strategies : one for LASSO and ABESS and another one for Boruta. Indeed, a more efficient strategy can be used for Boruta, since it is a wrapper around the random forest algorithm (Breiman, 1996) that is heavily based on bootstrap aggregation (bagging).

To tune the hyperparameters of LASSO and ABESS, first, we fit prediction models (named learners) to the training set by fixing the hyperparameter value based on a pre-

defined grid. Then, for each hyperparameter value, we obtain the prediction errors of the learners in the validation set. The value of the tuning parameter associated with the smallest validation root mean squared error (RMSE) is considered optimal.

The hyperparameter tuning procedure for the random forest behind the Boruta wrapper can be done more efficiently due to the bootstrap resampling applied to the initial training set (Breiman, 2001). However, the use of the original bootstrap strategy would destroy the underlying temporal dependence in time series. Therefore, we utilize the moving block bootstrap mechanism to respect the dynamic nature of the data while conducting the bagging process. The idea is to divide the initial training set into N small blocks such that the temporal structure is preserved within each block, and then to perform a bootstrapping using the N blocks (see Kunsch (1989); Y (1992)). Unlike LASSO and ABESS which have only one hyperparameter to be tuned each, in Boruta there are multiples of them, which makes it costly to train the latter. Finally, we tune four hyperparameters, three of them are generic to Boruta, namely, the number of trees, the minimum node size, the number of splitting variables, and one is due to the moving block bootstrap algorithm, namely, the block size (see Algo 6.2). Again, the set of hyperparameters associated with the smallest RMSE is deemed optimal.

Evaluation of The Predictive Performance and Comparison to The Benchmark Models

Once the optimal values of the hyperparameters are found, the variable selection is carried out by combining the training and the validation sets, and this is the end of *Step 2* of the VS-LTGARCHX algorithm. The final step, *Step 3* consists of fitting the log-TGARCHX model as explained in detail in Escribano and Sucarrat (2018); Sucarrat (2019) with the variables selected by the respective variable selection algorithms.

Moreover, we use the same portion of the time series to build the final model for both the benchmarks and the VS-LTGARCHX variants and use the same strategy for all models considered to generate one-step-ahead predictions. The predictions are based on log-TGARCHX(p_1, p_2, $p_{3\star}$, $p_{4\star}$) model represented in (2.9) which corresponds to *Step 3* of VS-LTGARCHX (see Appendix 6.1.5). More precisely, to measure the performance of each model, we generate one-step-ahead predictions from the test set (the same portion of the time series for all models for a fair comparison) using the rolling window scheme, and finally calculate six different conventional metrics for each model based on their respective forecasts. The procedure is outlined in Algo 6.3.

Remark 2.1. *The rolling window scheme is used instead of the recursive window scheme because it is known that the recursive window scheme may pose problems when evaluating nested models (Violante and Laurent, 2012, p. 466) and that the stationarity assumption of relative losses underlying the Model Confidence Set (MCS) test may not hold when using the expanding window scheme (Hansen et al., 2011).*

Remark 2.2. *We use the square root of daily realized volatility shown in (2.17) as a proxy for the true conditional standard deviation (σ_t) when calculating the performance metrics in accordance with the related literature (Violante and Laurent, 2012).*

As for the judgment about the predictive superiority of models relying solely on performance metrics may not be convincing due to underlying sampling issues, we use a formal hypothesis testing procedure, namely MCS due to Hansen et al. (2011), to make inferences about the ranking of the models. Moreover, MCS procedure allows different aspects of the models to be evaluated by using user-specified loss functions (Bernardi and Catania, 2015). In this study, we mainly use two loss functions, namely, RMSE and QLIKE within the MCS procedure due to their robustness (Patton, 2011). Other metrics such as mean error (ME), mean absolute error (MAE), mean absolute percentage error (MAPE) and mean percentage error (MPE) are also reported in Appendix 6.1.6.

The strategy used to assess the performance of the VS-LTGARCHX algorithm compared to the benchmark models is summarized in Algo 6.3 displayed in Appendix 6.1.5.

2.4.3 Experimental Results

Data preprocessing, data partition and parameters tuning

We collect daily data on the BTC market and 43 variables measured at daily frequency covering the period from December 18, 2017 to June 17, 2022. The missing values if any were filled using the last observation carried forward principle. The complete list of the variables is given in Table 6.1, Table 6.2 and Table 6.3 of Appendix 6.1.2. After the data are pre-processed and all variables are stationarized, and standardized to have zero mean and unit variance, we partition the dataset into train, validation and test sets in 60%-20%-20% chunks, initially (see Algo 6.1).

In the next subsections, we will shed light on the set of variables selected by LASSO, ABESS, and Boruta, respectively, and then compare the VS-LTGARCHX variants' predictive accuracy to those of the benchmark models.

Selected Variables by VS-LTGARCHX Algorithms

Before performing variable selection, the variable selection algorithms, namely LASSO, ABESS and Boruta, need to be calibrated. We followed the strategies introduced in Section 2.4.2. The hyperparameter tuning process for LASSO and ABESS is outlined in Algo 6.1. The tuning process for RF underlying the Boruta wrapper is summarized in Algo 6.2. The optimal values of the tuning parameters help us to perform the variable selection for the respective variable selection method. The variables selected by various selection procedures are given in Table 2.1.

	ABESS	LASSO	Boruta
1	d_VIX	dl_MKTCP	dl_CPTRA
2	d_gtrends	d_VIX	dl_ETRVU
3	rv_w	d_gtrends	dl_HRATE
4		rv_w	dl_MWNTD
5			dl_MWTRV
6			dl_NADDU
7			dl_NTRAN
8			dl_NTRBL
9			dl_NTREP
10			dl_SPY
11			dl_DJI
12			d_CPTRV
13			d_ETRAV
14			d_TRFEE
15			d_TRVOU
16			d_VIX
17			d_gtrends
18			rv_d
19			rv_w

Table 2.1 – Variables Selected by ABESS, LASSO and Boruta

ABESS selected the most parsimonious model as expected. Interestingly, the composition of the variables selected by LASSO and ABESS is almost the same, except that LASSO selects one additional variable, namely the first lag of the logarithmic difference of BTC market capitalization (dl_MKTCP). Moreover, the first lag of the difference in VIX index is selected by all three selection procedures. In fact, it has recently been found that the logarithmic difference of the VIX index has contemporaneous effects on BTC

volatility (López-Cabarcos et al., 2021). In our study, we focus solely on lagged effects because contemporaneous feedback is not usually helpful for investor decision-making.

Another variable selected by all three procedures is the first lag of differenced Google trends data. Interestingly, lags of Google BTC trends data were found to have predictive power over BTC returns (Arratia and López-Barrantes, 2021). Moreover, several other recent studies find the link of Google trends variable with conditional volatility (Bystrom and Krygier, 2018; Aslanidis et al., 2022; Bakas et al., 2022).

In addition, the first lag of the weekly realized volatility selected by the three selection procedures have already been shown to be effective in predicting BTC volatility (Aharon and Qadan, 2019; Bergsli et al., 2022). However, it is still an open question why the lag of daily and monthly realized volatilities are not selected by LASSO and ABESS as opposed to the latter references.

Suprisingly, among the three variable selection methods only Boruta selects a large set of technological variables. Indeed, we could not find a study to support this interesting finding and the effect of BTC technology variables on its volatility should be researched further.

Predictive Accuracy of VS-LTGARCHX Algorithms

Before delving into the analysis of results and comparative statistics, one fact is worth noting to assert the importance of explanatory variables for conditional volatility forecasting. Figure 2.1 illustrates the forecasts generated from the two benchmark models. According to the latter, the full-blown log-TGARCHX model performs better than log-GARCH(1,1) particularly during large volatility periods. Moreover, after the variable selection procedure is finished we generate one-step-ahead predictions from three variants of VS-LTGARHCX algorithm using the test set as shown in Algo 6.3. Figure 2.2 depicts one-step-ahead forecasts of BTC conditional variance from three different variants of the VS-LTGARCHX algorithm. As the figure reveals, the forecasts generated using the variants of the VS-LTGARCHX algorithm are more precise compared to the benchmark during relatively high volatility periods.

In addition, according to Table 2.2, the three variants of VS-LTGARCHX lead to better predictive accuracy than both the log-GARCH (1,1) and log-TGRACHX benchmarks in terms of RMSE and QLIKE. Moreover, the ABESS and LASSO variants always outperform the Boruta alternative with regard to the latter criteria.

On the other hand, the results of the MCS procedure are summarized in Table 2.3. The

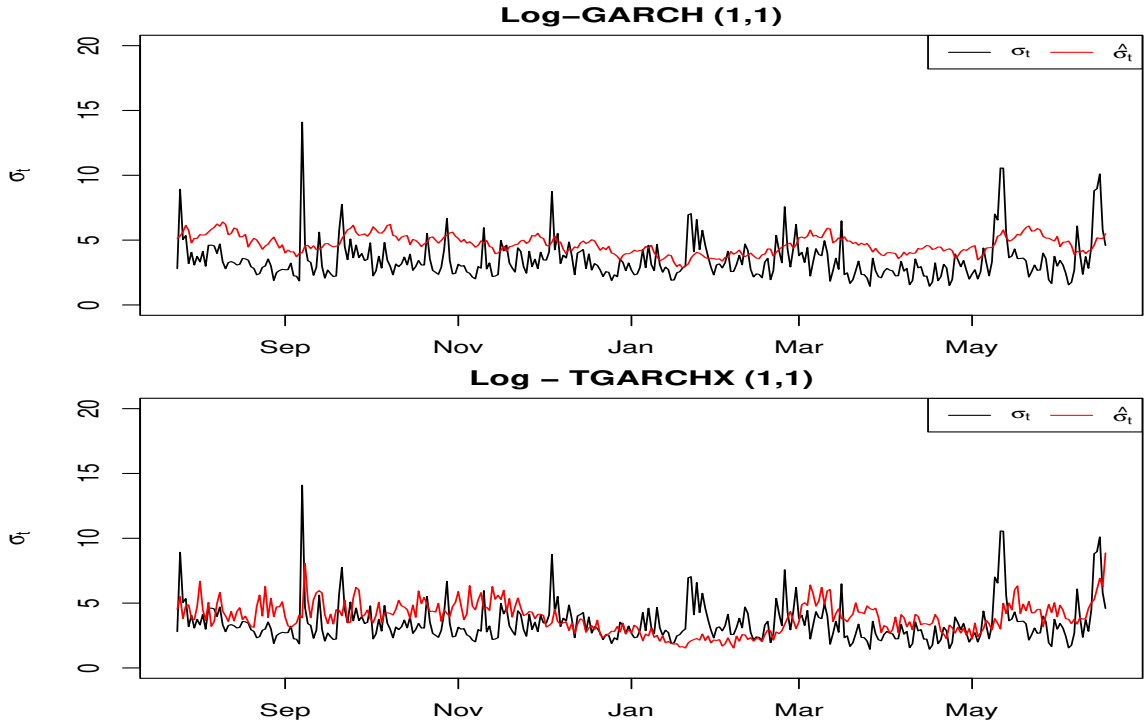


Figure 2.1 – Plot of Forecasts from Log-GARCH, Log-TGARCHX Models and Realized Standard Deviation

	RMSE	QLIKE
Log-GARCH	2.00	3.74
Log-TGARCHX	1.85	3.82
LASSO	1.60	**3.64**
ABESS	**1.59**	**3.64**
BORUTA	1.66	3.69

Table 2.2 – Predictive accuracy of models in the test set using rolling window

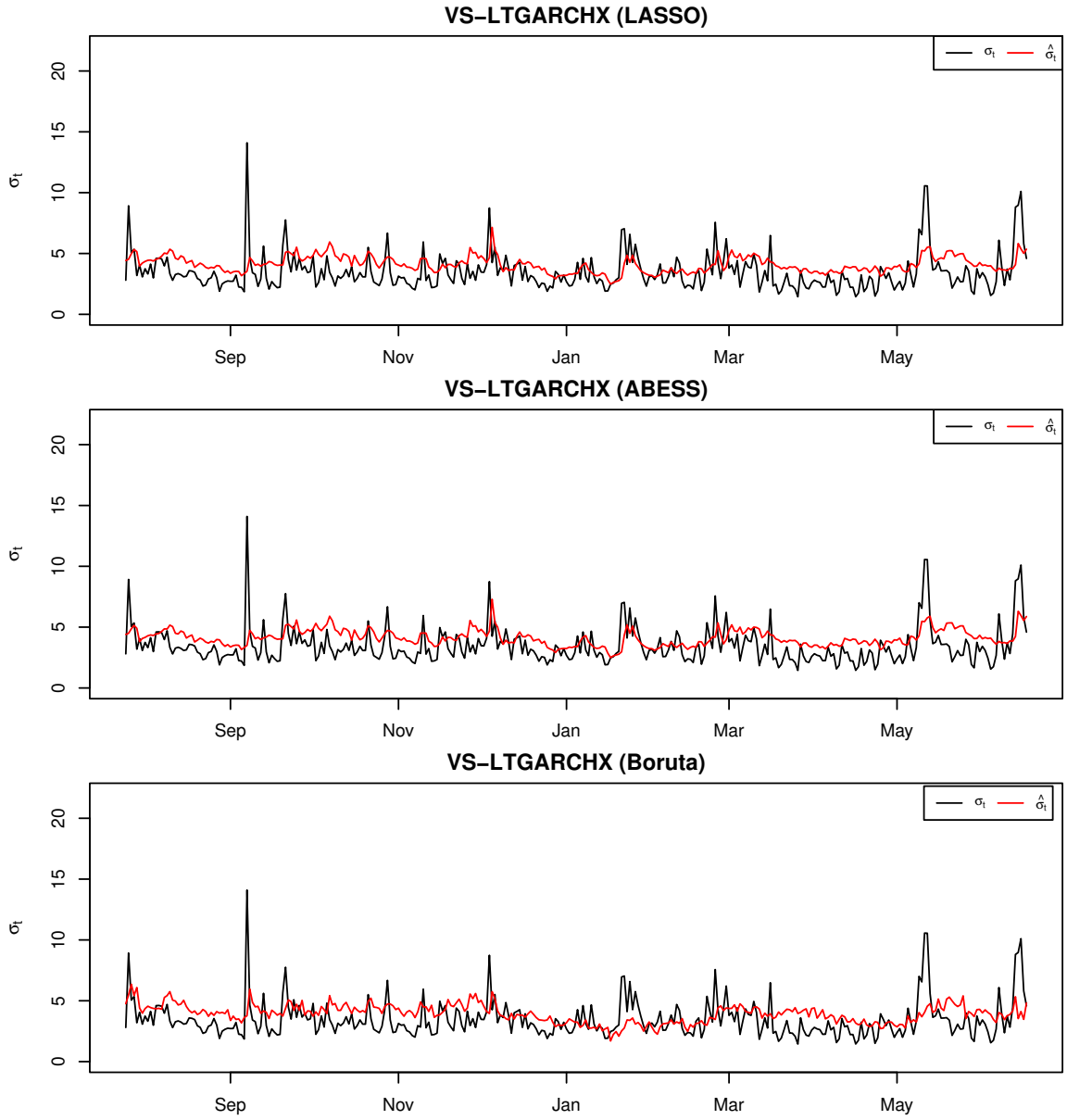


Figure 2.2 – Plot of forecasts from VS-LTGARCHX using LASSO, ABESS and Boruta variable selection procedures, respectively

MCS (QLIKE)		MCS (MSE)	
SSM	***p*-values**	**SSM**	***p*-values**
VS-LTGARCHX (ABESS)	1	VS-LTGARCHX (ABESS)	1
VS-LTGARCHX (LASSO)	0.33	VS-LTGARCHX (LASSO)	1
		VS-LTGARHCX (Boruta)	0.15

Table 2.3 – MCS Superior Set Models (SSM) using MSE and QLIKE as loss criteria

intrinsic hypothesis test for MCS was carried out at the significance level 5%. The null hypothesis states that the model has predictive performance equivalent to all other models considered (Bernardi and Catania, 2015). Then, the p-value for each model represents the probability of observing a sample performance of that particular model at least as adverse to the null hypothesis as the current one. Therefore, it is natural for models with a p-value lower than the significance level (5%) to be removed from the considered set. The MCS procedure continues until no model can be removed from the set of models and the final set of surviving models is referred to as superior set models (SSM) (Hansen et al., 2011; Bernardi and Catania, 2015).

It should be noted that the MCS procedure has eliminated VS-LTGARCHX (Boruta) while using RMSE as a loss criterion. This means that the latter is inferior to VS-LTGARHCX (LASSO) and VS-LTGARHCX (ABESS). However, it does not mean that VS-LTGARCHX (Boruta) has the same predictive ability as log-GARCH (1,1) and log-TGARCHX models. In another experiment, we restricted the initial set of models $\mathcal{M}^0$ to include only the VS-LTGARCHX (Boruta), log-GARCH (1,1) and log-TGARCHX models, and, not surprisingly, VS-LTGARCHX (Boruta) was found to be superior to the other two models while using both RMSE and QLIKE loss criteria. This means that the VS-LTGARCHX algorithm outperformed the benchmarks in all its variants. However, the Boruta variant is inferior to the LASSO and ABESS variants of VS-LTGARCHX.

As a robustness check, we report the extended list of performance metrics for the rolling window forecasting scheme in Table 6.4 of the Appendix 6.1.6. The extended list of the performance metrics includes not only QLIKE and RMSE, but also mean error (ME), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Mean Percentage Error (MPE) (see Algo 6.3 in Appendix 6.1.5 and Hyndman and

Athanasopoulos (2018) for details).

Furthermore, it is not surprising that the full-blown log-TGARCHX model outperforms all other models in terms of ME and MPE, as shown in Table 6.4. Tibshirani correctly points out that : "... the OLS estimates often have low bias but large variance; prediction accuracy can sometimes be improved by shrinking or setting to 0 some coefficients. By doing so, we sacrifice a little bias to reduce the variance of the predicted values and hence may improve the overall prediction accuracy" (Tibshirani, 1996, p. 267). The unrestricted log-TGARCHX model (estimated using OLS) should have less bias compared to its restricted alternatives, as inferred from the latter quote. Therefore, ME and MPE which are nothing but the average and the weighted average of forecast errors, respectively, should have the smallest value for the unrestricted log-TGARCHX model.

Moreover, as an additional robustness check, we follow Hansen et al. (2011) and test our algorithm using the recursive forecasting scheme. An extended list of performance metrics for the recursive window forecasts given in Table 6.5 of Appendix 6.1.6 supports our conclusions in this subsection, thus showing the robustness of the proposed algorithms to varying environments.

Finally, we plot the autocorrelation function (ACF) for daily realized volatility and the ACF of volatility predictions from the models considered. We find evidence of volatility clustering, a well-known feature of financial markets (Cont, 2001) in the BTC market. We show that the ACF structures of the predicted volatilities from different models overestimate the true underlying ACF structure (for details, see Appendix 6.1.7).

2.5 Conclusion

Conditional volatility forecasting of financial assets is essential for risk management, hedging, and portfolio optimization. For this reason, there are a plethora of models dealing with volatility forecasting. The GARCH models family is arguably the main approach. However, generic GARCH models are restrictive in the exogenous variables that can be included. For instance, they do not allow us to include some variables that may contain relevant information about conditional volatility. For this reason, we focus on log-GARCH models, which relax the restrictions on the covariates in the conditional variance equation, which also allows for a simpler estimation procedure. However, this flexibility comes at the price of potentially introducing overfitting. To overcome this problem, we propose the VS-LTGARCHX algorithm to incorporate the variable selection procedure into the

log-TGARCHX estimation process. The algorithm is very flexible in the sense that any variable selection procedure can be used within its framework. However, in this paper, we use LASSO, ABESS and Boruta variable selection procedures separately within the VS-LTGARCHX algorithm to select the subset of exogenous variables and asymmetry lags while estimating log-TGARCHX models. Furthermore, we apply the VS-LTGARCHX algorithm to extremely volatile BTC markets using 42 conditioning variables. Three different restricted versions of log-TGARHCX models obtained with the VS-LTGARCHX algorithm (using LASSO, ABESS, and Boruta selection procedures, respectively) outperform the benchmark log-GARCH(1,1) and log-TGARCHX(1,1) models in one-step-ahead conditional volatility prediction based on different accuracy metrics. To assert the superiority of the VS-LTGARCHX variants, we use a formal procedure called MCS which indicates that the forecasts generated from the three variants of VS-LTGARCHX are superior to those from the benchmark models.

As a result, the main contributions of this paper are twofold. On the one hand, our study proposes a novel approach to model conditional volatility in the GARCH models domain, which may be useful for academicians, investors, and policy makers. On the other hand, we analyze volatility in the evolving and relatively young BTC market with 42 different exogenous variables. To the best of our knowledge, no study has been conducted to study BTC volatility with as many exogenous covariates as we do.

Ultimately, irrespective of the novelties of our paper, there are several limitations which paves the way for further research. First, the set of exogenous variables can be extended. Second, the selection procedures in this study cannot handle the simultaneous selection of the subset of ARCH, GARCH, asymmetry terms and exogenous variables. Rather, our approach is based on the stepwise selection, which may be suboptimal due to the correlation structure of the variables. In addition, we have not considered ARCH/GARCH orders and asymmetry orders greater than 1 in our empirical applications, which presumably impeded further improvement of the predictive accuracy of VS-LTGARCHX. Furthermore, the application of the VS-LTGARCHX algorithm to log-ARCHX and ARCHX models, and its comparison to the respective existing methods might be an intriguing direction of research. Finally, a multivariate version of the proposed algorithm could be used to perform variable selection for the multivariate log-TGARCHX model to study the volatility of several cryptocurrencies simultaneously.

Chapter 3

Cross Entropy With Symmetry Breaking and K-Means - A Hybrid Algorithm for Clustering

3.1 Introduction

Centroid-based clustering and clusterwise linear regression problems have been subject to significant interest in statistics with various applications in different domains (Long et al., 2023). Lloyd was first to introduce the centroid-based clustering problem and the K-means algorithm to solve it (Lloyd, 1982). Späth (1979) introduced clusterwise linear regression problem and a solution method called exchange algorithm. The exchange algorithm to solve the clusterwise linear regression problem turns out to be an extension of K-means algorithm as documented in the related literature (Long et al., 2023).

Both K-means and exchange algorithms are fast heuristics but they are only guaranteed to converge to the local optimum (Lloyd, 1982; Späth, 1979). To improve the solution to clustering problem one can apply global search algorithms. In the seminal paper of Kroese et al. (2007), one of the well-known global search algorithms, namely the Cross Entropy (CE) method (Rubinstein, 1999) has been applied to the centroid-based clustering problem to find the optimal partition for several real and synthetic datasets. Their numerical experiments demonstrate that CE method performs better than the competitor algorithms, namely, K-means algorithm, Fuzzy K-means algorithm and Linear Vector Quantization in terms of reaching the global optimum. A detailed exposition of the clustering algorithms can be found in Xu and Tian (2015); Ezugwu et al. (2022) and references therein.

Kroese et al. (2007) conclude that the CE is *"an easily implementable and accurate*

alternative" to its competitors in terms of reaching the global optimum, however, it is computationally more demanding and, hence, the convergence is slower. Moreover, the authors show that the centroid-based clustering problem can be formulated either as a combinatorial or multi-extremal continuous optimization program and CE method can solve both of them.

In this paper, we delve into reasons why CE method converges slower and why the method returns less accurate results when applied to combinatorial formulation of the centroid-based clustering problem. In particular, we argue that the CE method's performance applied to this setting can be improved by making several adjustments. The proposed modifications to the generic CE method are based on well-established theoretical results (see Theorem 3.1 and Theorem 3.6). Our first finding is the fact that the generic CE method fails to converge theoretically to the global optimum in the presence of symmetries in the objective function which is the case of the clustering problem. Therefore, we suggest incorporating symmetry breaking constraints into the CE.

Our second contribution consists in modifying slightly the generic CE by the use of a geometrically decaying quantile rank parameter $\rho^{(t)}$ instead of a fixed quantile rank parameter ρ. We formally prove that using a geometrically decaying structure for $\rho^{(t)} = \rho \cdot e^{-qt}$ guarantees theoretical convergence to the global optimum, asssuming $q > 0$ and $\rho \in [0, 1]$. When the decay parameter is set to a high value in its range, for example q = 0.9, the convergence rate may be slowed down. Therefore, to enhance the convergence speed regardless to the choice of q, we propose to integrate K-means algorithm into the CE method. These three modifications incorporated into the generic CE algorithm constitute our novel hybrid algorithm, which we refer to as Cross Entropy with Symmetry Breaking and K-means (CESBKM) in short. Additionally, we show the convergence of the CESBKM algorithm and prove the monotonicity of quantile sequences generated by CE algorithm under much milder conditions compared to the related literature (Lieber, 1998; Rubinstein, 1999).

As a final contribution, we propose to apply the principle of clusterwise regression to the Heterogeneous Autoregressive (HAR) model by Corsi (2009), whose parameters can be estimated using both the CE and the CESBKM algorithm. We call this new extension the Clusterwise Heterogeneous Autoregressive (CHAR) model. Although many extensions of the HAR model exist (Qiu et al., 2019; Clements and Preve, 2021), the CHAR model has

not yet been considered in the literature. The advantage of using the CHAR model is that it is a simple semi-parametric method where no distributional assumption is made about the cluster assignment mechanism. Interestingly, our numerical analysis in Section 3.7 shows that the CHAR model provides a significant improvement over the generic HAR model in terms of the model fit for Bitcoin market. Finally, we apply the CHAR model with CESBKM on Bitcoin data. This application highlights the good performances of using the CESBKM instead of the generic CE method.

The paper is structured as follows. In Section 3.2, centroid-based clustering and clusterwise linear regression problems are briefly introduced. In Section 3.3, we describe how the K-means algorithm for clustering problems works and then, describe theoretical foundations of the CE method concisely. Furthermore, in Section 3.4 we provide motivations for modifying the generic CE method and hybridizing the latter with the K-means algorithm. Also, in Section 3.5 we present the CESBKM algorithm and provide a rigorous proof of its convergence. Finally, in Section 3.7 we perform simulation studies on 50 synthetic datasets, use the novel CESBKM algorithm to estimate the CHAR model for Bitcoin volatility and then, conclude.

3.2 Clusterwise Linear Regression and centroid-based clustering problems

Clusterwise linear regression problem, introduced by Späth (1979), requires methods that combine linear regression analysis and clustering. The core idea is to simultaneously fit multiple regression surfaces to a given dataset and attribute each response to the closest regression surface. On the other hand, the centroid-based clustering problem, formulated by Lloyd (1982), aims to partition the dataset in such a way that some empirical loss function is minimized.

To elaborate on both problems, we first present the clusterwise linear regression problem in Section 3.2.1. Subsequently, in Section 3.2.2, we present the centroid-based clustering problem and discuss some of its properties in detail.

3.2.1 Clusterwise Linear Regression Problem

Consider a dataset $\mathcal{Z} = \{(\boldsymbol{z}_j, \boldsymbol{y}_j)\}_{j=1}^{n}$ where $\boldsymbol{z}_j \in \mathbb{R}^{d_z}$ is the target vector for the j-th data point and $\boldsymbol{y}_j \in \mathbb{R}^{d_y}$ is a feature vector. The aim is to partition the dataset into K

distinct clusters, each defined by a specific regression model such that some empirical loss function is minimized. The loss function for clusterwise linear regression (DeSarbo and Cron, 1988) can be re-written as follows:

$$\mathcal{W}(\boldsymbol{c}, I) = \sum_{k=1}^{K} \sum_{i \in I_k} \|\boldsymbol{z}_i - \boldsymbol{c}_k \boldsymbol{y}_i\|^2, \tag{3.1}$$

where:

— $I = \{I_1, \ldots, I_K\}$ denotes a partition of the indices of the dataset ($\forall k \neq l$, $I_k \cap I_l = \emptyset$ and $\cup_k I_k = \{1, \ldots, n\}$).

— $\boldsymbol{c} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K\}$ is a collection of matrices, where $\boldsymbol{c}_k$ is the matrix of regression coefficients for cluster k.

— $\boldsymbol{c}_k \boldsymbol{y}_j$ is the predicted value of $\boldsymbol{z}_j$ using the regression model in cluster k.

There are two necessary conditions for the existence of a unique solution. First of them is $d_y \leq n_k$ where n_k is the number of observations in each cluster. This condition implies that $K \cdot d_y \leq \sum_{k=1}^{K} n_k = n$. Additionally, the second necessary condition is that for each cluster I_k, the design matrix $\boldsymbol{Y}_k = (\boldsymbol{y})_{i \in I_k} = (\boldsymbol{y}_{i_1}, \ldots, \boldsymbol{y}_{i_{n_k}})$ must have full row rank d_y to ensure the identification of the regression coefficients $\boldsymbol{c}_k$. This means that within each cluster, the explanatory variables must be linearly independent (DeSarbo and Cron, 1988, p. 253).

Interestingly, the conventional centroid-based clustering problem can be seen as a special case of clusterwise linear regression. In fact, if $\boldsymbol{y}_j = 1$ for all $j = 1, \ldots, n$, then the clusterwise linear regression problem reduces to the conventional centroid-based clustering problem shown in (3.2) below. Furthermore, logically (because the size of $\boldsymbol{y}_j$ is equal to one) the condition $d_y \leq n_k$ in clusterwise linear regression simplifies to $n_k \geq 1$ for all $k = 1, \ldots, K$, which in turn implies $K \leq n$.

For clarity of exposition, the rest of the paper focuses on the centroid-based clustering problem using the definitions in Section 3.2.2 unless stated otherwise. However, all the analysis is also applicable to the clusterwise linear regression problem, unless specified otherwise.

3.2.2 Centroid-based clustering problem

Suppose $\mathcal{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}$ is the set of points in d-dimensional Euclidean space. In the following, we will make use of the partition $R = \{R_1, \ldots, R_K\}$ of the dataset $\mathcal{Z}$ consisting

of $R_k = \{\boldsymbol{z}_i : i \in I_k\}$ for $k = 1, \ldots, K$. As before, the goal is to construct a partition of the dataset into K ($K \leq n$) distinct clusters. A frequently used empirical loss function to minimize in literature (Ikotun et al., 2023) is given as

$$\mathcal{W}(\boldsymbol{c}, R) = \sum_{k=1}^{K} \sum_{\boldsymbol{z} \in R_k} \|\boldsymbol{z} - \boldsymbol{c}_k\|^2. \tag{3.2}$$

Definition 3.1. *A partition (clustering) R of the dataset $\mathcal{Z}$ is said to be proper if,*

1. *It assigns at least one data point to each of the K clusters, which means $R_k \neq \emptyset$;*
2. *Each data point $\boldsymbol{z}_i$ is assigned to one and only one cluster R_k, which means $R_i \cap R_k = \emptyset$, for $i \neq k$;*
3. *The union of all clusters is equal to the complete dataset: $\cup_{k=1}^{K} R_k = \mathcal{Z}$.*

Although the loss function in (3.2) is more standard in literature, for the purposes of this study, it is more convenient to re-formulate the latter in terms of the centroid vectors and the *cluster vector* similar to Kroese et al. (2007). We adopt the notation of Kroese et al. (2007) as much as we can to make it easy for the readers to compare CE and the proposed hybrid algorithm in the context of the clustering problem. More precisely, using the notation in the latter reference, let

$$\boldsymbol{x} = \{x_1, \ldots, x_n\}$$

be the *cluster vector* where $x_i = k$ if $\boldsymbol{z}_i \in R_k$. Then, we consider

$$\boldsymbol{z}_{ik} = \mathbb{1}_{\{x_i = k\}} \boldsymbol{z}_i, \tag{3.3}$$

where $\mathbb{1}_{\{x_i = k\}}$ denotes the indicator function taking the value of the one if $x_i = k$ and zero, otherwise. Then, using these new notations, the clustering problem can be defined as the minimization of the following loss function which depends both on centroid vector $\boldsymbol{c}$ and the cluster vector $\boldsymbol{x}$:

$$\mathscr{W}(\boldsymbol{c}, \boldsymbol{x}) = \sum_{k=1}^{K} \sum_{i=1}^{n} \mathbb{1}_{\{x_i = k\}} ||\boldsymbol{z}_{ik} - \boldsymbol{c}_k||^2. \tag{3.4}$$

Furthermore, as will be shown below, the cluster vector $\boldsymbol{x}$ and the centroid vector $\boldsymbol{c}$ are interrelated in the sense that we can derive the optimal value of one of them given the optimal value of the other. Consequently, the task of minimizing the function $\mathscr{W}$ in (3.4)

can be redefined as either a multi-extremal continuous optimization problem or a combinatorial optimization problem, with both approaches yielding the same global solution (Kroese et al., 2007). To approach the clustering issue as a multi-extremal continuous optimization problem, the cluster vector is maintained at its optimal state, and the loss function is reformulated to depend solely on the centroid vector. Specifically,

$$\mathcal{L}(\boldsymbol{c}) = \min_{R} \mathcal{W}(\boldsymbol{c}, R) = \min_{\boldsymbol{x}} \mathscr{W}(\boldsymbol{c}, \boldsymbol{x}) = \sum_{k=1}^{K} \sum_{i=1}^{n} \mathbb{1}_{\{x_i^\star = k\}} ||\boldsymbol{z}_{ik} - \boldsymbol{c}_k||^2, \tag{3.5}$$

where $\boldsymbol{z}_{ik}$ as defined in (3.3) and the optimal solution $x_i^\star \in \boldsymbol{x}^\star$ for $i = 1, \ldots, n$ is obtained by assigning each data point $\boldsymbol{z}_i$ to cluster k, if $\|\boldsymbol{z}_i - \boldsymbol{c}_k\|^2 \leq \|\boldsymbol{z}_i - \boldsymbol{c}_l\|^2$ for all $l \neq k$. Here and in the sequel, in case of equalities, the assignments can be made randomly.

Furthermore, to address the clustering problem as a combinatorial optimization task, we set the centroid vector to its optimal value and minimize the discontinuous function below through combinatorial optimization:

$$\mathscr{L}(\boldsymbol{x}) = \min_{\boldsymbol{c}} \mathscr{W}(\boldsymbol{c}, \boldsymbol{x}) = \sum_{k=1}^{K} \sum_{i=1}^{n} \mathbb{1}_{\{x_i = k\}} ||\boldsymbol{z}_{ik} - \boldsymbol{c}_k^\star||^2, \tag{3.6}$$

where the optimal solution for centroids $\boldsymbol{c}_k$ has a closed form solution as below:

$$\boldsymbol{c}_k^\star = \frac{1}{\sum_{i=1}^{n} \mathbb{1}_{\{x_i = k\}}} \sum_{i=1}^{n} \boldsymbol{z}_{ik}, \quad \text{for} \quad k = 1, \ldots, K. \tag{3.7}$$

We have to note that the loss function $\boldsymbol{x} \mapsto \mathscr{L}(\boldsymbol{x})$ in (3.6) depends solely on cluster vector $\boldsymbol{x}$ which should be defined appropriately. Since the cluster vector $\boldsymbol{x}$ is closely related to the partition $R = \{R_1, \ldots, R_K\}$, it should also obey the conditions on the proper partition given in Definition 3.1. Incidentally, the loss function $\mathscr{L}(\cdot)$ in (3.6) possesses certain symmetry property with respect to the cluster labels. Section 3.4 provides a thorough discussion on the challenges it presents for numerical optimization. To fully comprehend the implications of this symmetry on numerical optimization, it is essential to first understand the optimization techniques employed in this study. Therefore, in Section 3.3.1 we discuss the heuristic K-means technique and briefly outline some of its properties. In Section 3.3.2 we describe the Cross-Entropy (CE) method, which is a pivotal numerical optimization technique utilized in this study, and leave important proofs related to the convergence of the algorithm to Section 3.5.

3.3 K-Means Algorithm and CE Algorithm

The K-means algorithm (Hartigan and Wong, 1979) is a popular heuristic for iteratively solving both the centroid-based clustering problem and the clusterwise linear regression problem. The key difference in applying the K-means algorithm to these two problems lies in the assignment step: for the centroid-based clustering problem, each data point is assigned to the nearest cluster centroid, while for the clusterwise linear regression problem, each response data point is assigned to the closest regression surface.

Another algorithm heuristic for the clusterwise linear regression problem is the exchange algorithm proposed by Späth (1979). However, this algorithm is a modified version of the K-means clustering algorithm (Long et al., 2023) and can be applied to the centroid-based clustering problem due to the interrelation between the two problems, as described in Section 3.2.1.

This section is organized as follows. In Section 3.3.1, we discuss the K-means algorithm for centroid-based clustering problem. In Section 3.3.2, we describe the CE algorithm for solving more general combinatorial optimization problems.

3.3.1 K-Means Algorithm

K-means is an iterative algorithm which begins with an initial centroid vector $\boldsymbol{c}^{(0)} = (\boldsymbol{c}_1^{(0)}, \ldots, \boldsymbol{c}_K^{(0)})$. Then, for each iteration $t \in \mathbb{N}^\star$, the algorithm consists of two steps:

1. **Cluster Assignment**: Given the centroid vector at iteration $t-1$, denoted as $\boldsymbol{c}^{(t-1)} = (\boldsymbol{c}_1^{(t-1)}, \ldots, \boldsymbol{c}_K^{(t-1)})$, each data point $\boldsymbol{z}_i$ is assigned to cluster k, i.e. $x_i^{(t)} = k$, if $\|\boldsymbol{z}_i - \boldsymbol{c}_k^{(t-1)}\|^2 \leq \|\boldsymbol{z}_i - \boldsymbol{c}_l^{(t-1)}\|^2$ for all $l \neq k$. This is equivalent to minimizing the objective function in (3.4) with respect to the cluster vector $\boldsymbol{x}$ and given a fixed centroid vector $\boldsymbol{c}^{(t-1)}$. The resulting cluster vector at iteration t is then $\boldsymbol{x}^{(t)} = \{x_1^{(t)}, \ldots, x_n^{(t)}\}$.
2. **Updating**: Each centroid $\boldsymbol{c}_k^{(t)}$ is recalculated (updated) as the mean of all points assigned to cluster k determined by the cluster vector $\boldsymbol{x}^{(t)}$, that is

$$\boldsymbol{c}_k^{(t)} = \frac{1}{\sum_{i=1}^n \mathbb{1}_{\{x_i^{(t)}=k\}}} \sum_{i=1}^n \mathbb{1}_{\{x_i^{(t)}=k\}} \boldsymbol{z}_i, \quad \text{for} \quad k = 1, \ldots, K. \tag{3.8}$$

This is equivalent to minimizing the objective function in (3.4) with respect to the centroid vector $\boldsymbol{c}$ and given a fixed cluster vector $\boldsymbol{x}^{(t)}$.

The above steps are iteratively performed until convergence, typically when the change in centroid positions or the assignment of data points to clusters becomes negligible or does not change at all between successive iterations.

The convergence properties of the K-means algorithm have been studied in Jin and Han (2017) and the references therein. It will be useful to restate some of these results as they are part of the motivations for our study.

Theorem 3.1 (Jin and Han (2017)). *For every fixed proper cluster vector $\boldsymbol{x}$ that obey the requirements of the proper partition in Definition 3.1, the function $\boldsymbol{c} \mapsto \mathcal{L}(\boldsymbol{c})$ in (3.5) has a unique minimum.*

Proof. The objective function $\mathcal{L}$ in (3.5) is strictly convex in $\boldsymbol{c}$ given the proper cluster vector and thus, has a unique minimum which completes the proof. □

Theorem 3.2 (Jin and Han (2017)). *The sequence of loss functions $\{\mathscr{W}(\boldsymbol{c}^{(t)}, \boldsymbol{x}^{(t)})\}_{t\in\mathbb{N}}$ generated by iterations of the K-means algorithm is non-increasing.*

Proof. Clearly, in each iteration t of the K-means algorithm, the inequality $\mathscr{W}(\boldsymbol{c}^{(t)}, \boldsymbol{x}^{(t)}) \leq \mathscr{W}(\boldsymbol{c}^{(t-1)}, \boldsymbol{x}^{(t-1)})$ holds. This is because the algorithm systematically reduces the loss function $\mathscr{W}$ in two steps: initially by optimizing $\boldsymbol{x}$ with a fixed $\boldsymbol{c}^{(t-1)}$ to determine $\boldsymbol{x}^{(t)}$, and subsequently by optimizing $\boldsymbol{c}$ with the updated $\boldsymbol{x}^{(t)}$ to derive $\boldsymbol{c}^{(t)}$. This completes the proof. □

Corollary 3.1 (Jin and Han (2017)). *From Theorem 3.2, it can be inferred that the sequence $\{\mathscr{L}(\boldsymbol{x}^{(t)})\}_{t\in\mathbb{N}}$ and the sequence $\{\mathcal{L}(\boldsymbol{c}^{(t)})\}_{t\in\mathbb{N}}$ generated by cluster assignment step and by update step of the K-means algorithm, respectively, are also monotone non-increasing.*

Theorem 3.3 (Jin and Han (2017)). *The K-means algorithm is guaranteed to converge to a local minimum in a finite number of iterations.*

Proof. The algorithm converges to a local optimum within a finite number of iterations, due to the following reasons:

1. According to Theorem 3.2, the sequence $\{\mathscr{W}(\boldsymbol{c}^{(t)}, \boldsymbol{x}^{(t)})\}_{t\in\mathbb{N}}$ is non-increasing.

2. Theorem 3.1 guarantees the uniqueness of the optimal solution in each update step. This uniqueness, combined with the monotonicity of $\{\mathscr{W}(\boldsymbol{c}^{(t)}, \boldsymbol{x}^{(t)})\}_{t\in\mathbb{N}}$, ensures that the algorithm does not revisit an optimal solution $(\boldsymbol{c}^{(t)}, \boldsymbol{x}^{(t)})$ unless it has converged.
3. The total number of possible cluster vector configurations $\{1, \ldots, K\}^n$ is finite.

This completes the proof. □

In fact, from Theorem 3.3 it follows that the K-means algorithm converges in a finite number of steps but, maybe, to a local optimum. The following section describes the CE algorithm for combinatorial optimization and outlines the working mechanism and foundation of the CE method.

3.3.2 The CE algorithm for Optimization

For simplicity, we follow the notation and the narrative in de Boer et al. (2005) as closely as possible, and address the problem for combinatorial optimization only. Nonetheless, the results can be well extended to continuous cases.

Consider minimizing a real-valued multivariate loss function $\boldsymbol{x} \mapsto L(\boldsymbol{x})$ where $\boldsymbol{x} \in \mathcal{X}$ and $\mathcal{X}$ is a finite set of states. Denote the unique global minimum of L over $\mathcal{X}$ by $\gamma^\star$:

$$\gamma^\star = \min_{\boldsymbol{x}\in\mathcal{X}} L(\boldsymbol{x}). \tag{3.9}$$

In (3.9) the goal is to find both $\gamma^\star$ and the corresponding state

$$\boldsymbol{x}^\star = \operatorname*{argmin}_{\boldsymbol{x}\in\mathcal{X}} L(\boldsymbol{x}). \tag{3.10}$$

To that end, the CE algorithm associates this minimization problem with that of the rare-event probability estimation. It is assumed that there is a random vector $\boldsymbol{X}$ following some probability law $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{X} = \boldsymbol{x}) = f(\boldsymbol{x}; \boldsymbol{\theta})$ over $\mathcal{X}$, with $\boldsymbol{\theta}$ being the parameter vector. Thereby, the minimization problem defined in (3.9) is reformulated as the efficient estimation of the probability of the rare event

$$l = \mathbb{P}_{\boldsymbol{\theta}}(L(\boldsymbol{X}) \leq \gamma^\star) = \mathbb{E}_{\boldsymbol{\theta}}\left[\mathbb{1}_{\{L(\boldsymbol{X})\leq\gamma^\star\}}\right] = \sum_{\boldsymbol{x}} \mathbb{1}_{\{L(\boldsymbol{x})\leq\gamma^\star\}} f(\boldsymbol{x}; \boldsymbol{\theta}). \tag{3.11}$$

The usual empirical estimator of the latter probability is

$$\hat{l}(\gamma^\star) = \frac{\sum_{i=1}^{N} \mathbb{1}_{\{L(\boldsymbol{X}_i)\leq\gamma^\star\}}}{N}, \tag{3.12}$$

where $\boldsymbol{X}_i \overset{\text{iid}}{\sim} f(\boldsymbol{x};\boldsymbol{\theta})$ and $\mathbb{1}_{\{L(\boldsymbol{X})\leq\gamma^\star\}}$ is the indicator function that equals 1 if $L(\boldsymbol{X}) \leq \gamma^\star$ and 0 otherwise. Incidentally, the estimator in (3.12) is inefficient due to rarity of the event because in practice, simulation from $f(\boldsymbol{x};\boldsymbol{\theta})$ with finite samples may return very few if any realizations for which $\mathbb{1}_{\{L(\boldsymbol{X}_i)\leq\gamma^\star\}}$ is equal to one. There are several methods to tackle this problem in related literature. One of them is called *importance sampling* which itself has many variations. The fundamental principle of importance sampling lies in identifying an alternative probability distribution g, through a change of measure. This alternative distribution is selected such that the probability of the event of interest is not considered rare under g. Mathematically, the probability l of the event $\{L(\boldsymbol{X}) \leq \gamma^\star\}$ under the original distribution can be expressed as:

$$l = \mathbb{P}(L(\boldsymbol{X}) \leq \gamma^\star) = \mathbb{E}_{\boldsymbol{\theta}}\left[\mathbb{1}_{\{L(\boldsymbol{X})\leq\gamma^\star\}}\right] = \mathbb{E}_g\left[\mathbb{1}_{\{L(\boldsymbol{X})\leq\gamma^\star\}}\frac{f(\boldsymbol{X};\boldsymbol{\theta})}{g(\boldsymbol{X})}\right], \tag{3.13}$$

where $\mathbb{E}_{\boldsymbol{\theta}}$ and $\mathbb{E}_g$ denote the expected values with respect to the original distribution $f(\boldsymbol{x};\boldsymbol{\theta})$ and the alternative distribution g, respectively. Moreover, we assume that the density function g dominates the product of the indicator function $\mathbb{1}_{\{L(\boldsymbol{X})\leq\gamma^\star\}}$ and the probability density function $f(\boldsymbol{X};\boldsymbol{\theta})$, i.e. $\mathbb{1}_{\{L(\boldsymbol{X})\leq\gamma^\star\}}f(\boldsymbol{X};\boldsymbol{\theta}) = 0$, whenever $g(\boldsymbol{X}) = 0$ yet g does not dominate $f(\boldsymbol{X};\boldsymbol{\theta})$ alone (Rubinstein and Kroese, 2016). The quotient $\frac{f(\boldsymbol{X};\boldsymbol{\theta})}{g(\boldsymbol{X})}$ is the *likelihood ratio* which provides the penalty for bias introduced by g and ensures that the expectation of the expression under g given in (3.13) is equivalent to the expectation of the indicator function under $f(\boldsymbol{x};\boldsymbol{\theta})$. Consequently, the probability of interest, l, can be estimated by sampling from distribution g rather than from $f(\boldsymbol{x};\boldsymbol{\theta})$, and by employing the *likelihood ratio estimator* as delineated below:

$$\hat{l}(\gamma^\star) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{\{L(\boldsymbol{X}_i)\leq\gamma^\star\}}\frac{f(\boldsymbol{X}_i;\boldsymbol{\theta})}{g(\boldsymbol{X}_i)}, \tag{3.14}$$

where $\boldsymbol{X}_i \overset{\text{iid}}{\sim} g(\boldsymbol{x})$.

For further details and foundational concepts, refer to Rubinstein and Kroese (2004) and the references therein.

Undoubtedly, the natural question is how to find the optimal density $g^\star$ which minimizes the variance of the likelihood ratio estimator in (3.14) above. Theoretically, the optimal importance sampling density $g^\star$, that sometimes referred to as *zero-variance change of measure* (because it holds zero variance for the likelihood ratio estimator), should be proportional to $f(\boldsymbol{x};\boldsymbol{\theta})$ as shown in Rubinstein et al. (1998):

$$g^\star(\boldsymbol{x}) = \begin{cases} \frac{f(\boldsymbol{x};\boldsymbol{\theta})}{\sum_{\boldsymbol{x}':L(\boldsymbol{x}')\leq\gamma^\star} f(\boldsymbol{x}';\boldsymbol{\theta})} & \text{if } L(\boldsymbol{x}) \leq \gamma^\star \\ 0 & \text{if else,} \end{cases} \tag{3.15}$$

where $f(\boldsymbol{x};\boldsymbol{\theta})$ is the original distribution, referred to as *nominal distribution* in related literature (Rubinstein, 1999; de Boer et al., 2005).

The problem with the optimal density $g^\star$ is that it is exactly proportional to l, the unknown quantity that we want to estimate. Various methods have been proposed in literature to approximate the optimal density (Rubinstein et al., 1998). One of them is to choose the importance sampling density g from the same parametric family $\{f(\cdot;\boldsymbol{\theta}), \boldsymbol{\theta} \in \boldsymbol{\Theta})\}$ as the *nominal density* $f(\boldsymbol{x};\boldsymbol{\theta})$, i.e.

$$g(\boldsymbol{x}) = f(\boldsymbol{x};\boldsymbol{p}) \quad \text{where} \quad \boldsymbol{p} \in \boldsymbol{\Theta},$$

and parametrize the importance sampling density g in such a way that the Kullback-Leibler divergence between g and the theoretical zero-variance change of measure $g^\star$ is minimized (Rubinstein, 1999). Minimizing the Kullback-Leibler divergence is equivalent to maximization of the negative cross-entropy measure, and hence, the latter boils down to maximizing in $\boldsymbol{p}$ the function

$$\mathcal{D}(\boldsymbol{p}) = \mathbb{E}_{\boldsymbol{\theta}} \left[\mathbb{1}_{\{L(\boldsymbol{X})\leq\gamma^\star\}} \ln f(\boldsymbol{X};\boldsymbol{p}) \right]. \tag{3.16}$$

Due to the fact that $\{L(\boldsymbol{X}) \leq \gamma^\star\}$ is a rare event under the density $f(\boldsymbol{X};\boldsymbol{\theta})$, estimating *optimal reference parameter* $\boldsymbol{p}^\star$ by simulation from $f(\boldsymbol{X};\boldsymbol{\theta})$ is of no practical use (Rubinstein and Kroese, 2004). Consequently, a *two-phase* Cross-Entropy (CE) method is proposed, in which the *reference parameter* vector $\boldsymbol{p}$ and the *level* γ are both repeatedly updated. More precisely, the CE method is an iterative algorithm that generates a sequence of pairs $\{\boldsymbol{p}^{(t)}, \gamma^{(t)}\}_{t\in\mathbb{N}}$, aiming to identify or estimate the optimal reference parameter $\boldsymbol{p}^\star$ (Rubinstein and Kroese, 2004, p. 73). Under such an iterative scheme,

the optimal reference parameter $\boldsymbol{p}^\star$ will be reached in finite number of iterations and the optimal importance sampling density $f(\boldsymbol{x};\boldsymbol{p}^\star)$ will locate probability mass equal to one on the event of interest $\{L(\boldsymbol{X}) \leq \gamma^\star\}$, under regularity conditions (Rubinstein and Kroese, 2004). In particular, we need to choose a parametric family such that there exists a $\boldsymbol{p}$ where $f(\boldsymbol{x}^*;\boldsymbol{p}) = 1$. Therefore, just one sample from $f(\boldsymbol{x};\boldsymbol{p}^\star)$ is enough to generate $\boldsymbol{x}^\star$ corresponding to (3.10) with probability of one.

The provided explanation elucidates the connection between estimating the probability $l = \mathbb{P}_{\boldsymbol{\theta}}(L(\boldsymbol{X}) \leq \gamma^\star)$ and optimization of the function in (3.9). Under certain regularity conditions, the Cross-Entropy (CE) method facilitates efficient estimation of the probability $\mathbb{P}_{\boldsymbol{\theta}}(L(\boldsymbol{X}) \leq \gamma)$ for any real number γ, including for the optimal value $\gamma^\star = \min_{\boldsymbol{x}\in\mathcal{X}} L(\boldsymbol{x})$. If $\boldsymbol{p}^\star$ represents the optimal reference parameter vector, which solves the optimization problem specified in (3.16), then

$$\mathbb{P}_{\boldsymbol{p}^\star}(L(\boldsymbol{X}) \leq \gamma^\star) = 1.$$

The discussion above illustrates the interrelation between the efficient estimation of the probability in (3.11) and the minimization problem presented in (3.9). Nonetheless, a minor distinction exists between these two concepts. The primary distinction between applying the Cross-Entropy (CE) method for rare event probability estimation and CE for optimization tasks is that, in the context of rare event probability estimation, it is imperative to have a well-defined nominal (original) density function $f(\boldsymbol{x};\boldsymbol{\theta})$ established beforehand. On the other hand, for an optimization problem, such a nominal density is usually missing. Hence, in Algorithm 3.1, which presents the deterministic version of the CE algorithm for discrete optimization, before commencing the iterative process at $t = 1$, it is essential to initialize the nominal density function $f(\boldsymbol{x};\boldsymbol{\theta})$ by setting $\boldsymbol{\theta}$ equal to some $\boldsymbol{p}^{(0)} \in \boldsymbol{\Theta}$ for which $\mathbb{P}_{\boldsymbol{p}^{(0)}}(L(\boldsymbol{X}) \leq \gamma^\star) \neq 0$. Subsequently, for each iteration t (where $t > 1$), the importance sampling density from the previous iteration, $f(\boldsymbol{x},\boldsymbol{p}^{(t-1)})$, is adopted as the new nominal density. This updated nominal density is then utilized to derive the next importance sampling density, $f(\boldsymbol{x},\boldsymbol{p}^{(t)})$. The latter recursive relationship is established by iterative update of the parameter vector $\boldsymbol{p}^{(t)}$ as a solution to the following program:

$$\boldsymbol{p}^{(t)} = \underset{\boldsymbol{p}}{\operatorname{argmax}}\, \mathbb{E}_{\boldsymbol{p}^{(t-1)}}\left[\mathbb{1}_{\{L(\boldsymbol{X})\leq\gamma^{(t)}\}} \ln f(\boldsymbol{X};\boldsymbol{p})\right]. \tag{3.17}$$

Moreover, the update of $\boldsymbol{p}^{(t)}$ at each iteration t directly follows the update of another key parameter $\gamma^{(t)}$ as shown in Algorithm 3.1. In fact, $\boldsymbol{p}^{(t)}$ is updated by defining a set of indicator functions $\{\mathbb{1}_{\{L(\boldsymbol{X})\leq\gamma\}}\}$ for each level $\gamma^{(t)} \in \mathbb{R}^1$ such that $\{L(\boldsymbol{x}) \leq \gamma^{(t)}\} \neq \emptyset$. By adaptive update of the levels $\gamma^{(t)}$ and the parameter vector $\boldsymbol{p}^{(t)}$, the optimization algorithm is supposed to converge to the unique global minimum of L in (3.9) in finite number of iterations under some regularity conditions (Rubinstein, 1999). For such an optimization scheme to be successful, each update of the parameter vector $\boldsymbol{p}^{(t)}$ should lead the distribution $f(\cdot;\boldsymbol{p}^{(t)})$ to place a higher probability mass on the rare event $\{L(\boldsymbol{x}) \leq \gamma^{(t)}\}$ compared to $f(\cdot;\boldsymbol{p}^{(t-1)})$. Moreover, each update of $\gamma^{(t)}$ should ensure that the event $\{L(\boldsymbol{x}) \leq \gamma^{(t)}\}$ is a relatively rare event under $f(\cdot;\boldsymbol{p}^{(t)})$ to have high enough convergence speed, theoretically. To avoid misunderstanding, we have to note that $\gamma^{(t)}$ denotes the theoretical quantile and its update rule is determined by a fixed parameter ρ:

$$\gamma^{(t)} := \min\{s : \mathbb{P}_{\boldsymbol{p}^{(t-1)}}\Big(L(\boldsymbol{X}) \leq s\Big) \geq \rho\}, \tag{3.18}$$

and our discussion thus far has elaborated on ideal theoretical scenario depicted in Algorithm 3.1 below.

Algorithm 3.1 The CE method for discrete optimization with fixed ρ (deterministic version)

Initialize: Probability density function $f(\cdot;\boldsymbol{p}^{(0)})$ ensuring $\mathbb{P}_{\boldsymbol{p}^{(0)}}(L(\boldsymbol{x}) \leq \gamma^{\star}) \neq 0$, set the learning rate $\rho \in [0,1]$, and initialize iteration counter $t \leftarrow 1$.

1: **while** stopping criterion not met **do**

2: **Adaptive updating of** $\gamma^{(t)}$:

$$\gamma^{(t)} \leftarrow \min\Big\{s : \mathbb{P}_{\boldsymbol{p}^{(t-1)}}(L(\boldsymbol{X}) \leq s) \geq \rho\Big\}.$$

3: **Adaptive updating of** $\boldsymbol{p}^{(t)}$:

$$\boldsymbol{p}^{(t)} \leftarrow \underset{\boldsymbol{p}}{\operatorname{argmax}}\, \mathbb{E}_{\boldsymbol{p}^{(t-1)}}\Big[\mathbb{1}_{\{L(\boldsymbol{X})\leq\gamma^{(t)}\}} \ln f(\boldsymbol{X};\boldsymbol{p})\Big].$$

4: $t \leftarrow t+1$

5: **end while**

Evidently, the Algorithm 3.1 is of no practical use in finite sample setting. Therefore, we outline the stochastic version (in finite sample setting) of the generic CE method in Algorithm 6.4 (see Appendix 6.2.1). We avoid elaborating on that since this is not the major focus of our paper. Nonetheless, more information about the CE method and its

stochastic version can be found in (Rubinstein, 1997; 1999; de Boer et al., 2005).

The next section first deals with the problems when the generic CE algorithm is applied to the clustering problem. Next, we will suggest modifications of the canonical CE method and discuss in details the new proposed algorithm.

3.4 The Generic CE for Clustering Problem

This section first deals with the deficiencies of the canonical CE method within the clustering problem context and the solutions we propose. The diagram in Figure 3.1 displays the plan of this section. The modifications we propose in this section are the building blocks of the new algorithm that we propose to solve clustering problem. We call our new algorithm the CESBKM Algorithm (for Cross Entropy with Symmetry Breaking and K-means).

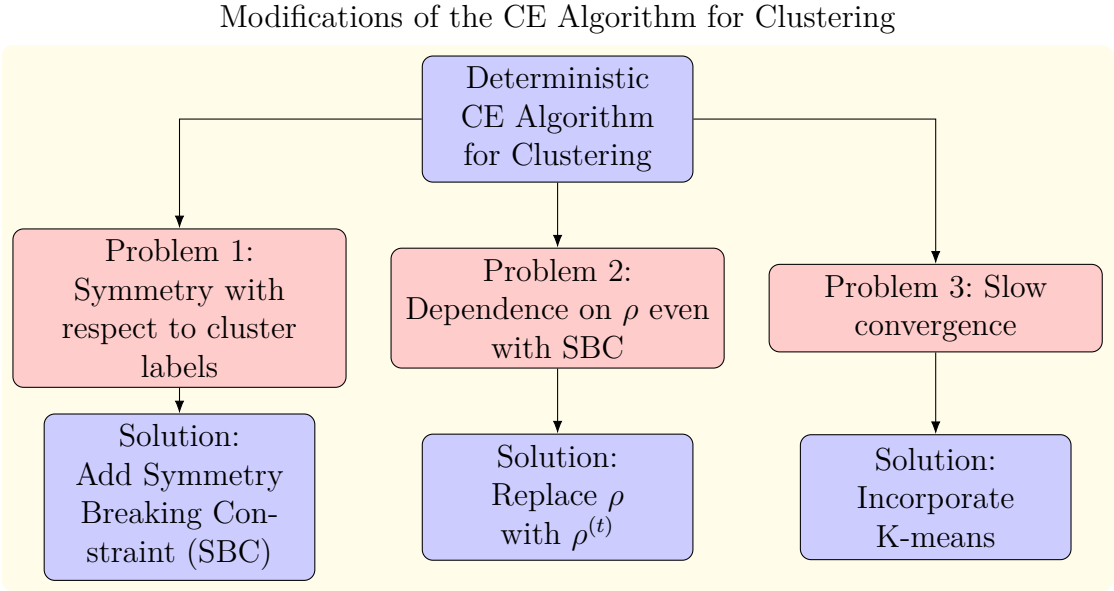


Figure 3.1 – Pitfalls Using The Deterministic CE for Clustering Problem and Their Suggested Solutions

3.4.1 Failure of the Generic CE Algorithm For Symmetric Functions

At this instance, we find it useful to shed light on the symmetry of different loss functions entertained in the context of the clustering problem with respect to cluster labels because the latter usually poses a challenge within numerical optimization context (Goldsztejn et al., 2015).

Definition 3.2. *Let $\mathcal{K}$ be an arbitrary set of labels, and let $\boldsymbol{x}$ be a vector whose elements are drawn from $\mathcal{K}$, denoted by $\boldsymbol{x} \in \mathcal{K}^n$. The function $L : \mathcal{K}^n \mapsto \mathbb{R}$ is said to exhibit symmetry with respect to the labels in $\mathcal{K}$ if, for any permutation σ from the symmetric group $\mathfrak{S}_K$ acting on the set $\mathcal{K}$, which induces a permutation of the vector $\boldsymbol{x}$ to $\boldsymbol{x}_\sigma$, the value of L remains invariant. That is, for every $\sigma \in \mathfrak{S}_K$, it holds that $L(\boldsymbol{x}) = L(\boldsymbol{x}_\sigma)$. Here, $\boldsymbol{x}_\sigma$ denotes the vector obtained by applying the permutation σ to each label in $\boldsymbol{x}$, effectively replacing each label $k \in \mathcal{K}$ with $\sigma(k)$.*

Indeed, the loss function $\mathscr{L}$, defined in (3.6), satisfies the requirements laid down in Definition 3.2 above. Let's make it more concrete with a simple example.

Example 1. *We use the same simple example as Kroese et al. (2007) (see Example 3.1) to show the symmetry of (3.6) with respect to the cluster labels. Let $n = 5$ and $K = 2$ and consider the cluster vector $\boldsymbol{x} = (1, 2, 2, 1, 2)$. Then, this corresponds to the partition $\{R_1, R_2\} = \{\{\boldsymbol{z}_1, \boldsymbol{z}_4\}, \{\boldsymbol{z}_2, \boldsymbol{z}_3, \boldsymbol{z}_5\}\}$ and the value of the loss function $\mathscr{L}$ is*

$$\mathscr{L}(\boldsymbol{x}) = \|\boldsymbol{z}_1 - \boldsymbol{c}_1^*\|^2 + \|\boldsymbol{z}_4 - \boldsymbol{c}_1^*\|^2 + \|\boldsymbol{z}_2 - \boldsymbol{c}_2^*\|^2 + \|\boldsymbol{z}_3 - \boldsymbol{c}_2^*\|^2 + \|\boldsymbol{z}_5 - \boldsymbol{c}_2^*\|^2, \tag{3.19}$$

with the corresponding centroids,

$$\boldsymbol{c}_1^* = \frac{\boldsymbol{z}_1 + \boldsymbol{z}_4}{2};\ \boldsymbol{c}_2^* = \frac{\boldsymbol{z}_2 + \boldsymbol{z}_3 + \boldsymbol{z}_5}{3}.$$

Now, consider the cluster vector $\boldsymbol{x}_{\boldsymbol{\sigma}} = (2, 1, 1, 2, 1)$ which is a swap permutation of $\boldsymbol{x}$. We can observe that,

$$\mathscr{L}(\boldsymbol{x}_\sigma) = \|\boldsymbol{z}_1 - \boldsymbol{c}_2^*\|^2 + \|\boldsymbol{z}_4 - \boldsymbol{c}_2^*\|^2 + \|\boldsymbol{z}_2 - \boldsymbol{c}_1^*\|^2 + \|\boldsymbol{z}_3 - \boldsymbol{c}_1^*\|^2 + \|\boldsymbol{z}_5 - \boldsymbol{c}_1^*\|^2,$$

with the corresponding centroids,

$$\boldsymbol{c}_1^* = \frac{\boldsymbol{z}_2 + \boldsymbol{z}_3 + \boldsymbol{z}_5}{3};\ \boldsymbol{c}_2^* = \frac{\boldsymbol{z}_1 + \boldsymbol{z}_4}{2}.$$

We have $\mathscr{L}(\boldsymbol{x}) = \mathscr{L}(\boldsymbol{x}_\sigma)$ *which illustrates that the loss function in clustering problems,* $\mathscr{L}$*, satisfies the symmetry property in Definition 1.*

Moreover, the assignment step of K-means corresponds to finding optimal partitions in (3.4) given centroids $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$. The solution consists in assigning each data point to the cluster whose centroid is closest. Mathematically, this means defining the sets $R_k = \{\boldsymbol{z} : \|\boldsymbol{z} - \boldsymbol{c}_k\|^2 \leq \|\boldsymbol{z} - \boldsymbol{c}_l\|^2, k \neq l\}$, for $k = 1, \cdots, K$. Given R is at optimal value then, it is easy to see that the function $\mathcal{L}$ in (3.5) is also symmetric with respect to cluster centroids, i.e. invariant under any permutation of the centroid vector. Let us explain it with a simple example.

Example 2. *Let* $n = 5$*,* $K = 2$ *and* $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$*. Moreover, suppose the optimal partition is* $\{R_1^\star, R_2^\star\} = \{\{\boldsymbol{z}_1, \boldsymbol{z}_3, \boldsymbol{z}_4\}, \{\boldsymbol{z}_2, \boldsymbol{z}_5\}\}$*. Then,*

$$\mathcal{L}(\boldsymbol{c}) = \|\boldsymbol{z}_1 - \boldsymbol{c}_1\|^2 + \|\boldsymbol{z}_3 - \boldsymbol{c}_1\|^2 + \|\boldsymbol{z}_4 - \boldsymbol{c}_1\|^2 + \|\boldsymbol{z}_2 - \boldsymbol{c}_2\|^2 + \|\boldsymbol{z}_5 - \boldsymbol{c}_2\|^2. \tag{3.20}$$

Because $K = 2$*, there is a single permutation* σ *which exchanges the positions of the elements of* $\boldsymbol{c}$*. Such a permutation by swap evidently leads to* $\boldsymbol{c}_\sigma = (\boldsymbol{c}_2, \boldsymbol{c}_1)$ *and to the corresponding partition* $\{R_2^\star, R_1^\star\} = \{\{\boldsymbol{z}_2, \boldsymbol{z}_5\}, \{\boldsymbol{z}_1, \boldsymbol{z}_3, \boldsymbol{z}_4\}\}$*. The loss function value at* $\boldsymbol{c}_\sigma$ *is*

$$\mathcal{L}(\boldsymbol{c}_\sigma) = \|\boldsymbol{z}_2 - \boldsymbol{c}_2\|^2 + \|\boldsymbol{z}_5 - \boldsymbol{c}_2\|^2 + \|\boldsymbol{z}_1 - \boldsymbol{c}_1\|^2 + \|\boldsymbol{z}_3 - \boldsymbol{c}_1\|^2 + \|\boldsymbol{z}_4 - \boldsymbol{c}_1\|^2, \tag{3.21}$$

which implies $\mathcal{L}(\boldsymbol{c}) = \mathcal{L}(\boldsymbol{c}_\sigma)$*.*

In the preceding examples, we explored the symmetry properties of clustering loss functions illustrated in equations (3.5) and (3.6), respectively, by assuming just two clusters.

This discussion can be naturally extended to the case of K clusters. Let the complete set of potential cluster vectors be denoted by $\mathscr{X} = \{1, 2, \ldots, K\}^n$. Then, the set of proper cluster vectors, similarly to the Definition 3.1, is defined as

$$\mathscr{X}_{\text{prop}} = \left\{ \boldsymbol{x} \in \mathscr{X} : \sum_{i=1}^{n} \mathbb{1}_{\{x_i = k\}} \neq 0, \forall k = 1, \ldots, K \right\}. \tag{3.22}$$

To compute the cardinality of $\mathscr{X}_{\text{prop}}$, one should use the inclusion-exclusion principle by considering the formula $K^n - \text{card}(A_1 \cup \ldots \cup A_K)$ where A_i is the set of vectors that do

not contain the i-th label. This gives

$$\begin{aligned} \operatorname{card}(\mathscr{X}_{\text{prop}}) &= K^n + \sum_{i=1}^{K} (-1)^i \binom{K}{i} (K-i)^n, \\ &= \sum_{i=0}^{K-1} (-1)^i \binom{K}{i} (K-i)^n. \end{aligned} \tag{3.23}$$

Since the function in (3.6) exhibits symmetry with respect to the cluster labels, i.e. $\mathscr{L}(\boldsymbol{x}) = \mathscr{L}(\boldsymbol{x}_\sigma)$ for every $\sigma \in \mathfrak{S}_K$, the set $\mathscr{X}_{\text{prop}}$ will be partitioned into equivalence classes with respect to the permutation of cluster labels. Specifically, we can represent $\mathscr{X}_{\text{prop}}$ as a union of $K!$ distinct subsets, $\mathscr{X}_{\text{prop}} = \bigcup_{\sigma \in \mathfrak{S}_K} \mathscr{X}_\sigma$, where each subset $\mathscr{X}_\sigma$ is mutually exclusive, i.e. satisfying $\mathscr{X}_\sigma \cap \mathscr{X}_\tau = \emptyset$ for any distinct permutations $\sigma, \tau \in \mathfrak{S}_K$.

In the subsequent theorem, we elucidate the significance of addressing symmetry in numerical optimization problems to ensure convergence. More precisely, we demonstrate that the deterministic version of the CE algorithm does not converge to the global minimum of the clustering loss function $\mathscr{L}$ which itself is symmetric. However, before the theorem statement and the associated proof we find it useful to define the convergence property of the CE method for discrete optimization problems.

Definition 3.3. *In the context of a discrete optimization problem, the deterministic version of the CE method delineated in Algorithm 3.1 is said to converge to the minimum of the function $\mathscr{L}$ denoted by $\gamma^\star = \min_{\boldsymbol{x}} \mathscr{L}(\boldsymbol{x})$, if at the final iteration $T^\star$, the optimal parameter vector $\boldsymbol{p}^{(T^\star)}$ is such that $\mathbb{P}_{\boldsymbol{p}^{(T^\star)}}(\mathscr{L}(\boldsymbol{X}) \leq \gamma^\star) = 1$.*

The above definition implies that if the function L possesses a unique global minimum and the CE method converges, then the vector of probabilities $\boldsymbol{p}_i^{(T^\star)}$ for the i-th data point will be 0 for $K-1$ clusters and 1 for the remaining one.

Remark 3.1. *Depending on the choice of algorithm's parameters, the deterministic version of the generic CE in Kroese et al. (2007) may not converge to the global optimum of (3.4) even in infinite number of iterations.*

Proof. We will give a generic example in which the convergence in the sense of Definition 3.3 is not reached. To solve combinatorial optimization problems, the CE distribution $f(\boldsymbol{X}; \boldsymbol{p})$ is typically chosen as a product of categorical distributions, i.e.

$$f(\boldsymbol{X}; \boldsymbol{p}) = \prod_{i=1}^{n} \operatorname{Cat}(\boldsymbol{X}_i; \boldsymbol{p}_i); \tag{3.24}$$

where $\boldsymbol{p}_i = (\boldsymbol{p}_{i1}, \ldots, \boldsymbol{p}_{iK})$ corresponds to the probabilities of obtaining the cluster label for the i-th data point, $P(\boldsymbol{X}_i = k) = \boldsymbol{p}_{ik}$.

If the inital values of these probabilities $\boldsymbol{p}^{(0)}$ are such that $\forall \boldsymbol{x} \in \{\boldsymbol{z} \in \mathscr{X}_{\text{prop}} : \mathscr{L}(\boldsymbol{z}) \leq \gamma\}$ for any $\gamma \geq \gamma^\star = \min_{\boldsymbol{x}} \mathscr{L}(\boldsymbol{x})$ and $\forall \sigma \in \mathfrak{S}_K$

$$P_{\boldsymbol{p}^{(0)}}(\boldsymbol{X} = \boldsymbol{x}) = P_{\boldsymbol{p}^{(0)}}(\boldsymbol{X} = \boldsymbol{x}_\sigma), \tag{3.25}$$

then, owing to the symmetry of the function $\mathscr{L}(\cdot)$ with respect to $K!$ permutations of its argument, we have

$$\boldsymbol{p}_{ik}^{(1)} = \frac{\mathbb{E}_{\boldsymbol{p}^{(0)}}\left[\mathbb{1}_{\{\mathscr{L}(\boldsymbol{X}) \leq \gamma\}} \mathbb{1}_{\{\boldsymbol{X}_i = k\}}\right]}{\mathbb{E}_{\boldsymbol{p}^{(0)}}\left[\mathbb{1}_{\{\mathscr{L}(\boldsymbol{X}) \leq \gamma\}}\right]} = \frac{1}{K}. \tag{3.26}$$

In particular, the condition (3.25) is satisfied when $\boldsymbol{p}_{ik}^{(0)} = 1/K$ for $i = 1, \ldots, n$ and $k = 1, \ldots, K$. Therefore, (3.26) holds true in this case. □

To overcome the possible absence of convergence of the CE method dealing with the optimization of symmetric functions, we propose to introduce a symmetry breaking constraint (SBC) using the lexicographical ordering of the clusters' centers.

Definition 3.4. *Given two vectors* $\mathbf{a} = (a_1, a_2, \ldots, a_d)$ *and* $\mathbf{b} = (b_1, b_2, \ldots, b_d)$ *in* d*-dimensional space, we say that* $\mathbf{a}$ *is lexicographically less than* $\mathbf{b}$ *(denoted as* $\mathbf{a} \prec \mathbf{b}$*) if there exists an index* k *(where* $1 \leq k \leq d$*) such that for all* $j < k$*,* $a_j = b_j$ *and* $a_k < b_k$*.*

In other words, this definition 3.4 say that $\mathbf{a} \prec \mathbf{b}$ lexicographically if, at the first position where the two vectors differ, the entry of $\mathbf{a}$ is smaller than the corresponding entry of $\mathbf{b}$.

Let us define $\mathscr{X}_{\text{break}}$, the set of $\boldsymbol{x} \in \mathscr{X}_{\text{prop}}$ whose clusters' centers $(c_1(\boldsymbol{x}), c_2(\boldsymbol{x}), \ldots, c_K(\boldsymbol{x}))$ are lexicographically ordered, meaning that

$$c_1(\boldsymbol{x}) \prec c_2(\boldsymbol{x}) \prec \ldots \prec c_K(\boldsymbol{x}). \tag{3.27}$$

Let us now consider the following loss function $\mathscr{L}(\cdot)$ restricted to $\mathscr{X}_{\text{break}}$: $\forall \boldsymbol{x} \in \mathscr{X}$, we set

$$\mathscr{L}^{lex}(\boldsymbol{x}) = \begin{cases} \mathscr{L}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathscr{X}_{\text{break}}, \\ +\infty & \text{otherwise.} \end{cases} \tag{3.28}$$

With the symmetry breaking constraint introduced above, the deterministic CE algorithm takes the form as presented in Algorithm 3.2.

Algorithm 3.2 Deterministic CE for Clustering Problem with Symmetry Breaking Constraint

Input: Dataset $\mathcal{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}$ in d-dimensional Euclidean space, number of clusters K.

Parameters: $f(\cdot; \boldsymbol{p}^{(0)})$ defined in (3.24) with $\boldsymbol{p}^{(0)}$, where $\boldsymbol{p}^{(0)}$ is an $n \times K$ matrix with each element $\boldsymbol{p}_{ik}^{(0)} (= \frac{1}{K}$ for $i = 1, \ldots, n$ and $k = 1, \ldots, K)$,
$\rho \in (0, 1)$, iteration counter $t = 1$.

1: **while** stopping criterion not met **do**
2: **Adaptively update** $\gamma^{(t)}$:
3: $\gamma^{(t)} \leftarrow \min\left\{s \mid \mathbb{P}_{\boldsymbol{p}^{(t-1)}}\left(\mathscr{L}^{lex}(\boldsymbol{X}) \leq s\right) \geq \rho\right\}$
4: **Adaptively update** $\boldsymbol{p}^{(t)}$:
5: $\boldsymbol{p}_{ik}^{(t)} \leftarrow \dfrac{\mathbb{E}_{\boldsymbol{p}^{(t-1)}}\left[\mathbb{1}_{\{\mathscr{L}^{lex}(\boldsymbol{X}) \leq \gamma^{(t)}\}} \mathbb{1}_{\{\boldsymbol{X}_i = k\}}\right]}{\mathbb{E}_{\boldsymbol{p}^{(t-1)}}\left[\mathbb{1}_{\{\mathscr{L}^{lex}(\boldsymbol{X}) \leq \gamma^{(t)}\}}\right]}$
6: Increment iteration counter: $t \leftarrow t + 1$
7: **end while**

Remark 3.2. *A possible procedure, as used in (de Boer et al., 2005), to partially remove symmetries consists of fixing the cluster label of the first element by setting* $\boldsymbol{p}_{11}^{(0)} = 1$ *in the CE categorical distribution. However, compared to our proposed method, this strategy is insufficient to remove all existing symmetries. Nevertheless, such a procedure could be used in addition to our method in order to more efficiently explore the space while satisfying the symmetry-breaking constraint.*

Remark 3.3. *Yet another possible way of symmetry breaking is to impose lexicographic ordering on the fitted values, rather than on the centroids. Let* $\hat{\boldsymbol{z}}_i(\boldsymbol{c}^*, \boldsymbol{x})$ *denote the fitted value for observation* i *under cluster vector* $\boldsymbol{x}$ *and optimal parameter vector* $\boldsymbol{c}^* = (\boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$. *Then for any observation* i*:*

$$\hat{\boldsymbol{z}}_i(\boldsymbol{c}^*, \boldsymbol{x}) = \begin{cases} \boldsymbol{c}_1^* \boldsymbol{y}_i & \text{if } x_i = 1 \\ \boldsymbol{c}_2^* \boldsymbol{y}_i & \text{if } x_i = 2 \end{cases}$$

For the swap permutation $\boldsymbol{x}_\sigma$ *with optimal parameter vector* $\boldsymbol{c}_\sigma^* = (\boldsymbol{c}_2^*, \boldsymbol{c}_1^*)$*:*

$$\hat{\boldsymbol{z}}_i(\boldsymbol{c}_\sigma^*, \boldsymbol{x}_\sigma) = \begin{cases} \boldsymbol{c}_2^* \boldsymbol{y}_i & \text{if } x_{\sigma i} = 1 \\ \boldsymbol{c}_1^* \boldsymbol{y}_i & \text{if } x_{\sigma i} = 2 \end{cases}$$

That is, if the fitted value for observation i *under* $\boldsymbol{x}$ *is* $\boldsymbol{c}_1^* \boldsymbol{y}_i$*, then under the swap permutation* $\boldsymbol{x}_\sigma$*, its fitted value becomes* $\boldsymbol{c}_2^* \boldsymbol{y}_i$*, and vice versa.*

In the next subsection, we use a simple example to illustrate that the CE algorithm converges when using a SBC but only for small enough values of the quantile rank parameter ρ. Unfortunately, the set of those optimal values of ρ (for which the algorithm converges) are not known in advance. Therefore, in the subsequent part of the paper we propose to introduce a geometric decay of ρ which ensures the convergence of the algorithm to the global optimum.

3.4.2 Empirical study of CE convergence

The purpose of this Section is to analyze the convergence behavior of the modified CE method for clustering problem outlined in Algorithm 3.2. Particularly, we use another simple example to demonstrate that although symmetry breaking constraints are useful, they do not always guarantee the convergence to the optimal solution. More precisely, the example shows that after posting symmetry breaking constraints the deterministic algorithm may converge to the global optimum given ρ is initialized to small enough value. The latter fact necessitates making some modifications to the ρ parameter. Therefore, we modify Algorithm 3.2 further by introducing an exponential decay such that

$$\rho^{(t)} = \rho \cdot e^{-q \cdot t},$$

where $\rho \in (0, 1)$ is the fixed quantile rank parameter, $q \geq 0$ is the decay rate, and t is the iteration counter. Note that, when $q = 0$, then $\rho^{(t)} = \rho$ which is the conventional fixed quantile rank parameter characteristic to the generic CE. As it is shown hereinfurther, by introduction of a geometric decay, we decrease the dependence of the theoretical convergence of the algorithm on the fixed value of ρ. Furthermore, within the framework of our investigation, the implementation of an exponentially decaying sequence $\rho^{(t)}$ ensures that the series $\{\gamma^{(t)}\}_{t \in \mathbb{N}}$ where $\gamma^{(t)} = F^{-1}(\rho^{(t)}; \boldsymbol{p}^{(t-1)})$ and F is the cdf of $\mathscr{L}(\boldsymbol{X})$, converges to the optimal solution within a finite number of iterations. Later in this paper we provide a rigorous demonstration affirming that for any given function $\boldsymbol{x} \mapsto \mathscr{L}(\boldsymbol{x})$ possessing a unique global minimum, an exponentially diminishing $\rho^{(t)}$ guarantees theoretical convergence of the CE in a finite number of steps. Moreover, we conjecture that there may be other advantages of this exponential decay formula for the stochastic version of the CE:

— By using the exponential decay with a value of q closer to 1, the CE algorithm may explore search space more thoroughly in initial iterations as opposed to default setting $\rho \in (0.01, 0.1)$ suggested in the related literature (de Boer et al., 2005).

Adopting an adaptive approach to determine $\rho^{(t)}$ may help to avoid premature convergence of the stochastic version of the CE algorithm and may increase the likelihood of not missing best solutions;

— This may increase the speed of convergence in some complex problems where fixed ρ leads small or no change in $\gamma^{(t)}$;

— It may be easy for the practitioner to adjust the algorithm to his/her computational budget and to the complexity of the optimization problem at hand by tuning the parameter b.

In Example 3 below we show that introducing geometric decay in parameter ρ guarantees the convergence of the algorithm with the symmetry breaking constraints discussed in Section 3.4.1.

Example 3. *For simplicity, suppose $n = 3$ and $K = 2$, then consider the complete set of potential cluster vectors $\mathscr{X} = \{1, 2, \ldots, K\}^n$ defined as:*

$$\mathscr{X} = \begin{bmatrix} 1 & 2 & 1 & 2 & 2 & 1 & 2 & 1 \\ 1 & 2 & 2 & 1 & 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 & 2 & 1 & 1 & 2 \end{bmatrix}.$$

As defined by (3.22), the set of proper cluster vectors is given by

$$\mathscr{X}_{prop} = \begin{bmatrix} 1 & 2 & 2 & 1 & 2 & 1 \\ 2 & 1 & 1 & 2 & 2 & 1 \\ 2 & 1 & 2 & 1 & 1 & 2 \end{bmatrix}.$$

As discussed before, the set $\mathscr{X}_{prop}$ contains $K!$ permutations for which the cost function is invariant. Moreover, we have defined $\mathscr{X}_{break}$, the set of $\boldsymbol{x} \in \mathscr{X}_{prop}$ whose clusters' centers $(c_1(\boldsymbol{x}), c_2(\boldsymbol{x}), \ldots, c_K(\boldsymbol{x}))$ are lexicographically ordered. To be more concrete, for cases in Panels (a) and (c) of Figure 3.2, $\mathscr{X}_{break}$ is defined as follows:

$$\mathscr{X}_{break} = \begin{array}{c} \begin{matrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \boldsymbol{x}_3 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix} \end{array}, \tag{3.29}$$

and for Panels (b), (d), (e) and (f),

$$\mathscr{X}_{break} = \begin{array}{c} \begin{matrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \boldsymbol{x}_3 \end{matrix} \\ \begin{bmatrix} 2 & 1 & 2 \\ 1 & 1 & 1 \\ 1 & 2 & 2 \end{bmatrix} \end{array}. \tag{3.30}$$

In this example, the symmetry breaking constraint is taken into account by the modification of the original cost function $\mathscr{L}$ to $\mathscr{L}^{lex}$ as defined in (3.28). As a consequence, for the two clusters' example, the quantile $\gamma^{(t)}$ and the probabilities of the sampling distribution of the CE are updated as follows:

$$\gamma^{(t)} = \min\{s : \mathbb{P}_{\boldsymbol{p}^{(t-1)}}\Big(\mathscr{L}^{lex}(\boldsymbol{X}) \leq s\Big) \geq \rho\}, \tag{3.31}$$

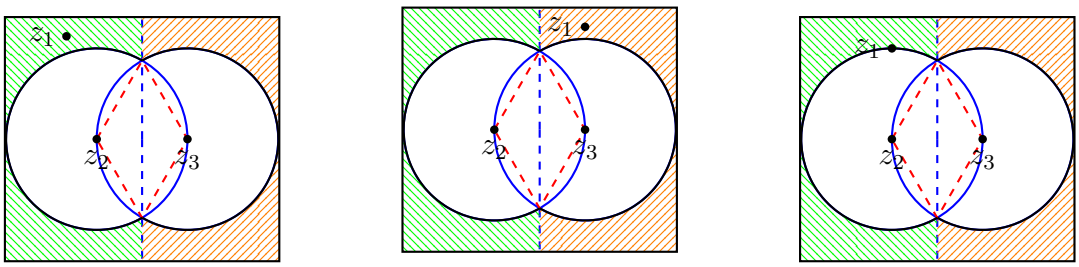
and

$$\boldsymbol{p}_i^{(t)} = \frac{\mathbb{E}_{\boldsymbol{p}^{(t-1)}}\left[\mathbb{1}_{\{\mathscr{L}^{lex}(\boldsymbol{X})\leq\gamma^{(t)}\}}\mathbb{1}_{\{\boldsymbol{X}_i=1\}}\right]}{\mathbb{E}_{\boldsymbol{p}^{(t-1)}}\left[\mathbb{1}_{\{\mathscr{L}^{lex}(\boldsymbol{X})\leq\gamma^{(t)}\}}\right]}, \quad \text{for} \quad i = 1, \ldots, n. \tag{3.32}$$

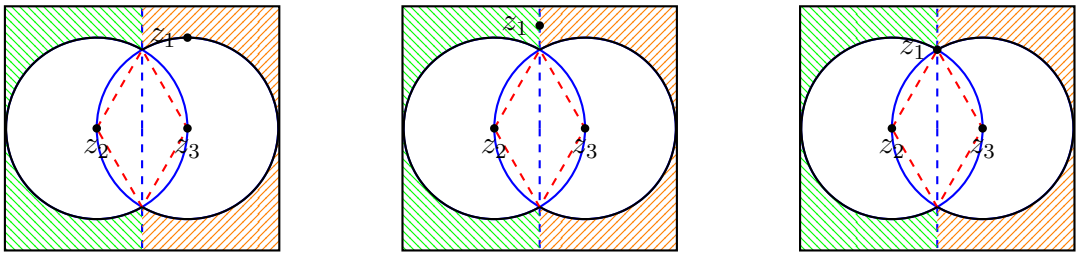
To perform the deterministic Algorithm 3.2, consider three points $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, and $\boldsymbol{z}_3$ in a two-dimensional Euclidean space, where $d(\omega, \omega')$ denotes the Euclidean distance between any two points ω and ω'. It can be inferred from Figure 3.2 that when $\boldsymbol{z}_1$ is positioned on the arcs highlighted by black color or on the exterior of either circles with centers $\boldsymbol{z}_2$ and $\boldsymbol{z}_3$, respectively, the distance $d(\boldsymbol{z}_2, \boldsymbol{z}_3)$ is either the minimum or equal to the minimum of the set $\{d(\boldsymbol{z}_1, \boldsymbol{z}_2), d(\boldsymbol{z}_1, \boldsymbol{z}_3), d(\boldsymbol{z}_2, \boldsymbol{z}_3)\}$. This relationship can be expressed as:

$$d(\boldsymbol{z}_2, \boldsymbol{z}_3) \leq \min\{d(\boldsymbol{z}_1, \boldsymbol{z}_2), d(\boldsymbol{z}_1, \boldsymbol{z}_3)\}. \tag{3.33}$$

However, if $\boldsymbol{z}_1$ is located within the interior of one of the two circles, the configuration of the points changes, rendering the previous inequality inapplicable. In this altered configuration, the distance from $\boldsymbol{z}_1$ to either $\boldsymbol{z}_2$ or $\boldsymbol{z}_3$ may become the shortest, which contradicts the initial relationship. Consequently, two new circles can be drawn with centers at the points corresponding to the minimum of the distances $\{d(\boldsymbol{z}_1, \boldsymbol{z}_2), d(\boldsymbol{z}_1, \boldsymbol{z}_3), d(\boldsymbol{z}_2, \boldsymbol{z}_3)\}$, and the analysis can proceed similarly. Therefore, without loss of generality, we can assume



Panel (a) Panel (b) Panel (c)



Panel (d) Panel (e) Panel (f)

Figure 3.2 – Comparative analysis of geometric configurations of data points. Panel (a) corresponds to $\mathscr{L}^{lex}(\boldsymbol{x}_1) < \mathscr{L}^{lex}(\boldsymbol{x}_2) < \mathscr{L}^{lex}(\boldsymbol{x}_3)$ and Panel (b) correspond to $\mathscr{L}^{lex}(\boldsymbol{x}_1) < \mathscr{L}^{lex}(\boldsymbol{x}_3) < \mathscr{L}^{lex}(\boldsymbol{x}_2)$. Panel (c) corresponds to $\mathscr{L}^{lex}(\boldsymbol{x}_1) = \mathscr{L}^{lex}(\boldsymbol{x}_2) < \mathscr{L}^{lex}(\boldsymbol{x}_3)$. Panel (d) corresponds to $\mathscr{L}^{lex}(\boldsymbol{x}_1) = \mathscr{L}^{lex}(\boldsymbol{x}_3) < \mathscr{L}^{lex}(\boldsymbol{x}_2)$. Panel (e) corresponds to $\mathscr{L}^{lex}(\boldsymbol{x}_1) < \mathscr{L}^{lex}(\boldsymbol{x}_2) = \mathscr{L}^{lex}(\boldsymbol{x}_3)$. Panel (f) corresponds to the trivial case $\mathscr{L}^{lex}(\boldsymbol{x}_1) = \mathscr{L}^{lex}(\boldsymbol{x}_2) = \mathscr{L}^{lex}(\boldsymbol{x}_3)$.

that the relationship holds and restrict our analysis to the cases where $\boldsymbol{z}_1$ is positioned on the exterior or on the arcs of the two circles highlighted by black color as depicted in the Figure 3.2.

Indeed, Panel (a)-(f) of Figure 3.2 describes the totality of all configurations corresponding to the relationship in (3.33). Below, we will apply Algorithm 3.2 with different values of quantile rank $\rho = 0.33, 0.5, 0.67, 1$, respectively, to five cases corresponding to Panel (a)-(e), and ignore the trivial case shown in Panel (f).

Panel (a). Let $C(\boldsymbol{z}_2, r)$ and $C(\boldsymbol{z}_3, r)$ be the two circles in Figure 3.2, with the same radius r and with centers at points $\boldsymbol{z}_2$ and $\boldsymbol{z}_3$, respectively. Evidently, positioning $\boldsymbol{z}_1$ anywhere on the green shaded area excluding the arc of the circle $C(\boldsymbol{z}_2, r)$ highlighted by black color and excluding the blue dashed line, will lead to the following:

$$\mathscr{L}^{lex}(\boldsymbol{x}_1) < \mathscr{L}^{lex}(\boldsymbol{x}_2) < \mathscr{L}^{lex}(\boldsymbol{x}_3),$$

where $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3 \in \mathscr{X}_{\text{break}}$ as defined in (3.30).

Without loss of generality, we set $\boldsymbol{z}_1 = (-0.5, 1.7)$, $\boldsymbol{z}_2 = (0, 0)$ and $\boldsymbol{z}_3 = (1.5, 0)$ to be consistent with the case in Panel (a), leading to $\mathscr{L}^{lex}(\boldsymbol{x}_1) = 1.06$, $\mathscr{L}^{lex}(\boldsymbol{x}_2) = 1.25$ and $\mathscr{L}^{lex}(\boldsymbol{x}_3) = 1.86$. Table 3.1 shows how $\gamma^{(t)}$ and $\boldsymbol{p}^{(t)}$ change over iterations of Algorithm 3.2 for different values of ρ parameter. According to these results, convergence as defined in 3.3 is achieved at $t = 2$ for $\rho \in (0, 0.33]$, and at $t = 3$ for $\rho \in (0.33, 0.5]$, and for $\rho > 0.5$ the algorithm does not converge.

Table 3.1 – Behavior of Algorithm 3.2 for the cases akin to Panel (a) of Figure 3.2 for different initializations of the quantile rank parameter ρ.

	$\rho = 0.33$		$\rho = 0.5$		$\rho = 0.67$		$\rho = 1$	
t	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$
1	(0.5, 0.5, 0.5)	1.86	(0.5, 0.5, 0.5)	1.86	(0.5, 0.5, 0.5)	1.86	(0.5, 0.5, 0.5)	1.86
2	(1, 0, 0)	1.06	(1, 0.5, 0)	1.25	(0.67, 0.67, 0)	1.86	(0.67, 0.67, 0)	1.86
3	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.67, 0)	1.25	(0.75, 0.75, 0)	1.86
4	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.67, 0)	1.25	(0.8, 0.8, 0)	1.86
5	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.67, 0)	1.25	(0.83, 0.83, 0)	1.86
6	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.67, 0)	1.25	(0.86, 0.86, 0)	1.86
⋮		⋮		⋮		⋮		⋮
∞	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.67, 0)	1.25	(1, 1, 0)	1.25

On the other hand, adopting varying quantile rank rule forces the algorithm to con-

verge to the global minimum for all initializations. The results in Table 3.2 show that when $\rho^{(t)} = q^t$ for all $q \in (0, 1)$, Algorithm 3.2 reaches the global minimum. We initialize $q = 0.33, 0.5, 0.97$ and 0.9, respectively, as the first three are critical points for the fixed ρ case as suggested by the results in Table 3.1.

Table 3.2 – Behavior of Algorithm 3.2 with geometric decay in the quantile rank parameter $\rho^{(t)}$ for the cases akin to Panel (a) of Figure 3.2 for different initializations of q.

	$q = 0.33$		$q = 0.5$		$q = 0.67$		$q = 0.9$	
t	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$
1	(0.5, 0.5, 0.5)	1.86	(0.5, 0.5, 0.5)	1.86	(0.5, 0.5, 0.5)	1.86	(0.5, 0.5, 0.5)	1.86
2	(1, 0, 0)	1.06	(1, 0.5, 0)	1.25	(0.67, 0.67, 0)	1.86	(0.67, 0.67, 0)	1.86
3	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.67, 0)	1.25	(0.75, 0.75, 0)	1.86
4	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.75, 0)	1.25
5	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.75, 0)	1.25
6	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.75, 0)	1.25
7	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.75, 0)	1.25
8	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.75, 0)	1.25
9	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.75, 0)	1.25
10	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.75, 0)	1.25
11	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.75, 0)	1.25
12	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.75, 0)	1.25
13	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.75, 0)	1.25
14	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.75, 0)	1.25
15	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06

Panel (b). In this second case, the point $\boldsymbol{z}_1$ could lie at any part of the non-shaded area to the right of the blue dashed line excluding the circle $C(\boldsymbol{z}_3, r)$ and the blue dashed line. Panel (b) shows just one of the infinitely many possibilities of the Case 2 which arises whenever

$$\mathscr{L}^{lex}(\boldsymbol{x}_1) < \mathscr{L}^{lex}(\boldsymbol{x}_3) < \mathscr{L}^{lex}(\boldsymbol{x}_2). \tag{3.34}$$

Without loss of generality, we set $\boldsymbol{z}_1 = (1.5, 1.7)$, $\boldsymbol{z}_2 = (0, 0)$ and $\boldsymbol{z}_3 = (1.5, 0)$ to be consistent with the case in Panel (b), leading to $\mathscr{L}^{lex}(\boldsymbol{x}_1) = 1.06$, $\mathscr{L}^{lex}(\boldsymbol{x}_2) = 1.60$ and $\mathscr{L}^{lex}(\boldsymbol{x}_3) = 1.20$.

As in the previous example, as shown in Table 3.3, the convergence of Algorithm 3.2 depends on the value ρ. In Table 3.4, we show how the CE method reaches global optimum when the parameter $q = 0.33, 0.5, 0.67, 0.9$, respectively. In fact, this is not surprising because the exponential decay forces the $\rho^{(t)}$ to settle within $(0, 0.5]$ exponentially fast and depending on initialization of the parameter q. Evidently, the same property will hold

Table 3.3 – Behavior of Algorithm 3.2 for the cases akin to Panel (b) of Figure 3.2 for different initializations of the quantile rank parameter ρ.

	$\rho = 0.33$		$\rho = 0.5$		$\rho = 0.67$		$\rho = 1$	
t	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$
1	(0.5, 0.5, 0.5)	1.60	(0.5, 0.5, 0.5)	1.60	(0.5, 0.5, 0.5)	1.60	(0.5, 0.5, 0.5)	1.60
2	(0, 1, 1)	1.06	(0, 1, 0.5)	1.20	(0.33, 1, 0.33)	1.60	(0.33, 1, 0.33)	1.60
3	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.33)	1.20	(0.25, 1, 0.25)	1.60
4	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.33)	1.20	(0.2, 1, 0.2)	1.60
5	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.33)	1.20	(0.17, 1, 0.17)	1.60
6	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.33)	1.20	(0.14, 1, 0.14)	1.60
⋮		⋮		⋮		⋮		⋮
∞	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.33)	1.20	(0, 1, 0)	1.20

for all other cases in Figure 3.2 and therefore, we do not report results of Algorithm 3.2 for the rest 3 cases.

Table 3.4 – Behavior of Algorithm 3.2 with geometric decay in the quantile rank parameter $\rho^{(t)}$ for the cases akin to Panel (b) of Figure 3.2 for different initializations of q.

	$q = 0.33$		$q = 0.5$		$q = 0.67$		$q = 0.9$	
t	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$
1	(0.5, 0.5, 0.5)	1.60	(0.5, 0.5, 0.5)	1.60	(0.5, 0.5, 0.5)	1.60	(0.5, 0.5, 0.5)	1.60
2	(0, 1, 1)	1.06	(0, 1, 0.5)	1.20	(0.33, 1, 0.33)	1.60	(0.33, 1, 0.33)	1.60
3	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.33)	1.20	(0.25, 1, 0.25)	1.60
4	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.25)	1.20
5	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.25)	1.20
6	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.25)	1.20
7	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.25)	1.20
8	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.25)	1.20
9	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.25)	1.20
10	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.25)	1.20
11	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.25)	1.20
12	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.25)	1.20
13	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.25)	1.20
14	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 0.25)	1.20
15	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06

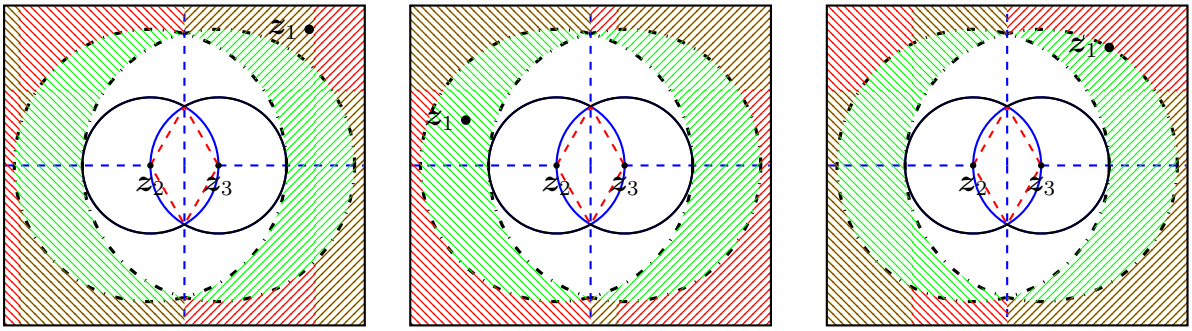
This simple example empirically shows that the convergence of Algorithm 3.2 for the clustering problem can be achieved for all cases discussed when symmetry breaking constraint is used and when the parameter ρ is correctly chosen i.e. the convergence is heavily dependent on the value of ρ. In other terms, when ρ is not sufficiently small then, convergence of the CE algorithm in the context of the symmetric functions may not be achieved even in the presence of symmetry breaking constraints. We have already

made a suggestion in this section to tackle this problem. Let's assume, for achieving convergence, the desirable interval for ρ is $(0, \epsilon)$ where $0 < \epsilon < 1$. Instead of fixing ρ we can introduce a geometrically decaying structure to it, such as $\rho^{(t)} = \rho \cdot e^{-qt}$ where $q \in [0, 1]$. Interestingly, this kind of law of motion for the quantile rank parameter encapsulates the fixed case as well: when $q = 0$ then, $\rho^{(t)} = \rho$, and when $0 < q \leq 1$ it assumes geometric decay . Undoubtedly, such a geometrically decreasing structure of $\rho^{(t)}$ (with $0 < q \leq 1$) ensures that $\rho^{(t)}$ settles within $(0, \epsilon)$ interval exponentially fast and makes the algorithm to converge for all cases discussed above, given symmetry breaking constraint is used. Adopting a varying quantile rank rule alleviates the burden of searching for a desirable ρ on practitioners and may generate other desirable finite sample properties conjectured previously. However, we have to start from a high value of ρ closer to one because the critical interval for the fixed ρ case is never known in advance, and more importantly, we want to avoid premature convergence of the stochastic version of the CE algorithm. Nevertheless, initializing ρ to a large and q to a relatively small value may decrease the speed of convergence significantly if the critical interval of ρ for convergence is very close to zero. To tackle this problem, in the next subsection, we suggest to incorporate K-means algorithm within CE method along with symmetry breaking constraint and geometrically decreasing quantile rank parameter.

3.4.3 Incorporating K-Means Algorithm Into CE Method

In this section, we propose to integrate K-means algorithm within the CE method. The reason to fuse the two procedures is to take advantage of good properties of both. More precisely, the CE method is known to converge to the global optimum with high probability under regularity conditions, as demonstrated in (Costa et al., 2007), while the K-means algorithm is guaranteed to converge only to a local optimum, as shown in Theorems 3.1 and 3.3. Nonetheless, the K-means algorithm is recognized as a computationally efficient heuristic for solving the clustering problem in literature (Ahmed et al., 2020). Moreover, numerical analyses have shown that the K-means method can exhibit faster convergence speeds than the CE method (Kroese et al., 2007). The simple example below illustrates empirically that the incorporation of the K-means procedure within the CE method can effectively reduce the number of iterations required for convergence under certain conditions.

Example 4. *Let us take three special cases from Example 3 as depicted in Figure 3.3.*



Panel (a) Panel (b) Panel (c)

Figure 3.3 – The Behavior of K-means algorithm for different configurations of three points.

Interestingly, incorporating K-means increases the speed of convergence for all $\rho \in (0, 1)$ values in this specific example and the convergence is achieved in just 2 iterations for $\rho \in (0, 0.33)$. Moreover, Panel (b) and (c) of Figure 3.3 show other cases for which K-means algorithm brings about significant efficiency gains in terms of the convergence behavior of Algorithm. Indeed, incorporating K-means algorithm within the CE procedure both decreases the dependence of the CE method from initialization of ρ and increases the speed of convergence. The K-means procedure reduces the dependence of the CE method from the initialization of parameter ρ because we can see from Table 3.5 that the convergence is achieved in 2 iterations for all $\rho \in (0.5, 1)$ even though it was not possible in similar cases in Example 3 in the absence of varying ρ and K-means. Moreover, in Example 3, despite achieving convergence by using $\rho^{(t)}$ it sometimes took more than 10 iterations to converge. However, incorporating K-means algorithm within CE led to convergence in maximum three iterations even while keeping ρ fixed in any value within $(0, 1]$ interval. This finding may suggest to forget about the varying $\rho^{(t)}$ and just incorporate K-means procedure within CE method. Unfortunately, a careful examination of Figure 3.3 reveals that the K-means algorithm is helpful only when $\boldsymbol{z}_1$ is positioned on the green and red shaded areas. Therefore, varying $\rho^{(t)}$ still preserves its legitimacy in ensuring convergence for all $q \in (0, 1)$. The reason to have two different shaded regions, red and green, respectively, has a meaning. When $\boldsymbol{z}_1$ falls within red shaded area as in Panel (a) (where $\boldsymbol{z}_1 = (3.5, 3)$, $\boldsymbol{z}_2 = (0, 0)$ and $\boldsymbol{z}_3 = (1.5, 0)$), convergence at $t = 2$ is guaranteed because K-means procedure converts all cluster vectors to the one at which the function $\mathscr{L}^{lex}$ attains global minimum. This is due to the fact that for all $\boldsymbol{z}_1$ in the red shaded area, $d(\boldsymbol{z}_1, \boldsymbol{z}_3) > 2d(\boldsymbol{z}_2, \boldsymbol{z}_3)$

Table 3.5 – Behavior of Algorithm 3.2 for the cases akin to Panel (a) of Figure 3.3 when integrated with K-means algorithm.

	$\rho = 0.33$		$\rho = 0.5$		$\rho = 0.67$		$\rho = 1$	
t	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$
1	(0.5, 0.5, 0.5)	3.26	(0.5, 0.5, 0.5)	3.26	(0.5, 0.5, 0.5)	3.26	(0.5, 0.5, 0.5)	3.26
2	(0, 1, 1)	1.06	(0, 1, 1)	2.55	(0, 1, 1)	3.26	(0, 1, 1)	3.26
3	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06
4	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06
5	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06
6	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06

and $d(\boldsymbol{z}_1, \boldsymbol{z}_2) > 2d(\boldsymbol{z}_2, \boldsymbol{z}_3)$ where d is an Euclidean distance as mentioned previously. Therefore, both sub-optimal partitions can be enhanced by K-means algorithm. On the other hand, when $\boldsymbol{z}_1$ lies within green shaded area including the border with the red shaded area then, only one of the sub-optimal vectors can be improved by K-means method. For example, in Panel (b) of Figure 3.3, $\frac{d(\boldsymbol{z}_1,\boldsymbol{z}_2)}{2} \leq d(\boldsymbol{z}_2, \boldsymbol{z}_3)$; thus, in the sub-optimal partition where $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ are assigned to the same cluster R_1, the points $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ are individually closer to the centroid of that cluster $\boldsymbol{c}_1 = \frac{\boldsymbol{z}_1+\boldsymbol{z}_2}{2}$ in terms of the Euclidean distance than to $\boldsymbol{c}_2$. Therefore, the assignment stage of the K-means algorithm will not alter the previous cluster assignment. However, by visual inspection of Panel (b) we can observe that $\frac{d(\boldsymbol{z}_1,\boldsymbol{z}_3)}{2} > 2d(\boldsymbol{z}_2, \boldsymbol{z}_3)$. Therefore, the K-means algorithm will assign $\boldsymbol{z}_2$ and $\boldsymbol{z}_3$ to the same, and $\boldsymbol{z}_1$ to the other cluster which coincides with the partition at the global optimum. Unfortunately, for all other positions of $\boldsymbol{z}_1$ excluding the shaded regions and the interiors of $C(\boldsymbol{z}_2, r)$ and $C(\boldsymbol{z}_2, r)$, respectively, the K-means algorithm is not useful. The convergence behavior of CE method for the case in Panel (b) is given in Table 3.6 below. The corresponding coordinates of the points are: $\boldsymbol{z}_1 = (-2, 1)$, $\boldsymbol{z}_2 = (0, 0)$ and $\boldsymbol{z}_3 = (1.5, 0)$.

Table 3.6 – Behavior of Algorithm 3.2 for the cases akin to Panel (b) of Figure 3.3 when integrated with K-means algorithm.

	$\rho = 0.33$		$\rho = 0.5$		$\rho = 0.67$		$\rho = 1$	
t	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$
1	(0.5, 0.5, 0.5)	2.57	(0.5, 0.5, 0.5)	2.57	(0.5, 0.5, 0.5)	2.57	(0.5, 0.5, 0.5)	2.57
2	(1, 0, 0)	1.06	(1, 0.5, 0)	1.58	(1, 0.33, 0)	2.57	(1, 0.33, 0)	2.57
3	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.2, 0)	1.58
4	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0.11, 0)	1.58
5	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06
6	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06	(1, 0, 0)	1.06

One last case is about positioning $\boldsymbol{z}_1$ on the boundary of $C(\boldsymbol{z}_3, 2r)$ and is depited in Panel (c) of Figure 3.3. The coordinates of points are as follows: $\boldsymbol{z}_1 = (3, 2.6)$, $\boldsymbol{z}_2 = (0, 0)$ and $\boldsymbol{z}_3 = (1.5, 0)$. Table 3.7 summarizes the convergence behavior of the CE algorithm for ρ fixed at four different values.

Table 3.7 – Behavior of Algorithm 3.2 for the cases akin to Panel (c) of Figure 3.3 when integrated with K-means algorithm.

	$\rho = 0.33$		$\rho = 0.5$		$\rho = 0.67$		$\rho = 1$	
t	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$	$\boldsymbol{p}^{(t-1)}$	$\gamma^{(t)}$
1	(0.5, 0.5, 0.5)	2.81	(0.5, 0.5, 0.5)	2.81	(0.5, 0.5, 0.5)	2.81	(0.5, 0.5, 0.5)	2.81
2	(0, 1, 1)	1.06	(0, 1, 1)	2.12	(0, 1, 1)	2.81	(0, 1, 1)	2.81
3	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06
4	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06
5	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06
6	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06	(0, 1, 1)	1.06

All three modifications of the canonical CE algorithm that we proposed to this point constitute our CESBKM algorithm for clustering problem. In the next section, we will present both stochastic and deterministic versions of the algorithm, provide useful proofs for convergence of the deterministic version and finally, apply it to real data.

3.5 CESBKM Algorithm and Mathematical Properties

Here, we first describe our novel algorithm and then we study its mathematical properties.

3.5.1 Algorithm Description

Throughout Section 3.4, we have discussed the challenges of the deterministic version of the generic CE algorithm while applying it to the clustering problem. Furthermore, we proposed modifying the generic CE to guarantee convergence in the context of the clustering problem at a reasonable speed All the suggested modifications to the deterministic version of the CE method are what constitute our novel CESBKM algorithm for the clustering problem introduced in Section 3.2. The deterministic version of the CESBKM procedure is outlined in Algorithm 3.3.

Definition 3.5. *Define the following operator* $\varphi : \boldsymbol{x} \mapsto \varphi(\boldsymbol{x})$ *related to the K-means principle with symmetry contraint which is computed as follows*

1. *Update centroids using* $\boldsymbol{x}$,
2. *Assign new cluster vector* $\boldsymbol{x}'$ *based on updated centroids,*
3. *Return the vector* $\varphi(\boldsymbol{x}) = \boldsymbol{x}'_\sigma$, *where the permutation* $\sigma \in \mathfrak{S}_K$ *leads to associated lexicographically ordered centroids.*

Algorithm 3.3 The CESBKM Algorithm for Clustering Problem (Deterministic Version)

Input: Dataset $\mathcal{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}$ in d-dimensional Euclidean space, number of clusters K.

Parameters: $f(\cdot; \boldsymbol{p}^{(0)})$ defined in (3.24) with $\boldsymbol{p}^{(0)}$, where $\boldsymbol{p}^{(0)}$ is an $n \times K$ matrix with each element $\boldsymbol{p}_{ik}^{(0)} (= \frac{1}{K}$ for $i = 1, \ldots, n$ and $k = 1, \ldots, K)$,
decay rate $q \geq 0$, iteration counter $t = 1$, $\rho \in [0, 1]$.

Output: Optimal parameter array $\boldsymbol{p}^{(T^\star)}$ for which $\mathbb{P}_{\boldsymbol{p}^{(T^\star)}}(\mathscr{L}^{lex}(\varphi(\boldsymbol{X})) \leq \gamma^\star) = 1$.

1: **while** stopping criterion not met **do**
2: **Adaptively update** $\rho^{(t)}$ and $\tilde{\gamma}^{(t)}$:
3: $\rho^{(t)} \leftarrow \rho \cdot e^{-qt}$ and $\tilde{\gamma}^{(t)} \leftarrow \min\left\{s : \mathbb{P}_{\boldsymbol{p}^{(t-1)}}(\mathscr{L}^{lex}(\varphi(\boldsymbol{X})) \leq s) \geq \rho^{(t)}\right\}$
4: **Adaptively update** $\boldsymbol{p}^{(t)}$:
5: $\boldsymbol{p}_{ik}^{(t)} \leftarrow \dfrac{\mathbb{E}_{\boldsymbol{p}^{(t-1)}}\left[\mathbb{1}_{\{\mathscr{L}^{lex}(\varphi(\boldsymbol{X})) \leq \tilde{\gamma}^{(t)}\}} \mathbb{1}_{\{\boldsymbol{X}_i = k\}}\right]}{\mathbb{E}_{\boldsymbol{p}^{(t-1)}}\left[\mathbb{1}_{\{\mathscr{L}^{lex}(\varphi(\boldsymbol{X})) \leq \tilde{\gamma}^{(t)}\}}\right]}$
6: Increment iteration counter: $t \leftarrow t + 1$
7: **end while**

In Definition 3.3, we emphasized the sufficient condition to guarantee convergence of the algorithm, which is $\mathbb{P}_{\boldsymbol{p}^{(T^\star)}}(\mathscr{L}(\boldsymbol{x}) \leq \gamma^\star) = 1$ at final iteration $T^\star$. However, we have not discussed this condition thoroughly. Interestingly, the authors of the CE method propose to stop iterating when either $\boldsymbol{p}^{(t)}$ or $\gamma^{(t)}$ enter a cycle of length 1 (Rubinstein, 1999). Such a stopping criterion may be useful in practical applications, but the stabilization of the mentioned parameters are not sufficient for theoretical convergence we described. Indeed, as discussed in Remark 3.1, the parameters $\boldsymbol{p}^{(t)}$ are already stabilized at the second time step, see (3.26).

The convergence study of the deterministic version of the CESBKM enhances our understanding of the method in its deterministic version and also its stochastic one. Indeed, the stochastic version of the CESBKM asymptotically leads to the deterministic one. Moreover, the study of the deterministic version of the CE method suggests that the

stochastic version of it for clustering problems may escape from a stationary point (as in Remark 3.1) and converge due to the random fluctuations in Monte Carlo sampling.

More importantly, the stochastic version of the CESBKM method, described in Algorithm 3.4, performs at least as good as the stochastic version of the generic CE for clustering as empirically shown in the numerical experiments.

Algorithm 3.4 The Stochastic CESBKM Hybrid Algorithm (Minimization)

Input: Dataset $\mathcal{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}$ in d-dimensional Euclidean space, number of clusters K.
Parameters: $f(\cdot; \boldsymbol{p}^{(0)})$ defined in (3.24) with $\boldsymbol{p}^{(0)}$, where $\boldsymbol{p}^{(0)}$ is an $n \times K$ matrix with each element $\boldsymbol{p}_{ik}^{(0)} (= \frac{1}{K}$ for $i = 1, \ldots, n$ and $k = 1, \ldots, K)$,
decay rate $q \geq 0$, iteration counter $t = 1$, $\rho \in [0, 1]$.
Output: Optimal parameter array $\boldsymbol{p}^{(T^\star)}$ for which $\mathbb{P}_{\boldsymbol{p}^{(T^\star)}}(\mathscr{L}^{lex}(\varphi(\boldsymbol{X})) \leq \gamma^\star) = 1$.

1: **while** Stopping Criterion not met **do**
2: **Adaptively update** $\rho^{(t)}$:
3: $\rho^{(t)} \leftarrow \rho \cdot e^{-qt}$
4: **Adaptively update** $\hat{\hat{\gamma}}^{(t)}$:
5: **for** $l = 1$ to N **do**
6: Draw sample $\boldsymbol{X}_l \sim f(\cdot; \boldsymbol{p}^{(t-1)})$ and evaluate $\ell_l = \mathscr{L}^{lex}(\varphi(\boldsymbol{X}_l))$
7: **end for**
8: Get order statistic for $\mathscr{L}^{lex}$: $\ell_{(1)} \leq \cdots \leq \ell_{(N)}$ and set $\hat{\hat{\gamma}}^{(t)} \leftarrow \ell_{(\lceil N\rho^{(t)} \rceil)}$
9: **Update** $\boldsymbol{p}^{(t)}$:
10: $\boldsymbol{p}_{ik}^{(t)} \leftarrow \dfrac{\sum_{j=1}^{N} \mathbb{1}_{\{\mathscr{L}^{lex}(\varphi(\boldsymbol{X}_j)) \leq \hat{\hat{\gamma}}^{(t)}\}} \mathbb{1}_{\{\boldsymbol{X}_{ji}=k\}}}{\sum_{j=1}^{N} \mathbb{1}_{\{\mathscr{L}^{lex}(\varphi(\boldsymbol{X}_j)) \leq \hat{\hat{\gamma}}^{(t)}\}}}$
11: Increment iteration counter: $t \leftarrow t + 1$.
12: **end while**

In the subsequent part of the paper, we study the convergence of the deterministic version of the CESBKM method.

3.5.2 Properties and Convergence of the CESBKM Algorithm

To demonstrate the convergence of the deterministic and stochastic versions of the CESBKM procedure, we first show the monotonicity of the quantile sequence $\{\gamma^{(t)}\}_{t \in \mathbb{N}}$ generated by the generic CE method under much milder conditions than present in the related literature (Lieber, 1998).

Indeed, the monotonicity of the quantile sequence is an important pre-condition for convergence of the CE method in a finite number of iterations (Lieber, 1998; Rubinstein,

1999). The proof for the quantile sequence generated by the CE algorithm will be useful to prove the convergence of the CESBKM algorithm in finite number of iterations.

Theorem 3.4. *Let $L : \boldsymbol{x} \mapsto \mathbb{R}^1$ be a function. Consider that $\mathcal{A} = \{\boldsymbol{x} : L(\boldsymbol{x}) \leq \gamma\} \neq \emptyset$ for a given threshold value γ. Suppose further that for any fixed $\boldsymbol{x} \in \mathcal{A}$, the function $\boldsymbol{p} \mapsto f(\boldsymbol{x}; \boldsymbol{p})$, $\boldsymbol{p} \in \mathcal{V}$ is smooth (continuous and differentiable). Let $(\boldsymbol{p}, \tilde{\boldsymbol{p}}) \in \mathcal{V} \times \mathcal{V}$. If*

$$-\infty < \sum_{\boldsymbol{x}:L(\boldsymbol{x})\leq\gamma} f(\boldsymbol{x}; \tilde{\boldsymbol{p}}) \log f(\boldsymbol{x}; \tilde{\boldsymbol{p}}) \leq \sum_{\boldsymbol{x}:L(\boldsymbol{x})\leq\gamma} f(\boldsymbol{x}; \tilde{\boldsymbol{p}}) \log f(\boldsymbol{x}; \boldsymbol{p}). \tag{3.35}$$

Then,

$$\mathbb{P}_{\boldsymbol{p}}(L(\boldsymbol{X}) \geq \gamma) \geq \mathbb{P}_{\tilde{\boldsymbol{p}}}(L(\boldsymbol{X}) \geq \gamma). \tag{3.36}$$

Moreover, if there is a strict inequality in (3.35) then, there is a strict inequality in (3.36).

Proof. By the hypothesis of the theorem,

$$\sum_{\boldsymbol{x}:L(\boldsymbol{x})\leq\gamma} f(\boldsymbol{x}; \tilde{\boldsymbol{p}}) \log f(\boldsymbol{x}; \boldsymbol{p}) \geq \sum_{\boldsymbol{x}:L(\boldsymbol{x})\leq\gamma} f(\boldsymbol{x}; \tilde{\boldsymbol{p}}) \log f(\boldsymbol{x}; \tilde{\boldsymbol{p}}).$$

This implies that,

$$\sum_{\boldsymbol{x}:L(\boldsymbol{x})\leq\gamma} f(\boldsymbol{x}; \tilde{\boldsymbol{p}}) \log \frac{f(\boldsymbol{x}; \boldsymbol{p})}{f(\boldsymbol{x}; \tilde{\boldsymbol{p}})} \geq 0. \tag{3.37}$$

By (3.35), we have, $f(\boldsymbol{x}; \boldsymbol{p}) > 0$ and $f(\boldsymbol{x}; \tilde{\boldsymbol{p}}) > 0$ over $\{\boldsymbol{x} : L(\boldsymbol{x}) \leq \gamma\}$. Using the inequality $\log(\omega) \leq \omega - 1$, $\forall \omega \geq 0$, from (3.37) we get,

$$\sum_{\boldsymbol{x}:L(\boldsymbol{x})\leq\gamma} f(\boldsymbol{x}; \tilde{\boldsymbol{p}}) \Big[\frac{f(\boldsymbol{x}; \boldsymbol{p})}{f(\boldsymbol{x}; \tilde{\boldsymbol{p}})} - 1\Big] \geq 0.$$

The latter gives,

$$\mathbb{P}_{\boldsymbol{p}}(L(\boldsymbol{X}) \leq \gamma) \geq \mathbb{P}_{\tilde{\boldsymbol{p}}}(L(\boldsymbol{X}) \leq \gamma),$$

which completes the proof of (3.36). The second assertion of the theorem is obvious. □

Using the result of Theorem 3.4 we can establish that the sequence $\{\gamma^{(t)}\}_{t\in\mathbb{N}}$ is non-increasing.

Theorem 3.5. *Let $\boldsymbol{p}^{(t)}$ be the solution to (3.17) and $\boldsymbol{X} \sim f(\cdot; \boldsymbol{p}^{(t-1)})$ at iteration t of the CE in Algorithm 3.1. Then, for $0 < \rho \leq 1$, the sequence,*

$$\gamma^{(t)} := \min\{s : \mathbb{P}_{\boldsymbol{p}^{(t-1)}}(L(\boldsymbol{X}) \leq s) \geq \rho\}, \; t \in \mathbb{N}, \tag{3.38}$$

is non-increasing.

Proof. Note that, if the parameter $\boldsymbol{p}^{(t)}$ is the maximizer of (3.17), then, for $\gamma^{(t)}$ corresponding to a fixed $\rho \in (0,1)$ in (3.38),

$$\mathbb{P}_{\boldsymbol{p}^{(t)}}(L(\boldsymbol{x}) \leq \gamma^{(t)}) \geq \mathbb{P}_{\boldsymbol{p}^{(t-1)}}(L(\boldsymbol{x}) \leq \gamma^{(t)}) = \rho.$$

In turn, this implies that

$$\begin{aligned} \min\{s : \mathbb{P}_{\boldsymbol{p}^{(t)}}(L(\boldsymbol{X}) \leq s) \geq \rho\} := \gamma^{(t+1)} \leq \\ \gamma^{(t)} := \min\{s : \mathbb{P}_{\boldsymbol{p}^{(t-1)}}(L(\boldsymbol{X}) \leq s) \geq \rho\}. \end{aligned} \tag{3.39}$$

This completes the proof. □

Using the result of Theorem 3.5 and the consistency of the sample quantiles (Zieliński, 1998), it is easy to conclude the monotonicity of sample quantiles sequence $\{\hat{\gamma}^{(t)}\}_{t\in\mathbb{N}}$.

Corollary 3.2. *From Theorem 3.5 and Corollary 3.1 ($\mathscr{L}^{lex}(\varphi(\boldsymbol{x})) \leq \mathscr{L}^{lex}(\boldsymbol{x})$), the sequence $\tilde{\gamma}^{(t)}$ is also non-increasing, i.e.*

$$\gamma^{(1)} \geq \tilde{\gamma}^{(1)} \geq \tilde{\gamma}^{(2)} \cdots \geq \tilde{\gamma}^{(t)} \ldots. \tag{3.40}$$

After proving the monotonicity of quantile sequences, we need another important ingredient called the *Persistence of Non-Zero Probability Property*, to prove the convergence of the deterministic CESBKM algorithm to the global minimum of $\mathscr{L}^{lex}$ defined in (3.6). This property is stated in the following Corollary 3.3 and follows directly from Theorem 3.5 by considering $\gamma^{(0)} = \gamma^{\star}$.

Corollary 3.3. *Suppose parameter array $\boldsymbol{p}^{(0)}$ in the CESBKM procedure (Algorithm 3.3) is initialized in such a way that $\mathbb{P}_{\boldsymbol{p}^{(0)}}(\mathscr{L}^{lex}(\boldsymbol{X}) = \gamma^{\star}) \neq 0$. Then, it implies that for each $t \in \mathbb{N}$, the following holds:*

$$\mathbb{P}_{\boldsymbol{p}^{(t)}}(\mathscr{L}(\boldsymbol{X}) = \gamma^{\star}) \neq 0 \quad \textit{where} \quad \gamma^{\star} \in \min_{\boldsymbol{x}} \mathscr{L}(\boldsymbol{x}). \tag{3.41}$$

Theorem 3.6. *Using a decreasing sequence $\rho^{(t)}$, the sequences $\{\gamma^{(t)}\}_{t\in\mathbb{N}}$ and $\{\tilde{\gamma}^{(t)}\}_{t\in\mathbb{N}}$ generated by the CESBKM procedure outlined in Algorithm 3.3 will reach the global minimum $\gamma^{\star}$ of the function $\mathscr{L}$ in a finite number of iterations. This means that, there exists*

$T \in \mathbb{N}$ such that for any $t \geq T$, the following relationship holds:

$$\gamma^{\star} = \gamma^{(t)} = \tilde{\gamma}^{(t)} = \gamma^{(t+h)} = \tilde{\gamma}^{(t+h)} \quad \textit{for any} \quad h \in \mathbb{N}.$$

Proof. By Theorem 3.5, the equality $\gamma^{(t)} = \gamma^{(t+h)}$ leads to $\boldsymbol{p}^{(t-1)} = \boldsymbol{p}^{(t+h-1)}$. Then, the latter implies that

$$\begin{aligned} \gamma^{(t+h)} &= \min\{s : \mathbb{P}_{\boldsymbol{p}^{(t+h-1)}}(\mathscr{L}^{lex}(\boldsymbol{X}) \leq s) \geq \rho^{(t+h)}\} \\ &= \min\{s : \mathbb{P}_{\boldsymbol{p}^{(t-1)}}(\mathscr{L}^{lex}(\boldsymbol{X}) \leq s) \geq \rho^{(t+h)}\} \\ &= \min\{s : \mathbb{P}_{\boldsymbol{p}^{(t-1)}}(\mathscr{L}^{lex}(\boldsymbol{X}) \leq s) \geq \rho^{(t)}\} = \gamma^{(t)} \quad \text{for any} \quad h \in \mathbb{N}. \end{aligned} \tag{3.42}$$

Next, we show that the set $\{\mathscr{L}^{lex}(\boldsymbol{x}) \mid \mathscr{L}^{lex}(\boldsymbol{x}) \leq \gamma^{(t)}\}$ should be a singleton. To prove this, we proceed by contradiction. Suppose that the set is not a singleton which means that there are at least two diffent values of $\mathscr{L}^{lex}(\boldsymbol{x})$ equal or less than $\gamma^{(t)}$ for any $t \geq T$.Due to the geometrically decaying structure of $\rho^{(t)} = \rho \cdot e^{-qt}$ as defined in the CESBKM algorithm, from (3.42) it follows that there exists a finite $H \in \mathbb{N}$ such that for all $h \geq H$,

$$\begin{aligned} \gamma^{(t+h)} &= \min\{s : \mathbb{P}_{\boldsymbol{p}^{(t+h-1)}}(\mathscr{L}^{lex}(\boldsymbol{X}) \leq s) \geq \rho^{(t+h)}\} \\ &< \min\{s : \mathbb{P}_{\boldsymbol{p}^{(t-1)}}(\mathscr{L}^{lex}(\boldsymbol{X}) \leq s) \geq \rho^{(t)}\} = \gamma^{(t)}, \end{aligned} \tag{3.43}$$

which contradicts (3.42).

Since the set $\{\mathscr{L}^{lex}(\boldsymbol{x}) \mid \mathscr{L}^{lex}(\boldsymbol{x}) \leq \gamma^{(t)}\}$ is a singleton, then we have

$$\{\mathscr{L}^{lex}(\boldsymbol{x}) \mid \mathscr{L}^{lex}(\boldsymbol{x}) \leq \gamma^{(t)}\} = \{\mathscr{L}^{lex}(\boldsymbol{x}^{\star})\} = \{\gamma^{\star}\} = \gamma^{(t)} = \gamma^{(t+h)} \tag{3.44}$$

The proof for the sequences $\tilde{\gamma}^{(t)}$ follows by applying a similar reasoning. □

After proving the convergence of the CESBKM in finite number of iterations, it is meaningful to apply the algorithm to real dataset and compare its convergence properties to those of the generic CE. In the next section, we discuss the class of time series models called, Open Loop Threshold Autoregressive (OLTAR) Model (Chen, 1995), then, we show that identification of OLTAR parameters is closely related to the clustering problem we have discussed in Section 3.2, and finally, we apply CE, CESBKM and Bayesian algorithms to identify OLTAR parameters and compare there convergence speed.

3.6 Application of the CESBKM Algorithm To Synthetic Datasets

3.6.1 Simulation Protocol

The purpose of the following simulations is to find out whether the properties of the deterministic CESBKM algorithm and its variants (CE, CESB, CEKM and CESBKM) which we have proven earlier, carry forward to finite sample setting under different environments. More precisely, we want to analyze whether symmetry breaking, exponential decay and KM features enhance the generic CE in terms of convergence speed, number of iterations and the loss function value. To that end, we conducted comprehensive simulation studies using two types of datasets, three different symmetry breaking implementations with three different exponential decay parameters $q = \{0, 0.05, 0.1\}$. One should note that when $q = 0$ there is no decay in quantile rank parameter as in the case of the generic CE algorithm.

Data Generation

For each simulation configuration, we generated two types of datasets:

Threshold Model: We generated 100 synthetic datasets, each containing $T_1 = T_2 = 50$ observations, following:

$$z_i = \epsilon_i + \begin{cases} -2 + 0.5y_i & \text{if} \quad 1 \leq i \leq 50 \\ 0.5 - 1.8y_i & \text{if} \quad 51 \leq i \leq 100, \end{cases}$$

where $\epsilon_i \sim N(0, \sigma^2)$ with $\sigma^2 \in \{1, 4, 9\}$.

Pure Noise Model: We generated 100 datasets each with length 100 such that both response and predictor follow:

$$(z_i, y_i) \sim N(0, \sigma^2), \quad i = 1, \ldots, 100$$

with $\sigma^2 \in \{1, 4, 9\}$.

Symmetry Breaking Constraints (SBC)

We investigated three distinct symmetry breaking approaches:

— Lexicographic ordering based on fitted values
— Lexicographic ordering based on cluster centroids
— Fixed probability initialization ($p_{11}^{(0)} = 1$)

Parameter Grid

Each algorithm variant was evaluated across:
— Quantile parameter: $\rho \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$
— Decay parameter: $q \in \{0, 0.05, 0.1\}$
— Number of Monte Carlo samples at each iteration: $M \in \{500, 750, 1000\}$

Implementation Details

The simulations were executed with:
— Maximum iterations: 100
— Platform: R version 4.3.1 on Linux
— Parallel processing: 48-64 cores.

The configuration of simulations is outlined in Table 3.8.

Table 3.8 – Simulation Study Configurations

Component	**Specifications**
Data Types	• Threshold Model • Pure Noise Model
Algorithm Variants	• CE: Standard Cross-Entropy • CESB: CE with Symmetry Breaking • CEKM: CE with K-means • CESBKM: CE with SB and K-means
Symmetry Breaking	• Fitted Values Ordering • Centroid-based Ordering • Fixed Probability ($p_{11}^{(0)} = 1$)
Parameters	• $\rho \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$ • $q \in \{0, 0.05, 0.1\}$ • $M \in \{500, 750, 1000\}$ • $\sigma^2 \in \{1, 4, 9\}$
Implementation	• Maximum number of iterations: 100 • Parallel Cores: 48-64

3.6.2 Analysis of Simulation Results

To systematically analyze the extensive simulation results, we organize our findings into several key aspects, with detailed results presented through a series of boxplots in Appendix 6.2.2. More precisely, we analyze each (data type) × (SBC) configuration separately. For each symmetry breaking implementation (fitted values, centroids, and fixed probability), we generated 27 boxplots per metric, corresponding to different parameter combinations. These visualizations capture three performance metrics, namely, loss function values, the computation time, and the number of iterations until convergence. Moreover, the multi-way Analysis of Variance (ANOVA) is employed to systematically evaluate the effects of different factors and their interactions on algorithm performance. ANOVA decomposes the total variability in the response variable into components attributable to different factors and their interactions, enabling us to assess their statistical significance (Dean et al., 2017).

Threshold Model With SBC On Fitted Values

The performance of the CESBKM variants under different environments for Threshold Model with SBC on fitted values are shown in Appendix 6.2.2. Moreover, Table 3.9 below outlines the multiway ANOVA results which describe the statistical significance of different factors (parameters) on algorithm performance as measured by the loss function value. The ANOVA results reveal several significant findings.

Main Effects:

— The noise level (σ^2) exhibits the strongest effect ($F = 162700, p < 2e-16$), indicating that performance is primarily determined by the noise in the data. This is not suprising as higher noise level of data is usually expected to lead higher value of the loss function found by an optimization algorithm.

— Algorithm choice (method) shows substantial impact ($F = 787.2, p < 2e{-}16$). This is an important finding confirming the difference between different specifications of the CESBKM algorithm we discussed at the deterministic level.

— The quantile parameter ρ demonstrates strong influence (F = 692.5, p < 2e-16) which is in line with our findings in the deterministic case.

— The decay parameter q significantly affects performance (F = 149.3, p < 2e-16) which was examined comprehensively at deterministic level.

— The size of Monte Carlo samples (M) per iteration has a modest but significant

Table 3.9 – Multi-way ANOVA Results For Threshold Model With SBC On Fitted Values

Source	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
method	4	9458	2364	7.872e+02	<2e-16	***
ρ	1	2080	2080	6.925e+02	<2e-16	***
q	1	448	448	1.493e+02	<2e-16	***
M	1	20	20	6.726e+00	0.0095	**
σ^2	1	488681	488681	1.627e+05	<2e-16	***
method $\times$ ρ	4	2596	649	2.161e+02	<2e-16	***
method $\times$ q	4	442	111	3.681e+01	<2e-16	***
$\rho \times q$	1	457	457	1.522e+02	<2e-16	***
method $\times$ M	4	27	7	2.227e+00	0.0634	.
$\rho \times M$	1	0	0	7.100e-02	0.7905	
$q \times M$	1	8	8	2.545e+00	0.1107	
method $\times$ σ^2	4	7	2	6.150e-01	0.6517	
$\rho \times \sigma^2$	1	7	7	2.271e+00	0.1318	
$q \times \sigma^2$	1	1	1	4.790e-01	0.4889	
$M \times \sigma^2$	1	1	1	2.100e-01	0.6466	
method $\times$ $\rho \times q$	4	432	108	3.598e+01	<2e-16	***
method $\times$ $\rho \times M$	4	9	2	7.100e-01	0.5851	
method $\times$ $q \times M$	4	14	3	1.146e+00	0.3328	
$\rho \times q \times M$	1	4	4	1.458e+00	0.2272	
method $\times$ $\rho \times \sigma^2$	4	25	6	2.081e+00	0.0805	.
method $\times$ $q \times \sigma^2$	4	14	3	1.138e+00	0.3363	
$\rho \times q \times \sigma^2$	1	2	2	7.530e-01	0.3856	
method $\times$ $M \times \sigma^2$	4	1	0	7.200e-02	0.9907	
$\rho \times M \times \sigma^2$	1	6	6	2.035e+00	0.1537	
$q \times M \times \sigma^2$	1	1	1	4.480e-01	0.5033	
method $\times$ $\rho \times q \times M$	4	3	1	2.140e-01	0.9307	
method $\times$ $\rho \times q \times \sigma^2$	4	9	2	7.550e-01	0.5546	
method$\times\rho\times$M$\times\sigma^2$	4	13	3	1.076e+00	0.3664	
method$\times q\times$M$\times\sigma^2$	4	6	2	5.190e-01	0.7221	
$\rho \times q \times M \times \sigma^2$	1	1	1	3.120e-01	0.5764	
method $\times$ $\rho \times q \times M \times \sigma^2$	4	2	0	1.370e-01	0.9687	
Residuals	67420	202500	3			

Significance codes× 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

effect (F = 6.726, p = 0.0095).

Two-way Interactions:

— Strong interaction between method and ρ (F = 216.1, p < 2e-16)

— Significant interaction between method and q (F = 36.81, p < 2e-16)

— Notable interaction between ρ and q (F = 152.2, p < 2e-16). This is an important result for the purposes of our study which confirms that different levels of q and ρ parameters lead to different performances of the CESBKM in empirical level. This further supports the idea of adding an exponentially decaying quantile rank parameter which we discussed thouroghly in deterministic level.

Higher-order Interactions:

— The three-way interaction method $\times$ ρ $\times$ q is significant (F = 35.98, p < 2e-16)

— Most higher-order interactions are negligible.

The purpose of ANOVA table above is to understand whether the differences we see in boxplots of Appendix 6.2.2 are meaningful statistically. The eyeball inspection of the boxplots indeed reveal that high values of ρ lead to inferior solutions for the generic CE method, when the decay parameter $q = 0$. This is in line with our theoretical findings in the previous section. Indeed, the very purpose of introducing an exponential decay was to reduce the dependence of the CE algorithm on the quantile rank parameter. Moreover, we see from the corresponding boxplots in Appendix 6.2.2 that the convergence speed and time is significantly improved by SB and KM implementations.

Threshold Model With SBC On Centroids

The performance of CESBKM variants (when applied to the threshold model with SBC on centroids) in terms of the three metrics are visualized in Appendix 6.2.2. Interestingly, CESB does not exhibit the expected results in terms of loss metric. In terms of two other metrics, however, it behaves in line with the deterministic analysis in the previous section. We believe that imposing centroid based SBC is too strong and leads to premature convergence. This is reflected in fast convergence of CESB compared to CE method but its median loss value being less than that of the CE in almost all different environments. Ironically, as boxplots in Appendix 6.2.2 reveal SB seems to be a valuable addition for computation time and the number of iterations particularly for $\rho < 0.5$. Moreover, when SB is interacted with KM feature the algorithm works better than the CE method almost in all dimensions. Additionally, a multi-way ANOVA analysis in Table 3.10 reveal that the exponential decay (q), quantile rank (ρ) and their interaction are statistically significant

factors determining the performance of the CESBKM variants as measured by the loss value. The ANOVA results reveal several key findings:

Main Effects:

— Noise variance (σ^2) shows the strongest effect (F = 120500)

— Method choice has substantial impact (F = 4976)

— Quantile parameter ρ demonstrates strong influence (F = 3897)

— Size of Monte Carlo samples M shows modest but significant effect (F = 16.71)

Two-way Interactions:

— Strong interaction between method and ρ (F = 1827)

— Significant method $\times q$ interaction (F = 178.1)

— Notable method $\times\ \rho \times q$ interaction (F = 715.3)

Higher-order Interactions:

— Three-way interaction method $\times\ \rho \times q$ (F = 159.7)

— Complex interaction method $\times\ \rho \times q \times \sigma^2$ (F = 17.87)

Threshold Model With SBC On The First Element of The Cluster Vector

Interestingly, the results of imposing SBC on the first element of the cluster vector is similar to those of imposing SBC on centroids in the framework of the threshold model. In this case too, we do not see the expected effect of adding SB to the generic CE method in terms of the loss function. For brevity, we do not provide ANOVA results for this case. However, graphical representation of results are given in Appendix 6.2.2.

Noise Models

The behavior of the CESBKM variants with all three SBC's we used in this study was very similar to each other in the Noise Model framework. To our surprise, all variants of the CESBKM performed much better than expected in terms of the loss metric in Noise Model case. Indeed, we expected to see high spikes in CE variant when $\rho = 0.9$ and $q = 0$. However, It was not the case most of the time. Moreover, we expected the variants having KM to perform inferior to the pure CE version in terms of the loss metric because there is not any true centroid or class in pure noise models where usually KM algorithm is applied. Contrary to our expectations, variants involving KM performed better than the CE overall in the loss metric dimension in all noise models.

In number of iterations dimension, CE and CESB were inferior to other versions of the CESBKM algorithm. However, in terms of the computation time, CEKM and CESBKM

Table 3.10 – Multi-way ANOVA Results For Threshold Model With SBC On Centroids

Source	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
method	4	83805	20951	4.976e+03	< 2e-16	***
ρ	1	16407	16407	3.897e+03	< 2e-16	***
q	1	3101	3101	7.364e+02	< 2e-16	***
M	1	70	70	1.671e+01	4.35e-05	***
σ^2	1	507551	507551	1.205e+05	< 2e-16	***
method $\times$ ρ	4	30778	7695	1.827e+03	< 2e-16	***
method $\times$ q	4	2999	750	1.781e+02	< 2e-16	***
$\rho \times q$	1	3012	3012	7.153e+02	< 2e-16	***
method $\times$ M	4	59	15	3.524e+00	0.00700	**
$\rho \times M$	1	0	0	1.000e-03	0.97056	
$q \times M$	1	4	4	8.750e-01	0.34951	
method $\times$ σ^2	4	834	208	4.950e+01	< 2e-16	***
$\rho \times \sigma^2$	1	10	10	2.491e+00	0.11449	
$q \times \sigma^2$	1	7	7	1.643e+00	0.19988	
$M \times \sigma^2$	1	0	0	1.100e-02	0.91641	
method $\times$ $\rho \times q$	4	2689	672	1.597e+02	< 2e-16	***
method $\times$ $\rho \times M$	4	45	11	2.697e+00	0.02906	*
method $\times$ $q \times M$	4	17	4	9.990e-01	0.40675	
$\rho \times q \times M$	1	1	1	1.710e-01	0.67927	
method $\times$ $\rho \times \sigma^2$	4	155	39	9.178e+00	2.08e-07	***
method $\times$ $q \times \sigma^2$	4	246	61	1.459e+01	6.52e-12	***
$\rho \times q \times \sigma^2$	1	8	8	1.830e+00	0.17619	
method $\times$ $M \times \sigma^2$	4	1	0	6.300e-02	0.99260	
$\rho \times M \times \sigma^2$	1	1	1	2.300e-01	0.63118	
$q \times M \times \sigma^2$	1	0	0	8.000e-02	0.77742	
method $\times$ $\rho \times q \times M$	4	63	16	3.743e+00	0.00476	**
method $\times$ $\rho \times q \times \sigma^2$	4	301	75	1.787e+01	1.13e-14	***
method$\times\rho\times$M$\times\sigma^2$	4	7	2	4.220e-01	0.79274	
method$\times q\times$M$\times\sigma^2$	4	0	0	1.200e-02	0.99970	
$\rho \times q \times M \times \sigma^2$	1	0	0	2.000e-03	0.96450	
method $\times$ $\rho \times q \times M \times \sigma^2$	4	2	0	1.170e-01	0.97643	
Residuals	67420	283870	4			

Significance codes× 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

were not always outperforming the CE and CESB variants in the Noise Model framework. Interestingly, CESBKM variant performed superior to all other variants in most of the cases considered. The boxplots showing the simulation results on noise models can be found in Appendix 6.2.2, Appendix 6.2.2 and Appendix 6.2.2.

We present a multi-way ANOVA results for noise model with SBC on fitted values in Table 3.11. For brevity, we omit the ANOVA tables for the remaining model configurations as they demonstrate similar patterns of statistical significance across main effects and interactions. The complete set of ANOVA results is available in the supplementary materials [1]. The key finding across all configurations is the consistent significance of noise level (σ^2), method choice, quantile parameter (ρ) and exponential decay rate (q) as primary determinants of algorithm performance. This empirical evidence further justifies the proposed improvements to the generic CE model we made in the previous section.

3.7 Application of CESBKM To Bitcoin Market Volatility

3.7.1 Model Specification

Heterogenous Autoregressive (HAR) model due to Corsi (2009) has been very popular due to its ability to capture long-memory volatility dynamics inherent in stock markets. The key advantage of the model is that it incorporates volatilities measured over different horizons. The model is specified as follows:

$$RV_{i+1}^{(d)} = \alpha_0 + \alpha_1 RV_i^{(d)} + \alpha_2 RV_i^{(w)} + \alpha_3 RV_i^{(m)} + \eta_{i+1}, \tag{3.45}$$

where $\{RV_i^{(d)}\}_{i\in\mathbb{N}}$ is the time series of daily realized volatility and i is the time (day) index, $RV_i^{(w)}$ and $RV_i^{(m)}$ are weekly and monthly realized volatilities, respectively, measured at daily frequency.

We propose to use Clusterwise HAR (CHAR) model with two clusters corresponding to high and low volatility periods to capture the volatility clustering phenomenon inherent in financial markets and documented in finance literature (Cont, 2007). The model is

1. The supplementary material is available at https://github.com/salahaddiniayyubi/CESBKM_FINAL.git

Table 3.11 – Multi-way ANOVA Results For Noise Model With SBC On Fitted Values

Source	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
method	4	117	29	1.244e+02	< 2e-16	***
ρ	1	4	4	1.691e+01	3.93e-05	***
q	1	1	1	5.217e+00	0.02236	*
M	1	0	0	5.000e-01	0.47947	
σ^2	1	196046	196046	8.370e+05	< 2e-16	***
method $\times$ ρ	4	3	1	3.086e+00	0.01498	*
method $\times$ q	4	1	0	1.132e+00	0.33931	
$\rho \times q$	1	1	1	5.223e+00	0.02230	*
method $\times$ M	4	0	0	1.500e-01	0.96317	
$\rho \times M$	1	0	0	2.900e-02	0.86450	
$q \times M$	1	0	0	1.000e-03	0.96934	
method $\times$ σ^2	4	58	14	6.171e+01	< 2e-16	***
$\rho \times \sigma^2$	1	2	2	8.583e+00	0.00339	**
$q \times \sigma^2$	1	1	1	2.674e+00	0.10200	
$M \times \sigma^2$	1	0	0	1.870e-01	0.66582	
method $\times$ $\rho \times q$	4	1	0	1.395e+00	0.23296	
method $\times$ $\rho \times M$	4	0	0	1.000e-01	0.98264	
method $\times$ $q \times M$	4	0	0	0.000e+00	1.00000	
$\rho \times q \times M$	1	0	0	4.800e-02	0.82641	
method $\times$ $\rho \times \sigma^2$	4	1	0	1.546e+00	0.18574	
method $\times$ $q \times \sigma^2$	4	1	0	5.930e-01	0.66755	
$\rho \times q \times \sigma^2$	1	1	1	2.872e+00	0.09016	.
method $\times$ $M \times \sigma^2$	4	0	0	6.600e-02	0.99204	
$\rho \times M \times \sigma^2$	1	0	0	4.800e-02	0.82616	
$q \times M \times \sigma^2$	1	0	0	2.000e-02	0.88745	
method $\times$ $\rho \times q \times M$	4	0	0	2.100e-02	0.99915	
method $\times$ $\rho \times q \times \sigma^2$	4	1	0	8.440e-01	0.49715	
method$\times\rho\times$M$\times\sigma^2$	4	0	0	1.260e-01	0.97310	
method$\times q\times$M$\times\sigma^2$	4	0	0	1.300e-02	0.99966	
$\rho \times q \times M \times \sigma^2$	1	0	0	5.400e-02	0.81607	
method $\times$ $\rho \times q \times M \times \sigma^2$	4	0	0	2.700e-02	0.99864	
Residuals	67420	283870	4			

Significance codes× 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

specified as follows:

$$\begin{aligned} RV_{i+1}^{(d)} =& \mathbb{1}_{\{\boldsymbol{x}_i^{true}=1\}}(\alpha_{10} + \alpha_{11} RV_i^{(d)} + \alpha_{12} RV_i^{(w)} + \alpha_{13} RV_i^{(m)}) \\ &+ \mathbb{1}_{\{\boldsymbol{x}_i^{true}=2\}}(\alpha_{20} + \alpha_{21} RV_i^{(d)} + \alpha_{22} RV_i^{(w)} + \alpha_{23} RV_i^{(m)}) + \eta_{i+1} \end{aligned} \tag{3.46}$$

where η_i is the error term with zero mean, $\boldsymbol{x}_k^{true}$ denotes the true latent volatility cluster and a_{kj} for $k = 1, 2$ and $j = 0, 1, 2, 3$ are coefficients to be estimated.

Here, we apply this model to explain the Bitcoin volatility.

BTC price data and model estimation

The BTC price data were collected in 5-minute frequency from Binance exchange. The sample period runs from January 31, 2023 to April 14, 2024. Moreover, the BTC prices at 5-minute frequency were used to obtain the daily, weekly and monthly volatility series as described in (Bergsli et al., 2022) and in Chapter 2 (see 33).

CE and CESBKM algorithms are used to minimize the loss function of the form (3.1). More precisely, we first estimate the cluster vector by using the CE and CESBKM algorithms. We denote by $\hat{\boldsymbol{x}}_{CE}^{\star}$ and $\hat{\boldsymbol{x}}_{CESBKM}^{\star}$ of the optimal cluster vector $\boldsymbol{x}^{\star}$ obtained with CE and CESBKM respectively. Then, we use the Ordinary Least Squares (OLS) method to estimate the model parameters $\{a_{ij}\}$ for (3.45) and (3.46), respectively.

3.7.2 Empirical Results

Table 3.12 summarizes the estimation results. As shown by the table, estimation of the optimal cluster vector $\boldsymbol{x}^{\star}$ using the CESBKM algorithm with $\rho = 0.9$, $q = 0.1$ and $M = 4000$ (CHAR-CESBKM) returns better results than estimation with the generic CE (CHAR-CE) with $\rho = 0.025$ and $M = 4000$. Moreover, as expected, both CHAR-CE and CHAR-CESBKM outperform the baseline HAR model in terms of the model fit. CHAR-CESBKM has all parameters statistically significant at 0.05 significance level except for α_{13}. Incidentally, the α_{20} and α_{23} are statistically insignificant in CHAR-CE. In the generic HAR model, all coefficients except the coefficient of $RV^{(w)}$ variable and intercept are statistically insignificant even at 0.1.

To sum up, the results assert the volatility clustering phenomenon for Bitcoin market since estimation of two-regime models both provide a better fit to the data than a simple HAR and return mostly statistically significant results for parameters. Ultimately, the

estimation of the two-state CHAR model using the CESBKM algorithm results in better model fit to data.

Table 3.12 – Parameter Estimates and Model Fit Statistics

Model	Parameter	Coefficient	Std. Error	Significance
HAR	α_0	3.126	0.938	***
	α_1	0.054	0.054	
	α_2	0.300	0.118	**
	α_3	0.153	0.167	
	R^2	0.051		
CHAR-CE	α_{10}	27.195	1.963	***
	α_{11}	0.156	0.078	**
	α_{12}	-1.107	0.138	***
	α_{13}	-0.594	0.258	**
	α_{20}	-0.459	0.830	
	α_{21}	-0.104	0.053	**
	α_{22}	1.281	0.148	***
	α_{23}	-0.131	0.158	
	R^2	0.629		
CHAR-CESBKM	α_{10}	1.174	0.591	**
	α_{11}	0.130	0.048	***
	α_{12}	0.349	0.076	***
	α_{13}	0.171	0.105	
	α_{20}	61.639	2.516	***
	α_{21}	-0.135	0.053	**
	α_{22}	-2.069	0.245	***
	α_{23}	-1.639	0.374	***
	R^2	0.782		

Note: *** p < 0.01, ** p < 0.05

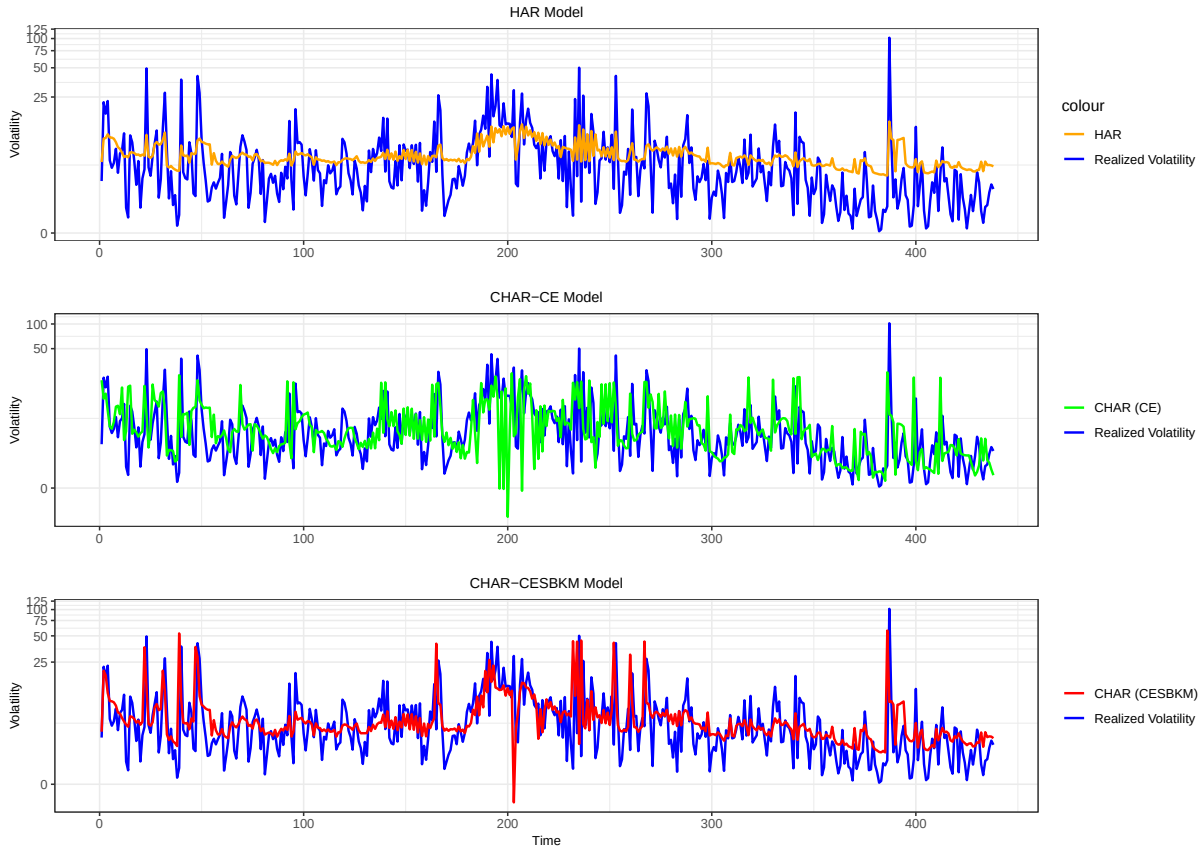


Figure 3.4 – Daily Realized Volatility and Its Forecasts (in Logarithmic Scale): HAR Model (top panel), Clusterwise HAR Model with CE-estimated Cluster Vector (middle panel), and Clusterwise HAR Model with CESBKM-estimated Cluster Vector (bottom panel). The blue solid lines represent realized volatility, while the colored lines show the corresponding model forecasts.

3.8 Conclusion

This paper proposed a novel CESBKM algorithm that hybridizes the CE and K-means algorithms to solve clustering and clusterwise linear regression problems more effectively. The CESBKM algorithm addresses key theoretical limitations of the generic CE algorithm related to symmetry issues and quantile rank parameter initialization.

Specifically, we show with few examples that the generic CE may not converge to the global optimum in the presence of symmetries. We tackled this by introducing symmetry-breaking constraints. However, even with these constraints, convergence was shown to depend on the choice of the quantile rank parameter ρ. We used a simple example to illustrate how initializing ρ too high can distort theoretical convergence. Therefore, we introduce geometrically decaying structure to $\rho^{(t)} = \rho \cdot e^{-qt}$ to guarantee convergence. To improve convergence speed, we integrated the K-means algorithm into the CE framework, establishing the CESBKM algorithm.

Additionally, we proved the convergence of the CESBKM algorithm. In doing so, we also proof the monotonicity of quantile sequences generated for both generic CE and CESBKM algorithm under much milder conditions than the existing literature.

Moreover, the simulation studies highlight the good performance of the stochastic CESBKM algorithm compared to the generic CE. Furthermore, applying both algorithms to estimate the clusterwise HAR model for Bitcoin volatility further reaffirms the CESBKM's advantages.

Chapter 4

Threshold Autoregressive Model with Decision Tree

4.1 Introduction

There is a huge body of literature on non-linear time series modeling which has been prospering since 1980's (Tong, 2002). Particularly, after the publication of the seminal paper of (Hamilton, 1989) the interest to nonlinear modeling has increased dramatically (Potter, 1999). Unlike linear models, the non-linear time series models are able to capture complex dynamics inherent in time series processes (Chen et al., 2011; Tong, 2011). Particularly, time reversibility feature of the linear models such as conventional Autoregressive Moving Average (ARMA) make the latter less attractive in financial and economic applications since economic and financial time series are usually highly non-linear and time irreversible (Tong, 2011). Regime Switching Autoregressive (RSAR) models as defined in time series literature currently comprise a wide class of non-linear time series models which can over-perform linear alternatives both in explanation and prediction of the underlying processes. Among non-linear models, Threshold Autoregressive (TAR) models have gained significant traction since 1980s. These models, as originally specified by Tong (1978; 1983), offer a powerful framework for modeling complex dynamics such as limit cycles, amplitude-dependent frequencies, and jump phenomena - characteristics often observed in economic and financial data (Tong, 2011).

The general form of a TAR model can be expressed as:

$$y_t = \boldsymbol{y}'_{t-1}\boldsymbol{\phi}^{(x_t)} + \epsilon_t^{(x_t)}, \quad t = 1, 2 \dots \tag{4.1}$$

where $t \in \mathbb{N}^\star$ denotes the time point, y_t is the target time series process. $q \in \mathbb{N}^\star$ the order

of the autoregressive process and $\boldsymbol{y}_{t-1} = (1, y_{t-1}, \ldots, y_{t-q})'$ corresponds to the history until the q-th lag and with the intercept. $x_t \in \{1, \ldots, k\}$ is a time-varying regime variable that we call state variable in the following. $\boldsymbol{\phi}^{(x_t)}$ is a vector of parameters corresponding to the state x_t, and $\epsilon_t^{(x_t)} \sim \mathcal{D}\left(0, \theta^{(x_t)}\right)$ is a zero mean error with $\mathcal{D}$ denoting some stationary distribution with parameters $\theta^{(x_t)}$. Interestingly, the threshold mechanism, also known as the regime switching mechanism, which defines the law of motion of the state variable x_t, gives rise to different dynamics and thus results in different class of TAR models, some of which have been described in Tong (2011). In this study, we focus on a special class of TAR models termed Open-Loop Threshold Autoregressive (OLTAR) Models whose threshold mechanism is driven by a latent function. As it will be demonstrated in the next parts, estimating the parameters of OLTAR models is usually challenging. Chen (1995) has developed a strategy to perform estimation for the latter class of models and has shown that the estimators are consistent under suitable conditions.

Here, we introduce a novel approach to OLTAR model estimation which combines both the Cross-Entropy (CE) method (Rubinstein, 1999) and the Classification And Regression Tree algorithm (CART) introduced by Breiman (1996). We call this new method the CEDT algorithm (for Cross Entropy Desicion Tree). Our approach addresses two key challenges in OLTAR modeling:

— the estimation of state variables in Open-Loop Threshold Autoregressive (OLTAR) models, where the threshold function and its parameters are unknown,

— the selection of relevant threshold variables from a potentially large feature space.

The CEDT algorithm leverages the global optimization capabilities of the CE method for state estimation while the decision tree algorithm allows to capture complex variables relationship thanks to its complete non-parametric nature. Moreover, our method can imporve interpretability of the threshold mechanism, espacially when the decison trees are simple. Therefore, this combination of CE and CART allows for a more comprehensive exploration of the model space, potentially leading to more accurate and interpretable learning of OLTAR models.

Furthermore, as a second contribution, we propose a novel OLTAR-HAR model for volatility modeling which is the blend of the TAR model and the popular Heterogenous Autoregressive (HAR) models. HAR model is a very simple model which is also very powerful model popular in financial literature to model and forecast the realized volatility

(Corsi, 2009). However, simple HAR models may not take into account regime-switching properties of financial markets well. Since the publication of the seminal paper of Corsi (2009), many extensions of the generic HAR model have been proposed in literature. For example, tree-HAR model (Audrino and Corsi, 2010; Corsi et al., 2012) uses Classification and Regression Trees (CART) of Breiman et al. (1984) with the generic HAR to capture long-memory and possible non-linear effects, HAR model of realized volatility with jumps (HAR-RV-J) and with continuous jumps (HAR-RV-CJ) of Andersen et al. (2007) to capture the abrupt price variations in prices of financial assets which have long-term effects on their volatility by decomposing volatility into continuos sample path and jumps, HAR with threshold jumps (HAR-TCJ) of Corsi et al. (2010) proposes threshold jumps, and finally, HAR with signed jumps (HAR-SJ) of Sheppard and Patton (2011) seek to capture the leverage effect by introducing signs of the intra-daily returns into the generic HAR model (for a more detailed survey of HAR models see Corsi et al. (2012) and Wang et al. (2016)). The OLTAR-HAR model we propose here is similar to the tree-HAR model in the sense that both have a HAR model in each regime. The major difference between the tree-HAR model and the OLTAR-HAR model is that OLTAR-HAR model do not need to assume any a priori assumption about the threshold function driving the regime switching mechanism. In that sense, the OLTAR-HAR model is more general than the tree-HAR model.

This chapter is structured as follows. We begin by presenting the problem statement in Section 4.2, highlighting the challenges of modeling regime-switching dynamics in financial time series. In Section 4.3, we introduce the general idea of the proposed inference algorithm that combines Cross-Entropy and Decision Trees (CEDT). Section 4.4 describes the classification stage of the algorithm, focusing on estimating the state variables and OLTAR parameters. In Section 4.5, we detail the searching stage, where decision trees are employed to model the threshold dynamics. Finally, in Section 4.6, we apply the proposed CEDT model to both synthetic datasets and real-world Bitcoin volatility data, presenting the simulation results and model performance.

4.2 Problem Statement

From (4.1), let consider a general two-state switching autoregressive model:

$$y_t = \phi_0^{(x_t)} + \sum_{i=1}^{q} \phi_i^{(x_t)} y_{t-i} + \epsilon_t \tag{4.2}$$

where y_t is the target time series at time t and $x_t \in \{1, 2\}$ is the state variable.

As mentioned previously, several models have been defined from the previous general formulation (4.2) depending on the knowledge on x_t.

Case 1) Threshold Autoregressive (TAR) Model: this model assumes that the dynamics of x_t is governed by a known threshold function:

$$x_t = \begin{cases} 1 & \text{if} \quad g(\boldsymbol{z}_{t-1}, \boldsymbol{y}_{t-1}; \Theta) \leq s \\ 2 & \text{if} \quad g(\boldsymbol{z}_{t-1}, \boldsymbol{y}_{t-1}; \Theta) > s \end{cases} \tag{4.3}$$

Here, $g(\boldsymbol{z}_{t-1}, \boldsymbol{y}_{t-1}; \Theta)$ is the threshold function, $\boldsymbol{z}_{t-1}$ is the set of lags of z_t up to time $t-1$ (exogenous variables), $\boldsymbol{y}_{t-1}$ are lags of endogenous variables up to time $t-1$, Θ is the set of parameters of the threshold function, and s is the threshold value (split point). In a typical TAR setup, as introduced in the seminal work of Tong (1978), the threshold function, its arguments and its parameters are known. So the primary task consists in determining the appropriate threshold value s.

Case 2) Digression Model: in this model, the state variable x_t is assumed to be random with $P(x_t = 1) = \theta_t$ for $t = 1, \ldots, T$ where θ_t are some unknown parameters. Moreover, x_t is assumed to be independent of ϵ_t. This model is called a digression model (Chen, 1995) or a Switching Autoregressive model (Wu and Chen, 2007).

Case 3) Open-Loop Threshold Autoregressive (OLTAR) Model: in this model, the threshold function, its arguments, and the threshold value exist but are completely unknown. We classify such models as *OLTAR*, following the terminology of Chen (1995).

For TAR model (case 1), the threshold s is typically estimated by searching over a grid of possible values. For each candidate s, the data is split into regimes through the deterministic threshold function, and the model parameters are estimated separately in each regime using OLS. The optimal s is selected as the one that minimizes the sum of squared residuals (SSR), providing the best fit for the data.

The inference for Digression and OLTAR models is significantly more complex than

for TAR models, primarily due to the unknown or random nature of the regime-switching mechanism, which requires advanced estimation techniques. As a first approach, Chen (1995) proposes to treat the OLTAR model as a digression model and solve the minimization problem without any regard to the unknown threshold function. However, this leads to inconsistent estimates. Then, Chen introduces the following approach. First, the state of each observation needs to be correctly identified. To perform this, he suggests two different algorithms, namely the discarding algorithm and the Bayesian algorithm that enables to obtain consistent estimates for some set of the state variables under suitable conditions. Next, once some states are correctly identified (i.e. some observations are discarded), the model corresponds to several classical linear regression problems and so parameter estimation can be easily performed by using the Ordinary Least Squares (OLS) In this case, OLS estimators are consistent, but inefficient due to the fact that a portion of observations is discarded (Tong, 1978; Chen, 1995). Interestingly, Chen (1995, p. 468) discusses the differences between the discarding and Bayesian algorithms. The author points out that while the discarding algorithm is fast in some cases, it is very sensitive to initial values. Moreover, it may largely fail when the autoregressive order is high and the sample size is small. The Bayesian algorithm, on the other hand, is computationally expensive but proves more reliable under these challenging conditions. After consistent parameter estimation of the regression parameters and the states, it remains to learn (approximate) the latent function $g(\boldsymbol{z}_{t-1}, \boldsymbol{y}_{t-1}; \Theta)$ by selecting suitable threshold variables from the feature space Ω_t. To perform this, Chen (1995) used graphical methods for this, which can be subjective and very time-consuming.

In this work, we focus on OLTAR models, as their potential for greater interpretability compared to TAR models makes them a promising approach for addressing the complexities of regime dynamics. In particular, we propose a novel algorithm combining existing methods' strengths with a more systematic approach to variable selection and state estimation.

4.3 General description of the proposed inference algorithm based on the Cross-Entropy and Decision Trees

To solve the challenging estimation problem of OLTAR models, we propose an algorithm named CEDT, which combines the cross-entropy method and decision trees. The method consists of two main stages:

1. **Classification Stage:** We apply the Cross Entropy (CE) algorithm (Rubinstein, 1997) to estimate $\{x_t\}_{t=1}^{T}$ and identify with the least uncertainty using a CE-specific selection criterion.
2. **Searching Stage:** We employ a decision tree algorithm to select variables from Ω_t that are predictive of the most informative regime estimates. This stage employs a systematic tree-based approach to identify the most relevant threshold variables, improving upon the graphical methods used in previous studies.

These two steps, collectively referred to as the CEDT algorithm, offer several advantages over existing methods:

— The advantage of the CE over the discarding algorithm is similar to that of the Bayesian algorithm because the CE is a global search algorithm (de Boer et al., 2005). While comparing the computational efficacy of the Bayesian algorithm (Chen, 1995) and the CE method is beyond the scope of this study, one notable advantage of the CE is that it does not require dealing with integrals, which can often be challenging.

— The decision tree approach in Stage 2 offers an interpretable and efficient method for threshold variable selection, capable of handling high-dimensional feature spaces, unlike the graphical method.

— The combination of these stages allows for a more comprehensive exploration of the model space, potentially leading to more accurate and interpretable learning of OLTAR models.

In the subsequent sections, we will provide a detailed description of each stage of the CEDT algorithm, including the specific implementation of the cross-entropy method and the decision tree approach.

4.4 The CEDT Algorithm's Classification Stage

In classification stage we perform two important operations. First, we use the CE method to estimate the sequence of the state variables. In the second step, we estimate the parameters of the OLTAR model by ignoring part of the observations in the sample at hand and choosing the ones for which we are most certain about their states. In Section 4.4.1, the minimization problem to get the estimates of the state sequence is presented. Then, in Section 4.4.2, we discuss how parameters of the OLTAR model can be estimated consistently given the estimated sequence of the state variables.

4.4.1 Estimating The Sequence of The State Variables Using The CE Method

In the classification stage, we aim to estimate the sequence $\{x_t\}_{t=1}^{T}$ by minimizing some empirical loss function, assuming we have T measurements for each of the temporal variables $(y_t, y_{t-1}, \ldots, y_{t-q})$ at each time point $t = 1, \ldots, T$. The problem to be solved at this stage is the well-known clusterwise linear regression problem discussed in the previous (3).

It is worth emphasizing that, the naive grid search method requires 2^T evaluations of the loss function at hand given a 2-state TAR model which can be cumbersome. Particularly, one has to note that the difficulty of the problem increases both in the number of states and the length of the time series. Therefore, estimation of the unknown parameters will be performed using the well-known and fast converging CE algorithm (Rubinstein and Kroese, 2004).

The CE method is motivated by the problem of estimating rare event probabilities in complex stochastic systems. Later on, it was realized that the method can be applied to solve complex deterministic optimization problems (Rubinstein, 1999). The CE algorithm whether applied to rare event probability estimation or optimization problems has intuitive nature combining two important steps:

1. Generation of random sample of trajectories based on a given random data generation mechanism;
2. Updating parameters of the random mechanism to produce better samples and iterate until a given stopping criterion is satisfied.

Since its discovery the CE method has been applied to wide variety of problems such as

buffer allocation, static simulation models, queuing models of telecommunication systems, control and navigation, DNA sequence alignment, sensor selection in heterogeneous sensor networks and to many others mentioned in (de Boer et al., 2005) and references therein.

Before implementing the CE algorithm, let us clearly define the empirical loss function. Let $\boldsymbol{x} = \{x_1, \ldots, x_T\}$ be the state vector where $x_t \in \{1, 2\}$. Moreover, let $\mathbb{1}_{\{x_t=k\}}$ be an indicator function taking the value of one if $x_t = k$ and zero, otherwise. Finally, $\boldsymbol{y}_{t-1} = (1, y_{t-1}, \ldots, y_{t-q})'$ is the column vector of unity and q lags of the measurement $y_t \in \mathbb{R}$. Then, the empirical loss function for the digression model can be re-written as:

$$\mathscr{S}(\boldsymbol{\phi}, \boldsymbol{x}) = \sum_{k=1}^{2} \sum_{t=1}^{T} \mathbb{1}_{\{x_t=k\}} \|y_t - \boldsymbol{\phi}_k' \boldsymbol{y}_{t-1}\|^2, \tag{4.4}$$

where $\boldsymbol{\phi} = (\boldsymbol{\phi}_1, \boldsymbol{\phi}_2)$ and $\boldsymbol{\phi}_k = (\phi_{k0}, \ldots, \phi_{kq})'$ for $k = 1, 2$, and where q is the order of the autoregressive process. Note that, (4.4) is strictly convex with respect to $\boldsymbol{\phi}$ under regularity conditions and hence, we can define

$$S(\boldsymbol{x}) = \min_{\boldsymbol{\phi}} \mathscr{S}(\boldsymbol{\phi}, \boldsymbol{x}) = \sum_{k=1}^{2} \sum_{t=1}^{T} \mathbb{1}_{\{x_t=k\}} \|y_t - \boldsymbol{\phi}_k^{\star\prime} \boldsymbol{y}_{t-1}\|^2, \tag{4.5}$$

where $\boldsymbol{\phi}_k^\star$ corresponds to optimal vector of parameters at state k given by

$$\boldsymbol{\phi}_k^\star = \Big(\sum_{t=1}^{T} \mathbb{1}_{\{x_t=k\}} \boldsymbol{y}_{t-1} \boldsymbol{y}_{t-1}' \Big)^{-1} \sum_{t=1}^{T} \mathbb{1}_{\{x_t=k\}} \boldsymbol{y}_{t-1} y_t, \quad \text{for} \quad k = 1, 2. \tag{4.6}$$

As discussed in the previous chapter, the CE algorithm requires the selection of a sampling distribution. In this chapter, independent Bernoulli distributions with parameters $\boldsymbol{p}_0 = (p_{0,1}, \ldots, p_{0,T})$ will be used for the T state variables. After initializing the parameters of the sampling distribution, we draw N samples from Bernoulli($\boldsymbol{p}_0$) where each sample $\boldsymbol{X}_j$ for $j = 1, 2, \ldots, N$ has a size equal to T and where T is the length of the time series observations at hand. Then, we calculate the order statistics for the performance function $S_{(1)}, \ldots, S_{(N)}$ by evaluating S at each $\boldsymbol{x}_j = (x_{j1}, \ldots, x_{jT})$, where each $\boldsymbol{x}_j$ is one realization of the random vector $\boldsymbol{X}_j$ for $j = 1, \ldots, N$. Moreover, we choose a fixed rarity (percentile rank) parameter ρ to select the best performing state vectors at each iteration to generate the optimal parameter vector for the new proposal distribution in the next iteration. More precisely, at each iteration i, we define $\gamma_i := S_{(\lceil \rho N \rceil)}$, and the realizations of $\boldsymbol{X}_j$ which lead to the value of the performance function S less than or equal to γ_i are called *elite samples*. Then, as discussed in the previous chapter, the parameters of the sampling

distribution are updated by minimizing the Kullback-Leibler (KL) divergence with the distribution that concentrates on these elite samples. Since the sampling distribution is an independent Bernoulli distribution and belongs to the exponential family, the parameter update at the i-th iteration is given by:

$$p_{i,t} = \frac{\sum_{j=1}^{N} \mathbb{1}_{\{S(\boldsymbol{X}_j) \leq \gamma_i\}} X_{jt}}{\sum_{j=1}^{N} \mathbb{1}_{\{S(\boldsymbol{X}_j) \leq \gamma_i\}}}. \tag{4.7}$$

In this article, we refer to the updating rule in (4.7) as *hard updating rule* and to the CE algorithm using such a rule as the *Hard CE*. The *Smooth CE* in its turn corresponds to the following *smooth updating rule*:

$$p_{i,t}^{s} = \omega p_{i,t} + (1 - \omega) p_{i-1,t}^{s}, \tag{4.8}$$

where $\omega \in (0, 1)$ is a smoothing parameter. This smooth update ensures that the distribution changes more gradually from iteration to iteration, which helps prevent overshooting and ensures more stable convergence, especially in noisy or complex optimization landscapes.

Different stopping criteria can be used. Convergence of $\boldsymbol{p}_i$ to either 0 or 1 is the stopping criterion we use for the hard version in this study. Meanwhile, the stopping criterion we use for the smooth CE is when the percentile $\hat{\gamma}_i$ in Algo 4.1 does not change for five consecutive iterations i.e. $\hat{\gamma}_i = \hat{\gamma}_{i+1} = \ldots, = \hat{\gamma}_{i+5}$ (Rubinstein and Kroese, 2004; de Boer et al., 2005). The proposed CE method estimation of the states of a two state OLTAR model is summarized in Algo 4.1. Note that, we set the initial parameters of the CE based on (Rubinstein and Kroese, 2004) and the references therein.

4.4.2 Estimating The OLTAR Model Parameters Consistently

As a global search algorithm, the CE finds the global minimum of the loss function with a high probability given suitable conditions as discussed in the previous chapter and (Rubinstein and Kroese, 2004; de Boer et al., 2005). However, the optimal parameter vector $\boldsymbol{x}^{\star}$ provides inconsistent estimates of the states of the OLTAR model in (4.2) (Chen, 1995). Therefore, the estimates of the autoregressive parameters in (4.6) are also inconsistent. To tackle the problem, Chen (1995) proposes an algorithm which works well under suitable conditions to which we refer as *selection algorithm*. The algoritm works as

Algorithm 4.1 Hard and Smooth CE for Two-State OLTAR Models

Input: $\{y_t\}_{t=1}^{T}$ time series, where $t = 0, 1, ...T$ (time index).

1: Initialize $\boldsymbol{p}_0 = (1/2, 1/2, ..., 1/2)$. Let i := 1 (iteration counter), $N := 5T$ $\rho := \mathbb{1}_{\{T \geq 100\}} 0.01 + \mathbb{1}_{\{T<100\}} \frac{\ln(T)}{T}$, if smooth CE then, $\omega = 0.7$.
2: **while** Stopping Criterion **do**
3: **Adaptive updating of** γ_i:
Draw samples: $\boldsymbol{X}_1, ..., \boldsymbol{X}_N \sim Bernoulli(\boldsymbol{p}_{i-1})$;
Evaluate (4.4) at $\boldsymbol{X}_j \quad \forall j = 1, 2, ..., N$;
Define the order statistics for $S(\boldsymbol{X}_j)$: $S_{(1)} \leq ... \leq S_{(N)}$;
Update $\hat{\gamma}_i := S_{(\lceil \rho N \rceil)}$, where $\hat{\gamma}_i$ (sample percentile) is an estimator of the $(100\rho)^{th}$ population percentile.
4: **Adaptive updating of** $\boldsymbol{p}_i$:
a: *Hard Updating Rule.* Using (4.7):

$$p_{i,t} = \frac{\sum_{j=1}^{N} \mathbb{1}_{\{S(\boldsymbol{X}_j) \leq \hat{\gamma}_i\}} X_{jt}}{\sum_{j=1}^{N} \mathbb{1}_{\{S(\boldsymbol{X}_j) \leq \hat{\gamma}_i\}}} \quad \forall X_{jt} \in \boldsymbol{X}_j, \forall j = 1, 2, ..., N, \forall t = 1, \ldots, T$$

b: *Smooth Updating Rule*:

$$p_{i,t}^{s} = \omega p_{i,t} + (1 - \omega) p_{i-1,t}^{s}$$

5: $t := t + 1$
6: **end while**

Output with Hard Updating: Given F is the final iteration (all elements of $\boldsymbol{p}_F$ has converged to either 0 or 1), generate one final sample, say $\boldsymbol{X}^F$ from Bernoulli($\boldsymbol{p}_F$) of size T. $\boldsymbol{X}^F = \hat{\boldsymbol{x}}^{\star}$ is an approximation of the solution in (4.4).

Output with Smooth Updating: Given F is the final iteration (for example, if $\hat{\gamma_F} = \hat{\gamma}_{F-1} =, ..., = \hat{\gamma}_{F-5}$) generate one final sample, say $\boldsymbol{X}^F$ from Bernoulli($\boldsymbol{p}_F$) of size T. $\boldsymbol{X}^F = \hat{\boldsymbol{x}}^{\star}$ is an approximation of the solution in (4.4).

follows:

1. Choose a constant M and the error margin $\boldsymbol{v} = (v_0, \ldots, v_q)$ for the intercept and the autoregressive coefficients and assume that $\mid \hat{\phi}_l^{\star(1)} - \phi_l^{(1)} \mid < v_l$ and $\mid \hat{\phi}_l^{\star(2)} - \phi_l^{(2)} \mid < v_l$ for $l = 0, \ldots, q$, where $\hat{\phi}_l^{\star}$ is the solution in (4.6) using the solution provided by the CE $\hat{\boldsymbol{x}}^{\star}$ and ϕ_l are true underlying coefficients of the OLTAR model in (4.2).
2. For each time series observation $(y_t, 1, y_{t-1}, \ldots, y_{t-q})$, compute the fitted values $r_{t1} = \hat{\boldsymbol{\phi}}^{(1)}\boldsymbol{y}_{t-1}$ and $r_{t2} = \hat{\boldsymbol{\phi}}^{(2)}\boldsymbol{y}_{t-1}$, and compute two residuals $e_{tk} = y_t - r_{tk}$ for each observation where for $k = 1, 2$ and for $t = 1, \ldots, T$. Moreover, the maximum distance between the true and estimated regression surfaces on each point is given by $E_t = 2\left(\sum_{l=1}^{q} v_l |y_{t-l}| + v_0\right)$.
3. Finally, define two sets:

$$T_1 = \{t : e_{t1}^2 < e_{t2}^2, \mid r_{t1} - r_{t2} \mid > M + E_t\} \tag{4.9}$$

$$T_2 = \{t : e_{t2}^2 < e_{t1}^2, \mid r_{t1} - r_{t2} \mid > M + E_t\}, \tag{4.10}$$

and compute

$$\hat{\boldsymbol{\phi}}_1 = \Big(\sum_{t \in T_1} \boldsymbol{y}_{t-1}\boldsymbol{y}_{t-1}'\Big)^{-1} \sum_{t \in T_1} \boldsymbol{y}_{t-1} y_t, \quad \text{and} \tag{4.11}$$

$$\hat{\boldsymbol{\phi}}_2 = \Big(\sum_{t \in T_2} \boldsymbol{y}_{t-1}\boldsymbol{y}_{t-1}'\Big)^{-1} \sum_{t \in T_2} \boldsymbol{y}_{t-1} y_t, \tag{4.12}$$

where $\hat{\boldsymbol{\phi}}_1$ and $\hat{\boldsymbol{\phi}}_2$ are parameter vectors obtained using T_1 and T_2, respectively.

Chen (1995) shows that the estimators obtained using the above algorithm are consistent if the constant M tends to infinity. In other words, allowing M to increase as the sample size increases reduces the rate of misclassified states and hence, the bias introduced by them. This algorithm of Chen (1995) is perfectly applicable after obtaining the sequence of the state variables using the CE method.

However, in this study, we propose a new *CE-based selection algorithm* (Algo 4.2), for which we conjecture consistent estimation of the OLTAR parameters. Our algorithm for choosing the most important states is given below:

1. Minimize the loss function in (4.4) using the smooth CE method and obtain an approximation of the optimal state vector $\hat{\boldsymbol{x}}^{\star}$ and $\boldsymbol{p}_F = (p_{F,1}, \ldots, p_{F,T})$ (the optimal parameter vector of the sampling distribution at the final iteration). Then, compute

a new vector:

$$\tilde{\boldsymbol{p}} = \boldsymbol{p}_F \odot (\mathbf{1} - \boldsymbol{p}_F) = (\tilde{p}_1, \ldots, \tilde{p}_T),$$

where $\odot$ is the element-wise Hadamard product. $\tilde{\boldsymbol{p}}$ are simply the variances of the random variables under the Bernoulli sampling distribution of the algorithm at the final iteration.

2. Choose $\lceil \lambda T \rceil$ smallest elements of the $\tilde{\boldsymbol{p}}$ vector where $\lambda \in (0, 1)$ is the trim parameter determining the fraction of the retained observations and obtain their time index as:

$$\mathcal{S} = \left\{ t \in \{1, 2, \ldots, T\} : \tilde{p}_t \leq \tilde{p}_{(\lceil \lambda T \rceil)} \right\} \tag{4.13}$$

where $\tilde{p}_{(\lceil \lambda T \rceil)}$ is the $\lceil \lambda T \rceil$-th order statistic of the T variances.

3. Estimate the model parameters using:

$$\hat{\boldsymbol{\phi}}_k = \Big(\sum_{t \in \mathcal{S}} \mathbb{1}_{\{\hat{x}_t^\star = k\}} \boldsymbol{y}_{t-1} \boldsymbol{y}_{t-1}' \Big)^{-1} \sum_{t \in \mathcal{S}} \mathbb{1}_{\{\hat{x}_t^\star = k\}} \boldsymbol{y}_{t-1} y_t, \quad \text{for } k = 1, 2. \tag{4.14}$$

The selected points will be those with the lowest variance, corresponding intuitively to areas where there is minimal uncertainty about the regime of the data points.

Algorithm 4.2 CE-based selection algorithm for consistent estimation of OLTAR parameters

1: Minimize the loss function in (4.4) using the smooth CE method to obtain $\hat{\boldsymbol{x}}^\star$ and $\boldsymbol{p}_F = (p_{F,1}, \ldots, p_{F,T})$
2: Compute $\tilde{\boldsymbol{p}} = \boldsymbol{p}_F \odot (\mathbf{1} - \boldsymbol{p}_F) = (\tilde{p}_1, \ldots, \tilde{p}_T)$
3: Choose $\lceil \lambda T \rceil$ smallest elements of $\tilde{\boldsymbol{p}}$
4: Obtain the set $\mathcal{S}$ as:
5: $\mathcal{S} = \left\{ t \in \{1, 2, \ldots, T\} : \tilde{p}_t \leq \tilde{p}_{(\lceil \lambda T \rceil)} \right\}$
6: Estimate model parameters:
7: $\hat{\boldsymbol{\phi}}_k = \Big(\sum_{t \in \mathcal{S}} \mathbb{1}_{\{\hat{x}_t^\star = k\}} \boldsymbol{y}_{t-1} \boldsymbol{y}_{t-1}' \Big)^{-1} \sum_{t \in \mathcal{S}} \mathbb{1}_{\{\hat{x}_t^\star = k\}} \boldsymbol{y}_{t-1} y_t, \quad \text{for } k = 1, 2.$

4.5 The CEDT Algorithm's Searching Stage: Modeling The Regime Dynamics Using The CART

This section deals with the estimation of the regime dynamics which implies both selecting a relevant set of threshold variables and fitting the threshold function $g(.)$. To this end, we propose to use the CART algorithm. This section is then organized as follows. First, the CART algorithm and the motivations to use it are described. Then, we explain how the decision tree algorithm is used into our CEDT algorithm to learn the regime dynamics of the OLTAR models.

4.5.1 CART Algorithm

Parametric and semi-parametric models have been used extensively in statistical analysis. However, as noted by Zhang and Singer (2010) these models are very sensitive to even minor departures from the underlying assumptions. While diagnostic checks such as residual plots are suggested in the literature, the authors point out that such checks become less useful as model complexity increases. Furthermore, they argue that model interpretation can become problematic with non-linear interactions among potent regressors. Therefore, non-parametric methods have been developed to relax or remove restrictive assumptions of parametric models, as emphasized by the authors. In this work, we focus on decision tree algorithms, and more specifically on one of the best-known decision tree algorithms, the CART algorithm (Breiman et al., 1984). For simplicity's sake, we'll also refer to it as a decision tree in the following text, although many other algorithms to build decision trees exist today. Decision trees can be used both for regression problems (where the outcome variable is continuous) and classification problems (where the outcome is a discrete categorical variable). This completely non-parametric algorithm is also famous because it is easy to interpret and to implement (Hastie et al., 2001). Moreover, decision trees benefit from a wide applicability, and so they have been applied to numerous and various problems.

Here, as we are interested in estimating OLTAR model and that we will only describe the case where there are only two stages (4.3), we focus on classification trees with a binary response variable. Nonetheless, decision trees can handle response variables with multiple categories and so our approach can be also used easily for OLTAR models with more than two stages. To describe the CART algorithm principle, we first need to define

some additional notations. Note that the choice of notation was made to be consistent with the OLTAR model definition (4.3). Let Z be an outcome variable taking values 0 or 1, and $Y = \{Y_1, Y_2, \ldots, Y_q\}$ be a set of discrete or continuous variables referred to as feature space. For instance, following the context of our article, Z might represent the volatility regimen (high or low) of a given financial market, while Y could include financial and geopolitical features, for instance. To predict the response variable, the CART algorithm consists in building a partition of the feature space. This partition is constructed iteratively by means of binary splits. Starting with the whole available dataset, the algorithm splits the features space into two regions, referred to as nodes, and repeat this procedure on the two resulting nodes. The splitting process is applied on each node until some stopping criteria are reached or until the node is pure (i.e. all samples in the node share the same label). Each split is defined according to the values of one feature. The choice of the optimal split is generally based on the maximization of the change in an impurity function. Indeed, the algorithm aims to divide the feature space into increasingly pure nodes. At the end of the splitting process, the nodes that are not split are called the terminal nodes or the leaves. They define a partition of the feature space that can be displayed as a tree. The classification rule is deduced from this partition. Precisely, in each terminal node, the samples are assigned to the most-represented class label in the terminal node. The resulting partition is often not optimal with respect to a given criterion. Figure 4.1 provides an illustration. Consequently, a pruning strategy is often applied to select an optimal tree, see for instance Breiman et al. (1984) that introduces the cost-complexity pruning, one of the most used pruning strategy. Then, the CART algorithm consists of two important steps: growing and pruning.

Tree Growing

This step is iterative. It consists in repeating the following process for each resulting node, until reaching some stopping rules and until all nodes are pure. Consider a node τ that we want to split. to do so, the algorithm will search for the optimal split which is defined by a splitting feature and a threshold value for this feature. Then, the optimal split $(y_j^\star, s_j^\star)$ are the couple that maximizes the decrease in impurity. Equivalently, the

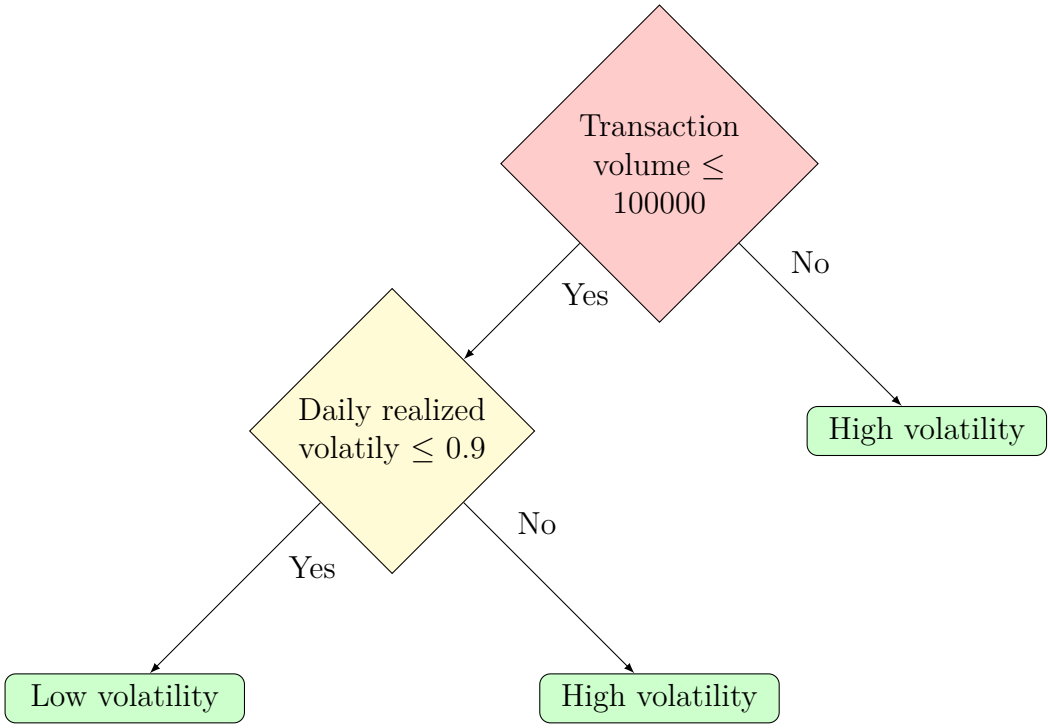


Figure 4.1 – A simple decision tree illustration. The resulting classification rule depends on only two features, namely the transaction volume and the daily realized volatility. Diamond shapes represent the non-terminal nodes, while rectangles represent the terminal nodes. The text inside the non-terminal nodes corresponds to the splitting rule while it is the classification rule in the terminal nodes.

optimal couple $(y_j^\star, s_j^\star)$ is the solution of the following problem

$$(y_j^*, s^*) = \underset{y_j \in y, s \in S_j}{\operatorname{argmax}} \Delta \mathrm{I}(\tau, y_j, s) \tag{4.15}$$

where $\Delta \mathrm{I}(\tau, y_j, s)$ is the impurity gain (or impurity decrease) for splitting using the couple (y_j, s_j). The impurity gain corresponds to the difference between the impurity in the parent node τ and the weighted sum of impurities in the child nodes, denoted by $I(\tau_{\text{left}})$ and $I(\tau_{\text{right}})$:

$$\mathrm{IG}(\tau, y_j, s) = I(\tau) - \left(\frac{N_{\text{left}}}{N} I(\tau_{\text{left}}) + \frac{N_{\text{right}}}{N} I(\tau_{\text{right}}) \right)$$

where:

— $I(\tau)$ is the impurity of the parent node

— $I(\tau_{\text{left}})$ and $I(\tau_{\text{right}})$ are the impurities of the left and right child nodes obtained after splitting according to the splitting couple (y_j, s_j),

— N is the total number of observations in the parent node $^{\tau}$,

— N_{left} and N_{right} are the number of observations in the left and right child nodes.

The optimization in (4.15) is a discrete optimization problem due to the finite number of possible split points. More precisely, the number of distinct observed values of variables in a dataset is finite, irrespective they are discrete or continuous. Therefore, the search for the optimal split can be performed exhaustively. Moreover, there are different impurity measures I in literature to be used within (4.15). The most widely used impurity measures are Gini index and entropy (Hastie et al., 2001). In this work, we use the entropy criterion due to its good properties stated in the related literature (Zhang and Singer, 2010). In binary classification, the entropy of a node τ is simply defined as:

$$I(\tau) = -p\log(p) - (1-p)\log(1-p),$$

where $p = \mathbb{P}\{Z = 1|\tau\}$ is the conditional probability of an observation within the node τ belonging to the class label 1.

Tree Pruning

At the end of the tree building process, we obtain a fully grown tree that can be very large and so quite complex. Most of the times, this maximal tree overfits the training data: the resulting model perfectly explains the training data but performs poorly on new samples. Therefore, to overcome this problem, pruning strategy is used to select a more generalizable model. Cost-complexity pruning Breiman et al. (1984), is one of the most used methods for this purpose. This strategy is based on the cost-complexity criterion. Let consider a non-trivial tree $\mathcal{T}$, the cost-complexity criterion for this tree is defined as:

$$R_\alpha(\mathcal{T}) = R(\mathcal{T}) + \alpha\left|\tilde{\mathcal{T}}\right|, \tag{4.16}$$

where $R(\mathcal{T})$ is the misclassification error of the tree $\mathcal{T}$(computed on the training set), $\left|\tilde{\mathcal{T}}\right|$ is the number of terminal nodes of $\mathcal{T}$, and $\alpha \geq 0$ is the positive real corresponding to the complexity parameter. It is noteworthy that any non-terminal node τ can be treated as a (sub-)tree and then, $R_\alpha(\tau)$ is the cost-complexity of that node. Breiman et al. (1984) proves that for all non-trivial tree $\mathcal{T}$, there exists a finite and unique increasing sequence of optimal values for α,

$$\alpha_0 = 0 < \alpha_1 < \cdots < \alpha_K,$$

$K \in \mathbb{N}^{\star}$, associated to a unique nested sequence of optimal subtrees

$$\mathcal{T} \subseteq \mathcal{T}_0 \subset \mathcal{T}_1 \subset \cdots \subset \mathcal{T}_K$$

such that for every $1 \leq k < K$,

$$\forall\, \alpha \in [\alpha_k; \alpha_0[, \quad \mathcal{T}_k \;=\; \underset{\mathcal{T}' \preceq \mathcal{T}_0}{\operatorname{argmin}}\, R_\alpha(\mathcal{T}'), \text{ and}$$

$$\forall\, \alpha \geq \alpha_K, \quad \mathcal{T}_K \;=\; \underset{\mathcal{T}' \preceq \mathcal{T}_{k-1}}{\operatorname{argmin}}\, R_\alpha(\mathcal{T}').$$

Therefore, the cost-complexity pruning method consists in constructing the sequence of optimal subtree of $\mathcal{T}_0$. Next, it remains to select the final tree among a sequence of optimal subtrees of $\mathcal{T}_0$ by minimizing a performance criterion (for instance the classification error) by cross-validation or by using an independent set. To search for the optimal sequence of subtrees, Breiman suggests a quite clever algorithm (see Algorithm 4.3). The algorithm starts with the fully grown tree $\mathcal{T}_0$ which is usually the optimal tree and the complexity parameter $\alpha_0 = 0$. Then, to calculate the next complexity parameter $\alpha_1 > \alpha_0 = 0$ and the corresponding optimal subtree $\mathcal{T}_1 \subset \mathcal{T}_0$ first we compute the cost-complexity for all non-terminal nodes $\{\tau : \tau \in \mathcal{T}_0 \text{ and } \tau \notin \tilde{\mathcal{T}}_0\}$ where $\tilde{\mathcal{T}}_0$ is the set of the terminal nodes for $\mathcal{T}_0$. Then, we compute the cost-complexity $R(\tilde{\mathcal{T}}_\tau)$ for the offspring terminal nodes $\tilde{\mathcal{T}}_\tau$ of each internal node τ. Then, the cost-complexity parameter for any internal node $\tau \in \mathcal{T}_0$, is calculated as $\alpha(\tau) = \frac{R(\tau) - R(\tilde{\mathcal{T}}_\tau)}{|\tilde{\mathcal{T}}_\tau| - 1}$. The smallest positive $\alpha(\tau)$ over all internal nodes of $\mathcal{T}_0$ is denoted by α_1. Finally, the optimal subtree $\mathcal{T}_1$ corresponding to α_1 is the collection of all internal nodes of $\mathcal{T}_0$ for which $R(\tau) + \alpha_1 \leq R(\tilde{\mathcal{T}}_\tau) + \alpha_1 |\tilde{\mathcal{T}}_\tau|$. The process is continued until reaching the root node which results in a sequence of $\alpha_0 > \alpha_1 > \ldots, \alpha_K$ and the corresponding optimal subtrees $\mathcal{T}_0 \supset \mathcal{T}_1 \supset \ldots \supset \mathcal{T}_K$.

In the following section, we explain how the CART algorithm is applied to learning the regime dynamics in the two-state OLTAR model.

4.5.2 Learning The Threshold Function Using Decision Trees

Principle

As mentioned previously, we apply the CART algorithm to learn the unknown threshold function from the time series dataset as opposed to Chen (1995) who uses a graphical approach. To that end, first, we partition the dataset at hand into three consecutive across

Algorithm 4.3 Cost-complexity Algorithm For Decision Tree Pruning

Initialize $\mathcal{T}_0$ as the fully saturated tree, iteration counter $k = 0$, complexity $\alpha_0 = 0$
while $\mathcal{T}_k$ has more than one node i.e. $|\mathcal{T}_k| > 1$ **do**
  for all $\tau \in \mathcal{T}_k$ and $\tau \notin \tilde{\mathcal{T}}_k$ **do**
    $\alpha(\tau) \leftarrow \frac{R(\tau) - R(\tilde{\mathcal{T}}_\tau)}{|\tilde{\mathcal{T}}_\tau| - 1}$
  end for
  $\alpha_{k+1} \leftarrow \min\{\alpha(\tau) > 0 : \tau \in \mathcal{T}_k \text{ and } \tau \notin \tilde{\mathcal{T}}_k\}$
  $\mathcal{T}_{k+1} \leftarrow \{\tau \in \mathcal{T}_k : \tau \notin \tilde{\mathcal{T}}_k \text{ and }$
    $R(\tau) + \alpha_{k+1} \leq R(\tilde{\mathcal{T}}_\tau) + \alpha_{k+1}|\tilde{\mathcal{T}}_\tau|\}$
  $k \leftarrow k + 1$
end while

time parts: a training set ($\mathcal{S}_{\text{train}}$), a validation set ($\mathcal{S}_{\text{valid}}$), and a test set ($\mathcal{S}_{\text{test}}$).

Then, we apply our CEDT method and so we start with the classification stage. Then, we apply the CE method to $\mathcal{S}_{\text{train}}$ to estimate the sequence of states $\{x_t\}$ and apply the CE-based selection (Algorithm 4.2) to choose the data points with minimal uncertainty about their regime. A trim parameter λ is key for selection of the most informative state estimates and reflects the variance-bias dilemma for the OLS estimators in (4.14). The subset of the training data, $\mathcal{S}_{\text{retained}}$, which contains $\lceil \lambda |\mathcal{S}_{\text{train}}| \rceil$ observations, is selected based on the rule defined in equation (4.13) and on Algorithm 4.2.

The second stage of the CEDT algorithm consists in learning the threshold function. To do so, we fit a maximal decision tree to the selected states obtained from the previous stage. Next, the fully grown tree is pruned based on Algorithm 4.3. After that, we select the final tree among the nested sequence of optimal subtrees based on the validation set. Specifically, the final tree corresponds to the subtree of the sequence that minimizes the classification error evaluated on the validation set.

Finally, to assess the final tree's performance, we predict states for the test set $\mathcal{S}_{\text{test}}$ using $\mathcal{T}^*$ and calculate the accuracy score for the test set. The details of validation and prediction accuracy for decision trees within the CEDT framework are outlined in Algorithm 4.4.

Variable Importances and Threshold Variables

A valuable byproduct of the tree-growing process is the calculation of variable (feature) importance, which allows to rank features according to the strength of their association with the response variable. Breiman et al. (1984) introduces an importance metric for the

Algorithm 4.4 Time Series Validation And Prediction Accuracy for Decision Tree

Require: Training set $\mathcal{S}_{\text{train}}$, Validation set $\mathcal{S}_{\text{valid}}$, Test set $\mathcal{S}_{\text{test}}$, Trim parameter λ
Ensure: Optimal complexity parameter α^*, Optimal subtree $\mathcal{T}^*$

$\mathcal{S}_{\text{retained}} \leftarrow$ Subset of $\mathcal{S}_{\text{train}}$ with $\lceil \lambda |\mathcal{S}_{\text{train}}| \rceil$ observations
Estimate states for $\mathcal{S}_{\text{retained}}$ using CE algorithm
Train fully grown decision tree $\mathcal{T}_0$ on $\mathcal{S}_{\text{retained}}$
Generate sequence of pruned subtrees $\{\mathcal{T}_i\}$ and complexity parameters $\{\alpha_i\}$ using cost-complexity pruning on $\mathcal{T}_0$ as shown in Algorithm 4.3
Initialize empty list scores
for each $(\alpha_i, \mathcal{T}_i)$ pair **do**
    Train decision tree $\mathcal{T}_i$ with complexity parameter α_i on $\mathcal{S}_{\text{retained}}$
    Predict states for $\mathcal{S}_{\text{valid}}$ using $\mathcal{T}_i$
    Calculate validation score s_i (accuracy)
    scores.append(s_i)
end for
$\alpha^* \leftarrow \text{argmax}_{\alpha_i}\{s_i\}$
$\mathcal{T}^* \leftarrow$ Subtree corresponding to α^*
Predict states for $\mathcal{S}_{\text{test}}$ using $\mathcal{T}^*$
Calculate test accuracy
return α^*, $\mathcal{T}^*$, test accuracy

CART tree that is based on the decreased in node impurity.

The importance of a variable is measured by its contribution to reducing the overall uncertainty of the model. Specifically, the entropy-based importance of a variable in a tree $\mathcal{T}$ is computed as the weighted average of the information gain (decrease in impurity) obtained by splitting nodes $\tau \in \mathcal{T}$ using that variable (Breiman et al., 1984).

Although this index has some weaknesses (see Strobl et al. (2007)) and so must be used with caution. This index can provide us with some additional information on the relevant variables. Indeed, this index can help us to identify the key factors influencing the threshold function, providing valuable insights into the underlying dynamics of the time series.

4.6 Simulation Studies

In this section, we study the performance of the proposed CEDT algorithm through two sets of experiments performed on synthetic data. The following subsection presents the simulation protocol.

4.6.1 Synthetic data

To evaluate our proposed algorithm, we use several scenarios. In each scenario, the data are modeled according to an OLTAR model. We consider two distinct OLTAR models denoted by M1 and M2 and defined as follows:

$$y_t = \epsilon_t + \begin{cases} 1 + 0.5y_{t-1} & \text{if} \quad x_t = 1 \\ 2 - 0.9y_{t-1} & \text{if} \quad x_t = 2 \end{cases} \tag{M1}$$

$$y_t = \epsilon_t + \begin{cases} 0.15 + 0.5y_{t-1} & \text{if} \quad x_t = 1 \\ -0.15 - 0.9y_{t-1} & \text{if} \quad x_t = 2 \end{cases} \tag{M2}$$

M1 corresponds to the "easy case" while M2 corresponds to the "sticky case". Indeed, when the intercepts in the equations of the two states are further from each other, it is easier to separate observations generated by the two states (Chen, 1995). On the contrary, it is more difficult to separate observations when the intercepts in the equations are closer.

Further, several threshold functions are considered to model the state dynamic. driving the state variable x_t:

$$x_t = \begin{cases} 1 & \text{if} \quad 3y_{t-3} - 2y_{t-2}^2 \leq 0 \\ 2 & \text{if otherwise} \end{cases} \tag{T1}$$

$$x_t = \begin{cases} 1 & \text{if} \quad y_{t-3} - 1 \leq 0 \cap y_{t-2} - 1 \leq 0 \\ 2 & \text{if otherwise} \end{cases} \tag{T2}$$

$$X_t = \begin{cases} 1 & \text{if} \quad y_{t-2} - 1 \leq 0 \\ 2 & \text{if otherwise} \end{cases} \tag{T3}$$

Here, x_t is the state at time t, and ϵ_t is a normally distributed zero-mean error term with standard deviation $\sigma = 0.5$ across all simulations. The shape of the relationship between features is different for each of the three threshold functions. We chose to use these threshold functions to check whether the CART algorithm can learn the true threshold function, even when this involves complex interactions between features.

Each scenario is denoted by MiTj, where $i = \{1, 2\}$ and $j = \{1, 2, 3\}$, resulting in six scenarios: M1T1, M1T2, M1T3, M2T1, M2T2, and M2T3. In stage 2, we aim at estimate the threshold function, we used the CART algorithm.

As explained in Section 4.5.2, the CART algorithm is applied on the subset of the

training data, $\mathcal{S}_{\text{retained}}$, which contains $\lceil \lambda | \mathcal{S}_{\text{train}} | \rceil$ observations selected based on the rule defined in equation (4.13) and on Algorithm 4.2. To approximate the threshold function using a decision tree, we provide 20 features (for each model) to the CART algorithm. These features are displayed in Table 4.1. The readers might be surprised that the decision trees are not constructed on the original elementary features. Here, we aim to assess the ability of the algorithm to identify the true threshold variables. That's why we choose to include the true expression of the threshold function and some other constructed features as features for the decision tree. Nonetheless, as explained before, the CART algorithm is, among other motivations, used in the CEDT algorithm because it is a fully non-parametric method and it can estimate complex relationships.

$3y_{t-3} - 2y_{t-2}^2$	y_{t-1}
y_{t-2}	y_{t-3}
y_{t-4}	$y_{t-1}^3 - 13y_{t-2} + 14$
$y_{t-2}^2 - 11y_{t-3} + 77$	$y_{t-3}^3 - 17y_{t-4} + 66$
$5U[1,0]$	$2N(0,1) + 3U[0,1]$
$y_{t-1}^4 + 2y_{t-1} + 8N(0,1)$	$y_{t-3}^4 + 3y_{t-3} + 4N(0,1)$
$y_{t-1}^3 + 4y_{t-1} + 3N(0,1)$	$y_{t-2}^2 + 5y_{t-3} + 6N(0,1)$
$y_{t-3}^2 + 6y_{t-3} + 7N(0,1)$	$y_{t-2}^2 + 7y_{t-2} + 4N(0,1)$
$y_{t-3}^3 + 8y_{t-3} + 8N(0,1)$	$y_{t-3}^3 + 9y_{t-3} + 3N(0,1)$
$y_{t-3}^3 + 10y_{t-3} + 2N(0,1)$	$y_{t-2}^2 + 11y_{t-2} + 4N(0,1)$

Table 4.1 – List of features used in the CART algorithm

4.6.2 Simulation I: Algorithm performance through several synthetic scenarios

Each model is simulated 50 times with different trajectories of ϵ_t. The CEDT algorithm is applied to each realization of the time series for all 6 models. The length of each simulated time series is set to 400 observations, divided as follows:

— *Training set* ($\mathcal{S}_{\text{train}}$): The first 100 observations, used to obtain estimates of states and model parameters via the CEDT algorithm, and obtain the set of complexity parameters and optimal subtrees as delineated in Algo 4.3.

— *Validation set* ($\mathcal{S}_{\text{valid}}$): The next 100 observations following the training set are used to choose the optimal complexity parameter and subtrees.
— *Test set* ($\mathcal{S}_{\text{test}}$): The remaining 200 observations, used to test prediction accuracy.

The trim parameter λ is set to 0.4, meaning that only 40% of $\mathcal{S}_{\text{train}}$ observations are retained for use within the DT algorithm training and pruning. These observations are referred to as the **retained set** ($\mathcal{S}_{\text{retained}}$), where $\mathcal{S}_{\text{retained}} \subseteq \mathcal{S}_{\text{train}}$. Moreover, we follow recommendations from (Rubinstein and Kroese, 2004; de Boer et al., 2005) to tune the CE algorithm hyperparameters. We use $\omega = 0.7$ for the smoothing parameter. We draw $N = 5T$ samples at each iteration (with T denoted the time series length) and we set the value of the percentile of elite sample ρ to 10%.

To assess the performance of the CEDT algorithm on each scenario, we used the following metrics:

1. *Classification stage (CE) accuracy*: The percentage of correctly classified states in $\mathcal{S}_{\text{retained}}$ averaged across 50 realizations of the artificial time series. We sometimes refer to that as the *stage 1* accuracy within plots and tables for brevity purposes.
2. *Searching stage (DT) accuracy*: The percentage of correctly classified states in $\mathcal{S}_{\text{retained}}$ by the DT algorithm averaged across 50 realizations. Sometimes we refer to it as *in-sample accuracy* or *stage 2 accuracy* within plots and tables.
3. *Test accuracy*: The percentage of correctly predicted states in the test set ($\mathcal{S}_{\text{test}}$) averaged across 50 realizations. Sometimes we refer to it as *out-of-sample accuracy* within plots.
4. *True threshold variable selection frequency*: The percentage of simulations where the true threshold variable was selected by the DT algorithm as one of the splitting variables.
5. *True threshold variable best predictor frequency*: The percentage of simulations where the true threshold variable was selected by the DT algorithm as the most important splitting variable.

Table 4.2, Figure 4.2 and Figure 4.3 display simulation results for the six scenarios. Moreover, to address the relatively low accuracies of M2 models, we also tested a smaller trim parameter value ($\lambda = 0.2$). The results of this adjustment are shown in the last three rows of Table 4.2.

The last two columns of Table 4.2 highlight the ability of the CART algorithm to

Table 4.2 – Model Performance and Variable Selection

Model	Stage 1 (CE) Accuracy	Stage 2 (DT) Accuracy	Test Accuracy	Selected	Selected As The Best
M1T1	97.45	98.27	97.65	96	94
M1T2	99.60	99.56	99.41	98	98
M1T3	99.20	96.25	95.76	82	82
M2T1	91.55	94.17	93.32	96	80
M2T2	84.55	89.33	88.85	72	52
M2T3	84.95	90.73	89.56	66	64
M2T1 ($\lambda = 0.2$)	94.60	93.27	92.14	88	74
M2T2 ($\lambda = 0.2$)	93.50	94.81	94.66	84	72
M2T3 ($\lambda = 0.2$)	90.40	88.19	87.61	52	52

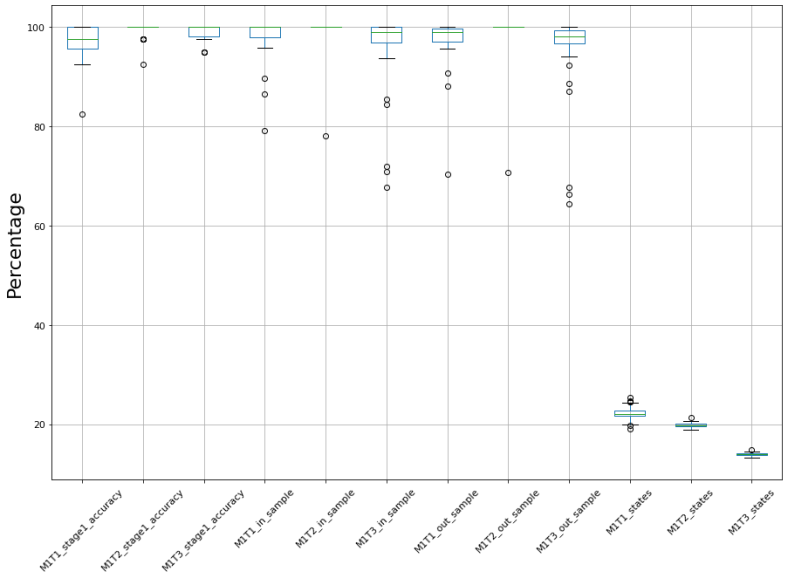


Figure 4.2 – Summary of simulation results for the “easy“ scenarios

learn the true threshold function for the entertained models. As shown in the table, for M1 model variants the true threshold function (variable) was selected as the best feature (the feature with the highest importance) by the CART algorithm in 94% (47 out of 50 cases), 98% (49 out of 50 cases), and 82% (41 out of 50 cases) of the cases, respectively. For the M2 variants this numbers are 80%, 52% and 64%, respectively. The 'Selected' column of the table shows the percentage of times the true threshold variable is either selected as the most important feature or has a feature importance greater than zero. In addition, as

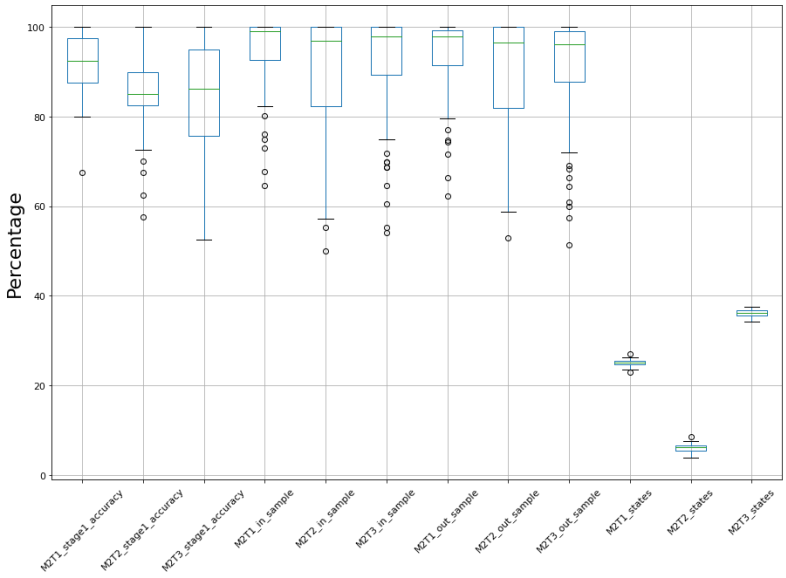


Figure 4.3 – Summary of simulation results for the "strichy" scenarios($\lambda = 0.4$)

(4.2) suggests, for M2T1, M2T2, and M2T3 models, the correct threshold function was not selected as often as in the previous cases for which the stage 1 accuracies are also relatively low compared to M1 model variants. The distributions of the corresponding accuracies for M2 model variants are given in Fig. 4.3.

Indeed, we see a higher variation inaccuracies for M2 models. We ran the same simulation to address the higher variability and lower average accuracy of M2 models, but this time by setting the trim parameter at $\lambda = 0.2$. In (4.4), the boxplots of accuracies for M2 model variants with a trim parameter $\lambda = 0.2$ are shown. Overall, we see some decrease in variation and increase in average of different accuracies, as was also shown in Table 4.2. The logic behind this is that the CEDT algorithm is a two-step procedure and its success depends on the first stage accuracy, and any errors in the first stage are

propagated to the second stage.

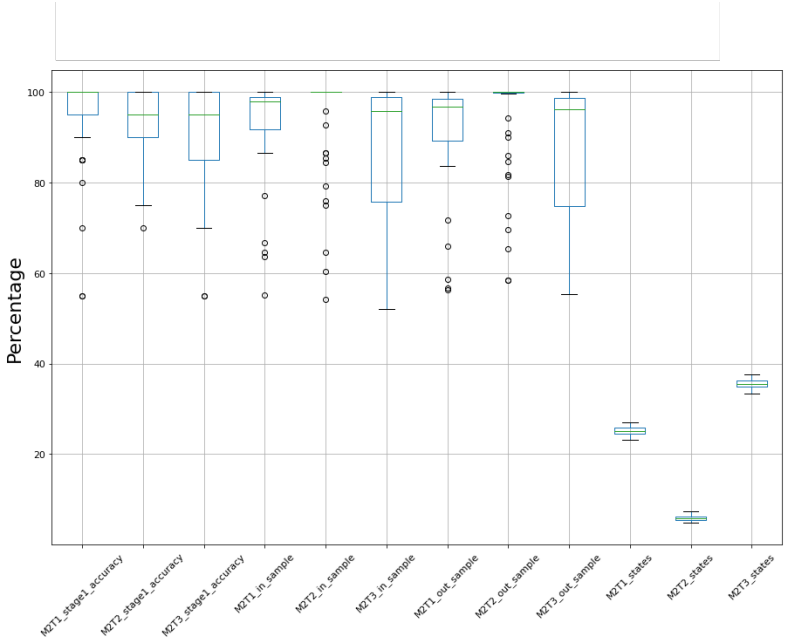


Figure 4.4 – Summary of simulation results for the "strichy" scenarios ($\lambda = 0.2$)

Finally, the last three boxplots in Figure 4.3 and Figure 4.4 display the distribution of the states. As the figures suggest, the datasets for all models are imbalanced, meaning that one of the two regimes happens much more frequently than the other. Our major concern in simulations was that if we apply the selection algorithm with small value of the trim parameter, then, in some of the simulation runs, observations on one of the two regimes may be completely dropped. This did not happen in this simulation study.

4.6.3 Simulation II: Algorithm performance over a longer time horizon

In this part of our study, we perform the following simulation. We simulate a time series of length 1000 from both models using T1 as threshold. Then, we choose $|\mathcal{S}_{\text{train}}| = 700$ and set $\lambda = 0.1, 0.5, 0.9$ to examine the impact of different threshold values on improving prediction accuracy and threshold selection. For the validation set $\mathcal{S}_{\text{valid}}$ and the test set $\mathcal{S}_{\text{test}}$, we use the same proportions as used in the previous simulation study, namely $\mathcal{S}_{\text{valid}} = 100$ and $\mathcal{S}_{\text{test}} = 200$.

Table 4.3 summarizes the results of the second simulation study, using the same names

and definitions as in Table 4.2. These results empirically confirm that decreasing the value of λ allows us to retain by the proposed CE-based selection method the states for which we are more certain of their values, as evidenced by the variable Stage 1 Accuracy. Although this reduces the amount of training data, the fact that the retained data contains fewer errors leads to better overall performance of the CEDT model. This is because the decision tree is trained on a smaller but more reliable set of states, improving its predictive ability.

Model	Stage 1 (CE) Accuracy	Stage 2 (DT) Accuracy	Forecast (CEDT) Accuracy	Selected	Selected As The Best
M1T1 ($\lambda = 0.9$)	80	88	83.5	Yes	Yes
M1T1 ($\lambda = 0.5$)	92	99	99.5	Yes	Yes
M1T1 ($\lambda = 0.1$)	99	100	100	Yes	Yes
M2T1 ($\lambda = 0.9$)	83	84	78.5	Not	Not
M2T1 ($\lambda = 0.5$)	99	99	96.5	Yes	Yes
M2T1 ($\lambda = 0.1$)	100	98	99.5	Yes	Yes

Table 4.3 – Summary of results for the M1T1 and M1T2 Models with fixed size of the training set ($\mid \mathcal{S}_{train} \mid = 700$) and with varying $\lambda = 0.1, 0.5, 0.9$.

4.7 OLTAR-HAR Model and its application to Bitcoin Realized Volatility

In this section, we propose an extension of the well-known HAR model of Corsi (2009), which we term the OLTAR-HAR model. We then apply it to the highly volatile bitcoin market, to demonstrate its ability to capture non-linear dynamics.

To estimate the OLTAR model, we use the CEDT algorithm. As highlighted previously, our new method, which shows good results on simulation studies, allows for overcoming the significant estimation challenges of the OLTAR models.

In the following subsections, we provide the OLTAR-HAR model specification, explain the data collection and the application of the CEDT algorithm to Bitcoin market data, and present the estimation results.

4.7.1 OLTAR-HAR Model

For the sake of simplicity, in this application, we consider an OLTAR-HAR model with two regimes. The HAR component of our model is based on the work of (Corsi, 2009),

which captures the heterogeneous nature of traders' behavior in financial markets. Our OLTAR-HAR model is defined as:

$$RV_{t+1} = \begin{cases} \beta_0^{(1)} + \beta_d^{(1)} RV_t^{(d)} + \beta_w^{(1)} RV_t^{(w)} + \beta_m^{(1)} RV_t^{(m)} + \epsilon_t^{(1)} & \text{if } X_t = 1 \\ \beta_0^{(2)} + \beta_d^{(2)} RV_t^{(d)} + \beta_w^{(2)} RV_t^{(w)} + \beta_m^{(2)} RV_t^{(m)} + \epsilon_t^{(2)} & \text{if } X_t = 2 \end{cases} \tag{4.17}$$

where $\{RV_t^{(d)}\}_{t\in\mathbb{N}^\star}$ is the time series of daily realized volatility, and t is the time (day) index. $RV_t^{(w)}$ and $RV_t^{(m)}$ represent the weekly and monthly averages of realized volatility, respectively, measured at daily frequency. In the literature, $RV_t^{(w)}$ is typically defined as a five-day moving average and $RV_t^{(m)}$ as a twenty-two-day moving average of daily realized volatility, corresponding to the number of working days in a week and a month, respectively (see (Bergsli et al., 2022) and the references therein). However, in our study, we use seven-day and thirty-day moving averages for weekly and monthly realized volatilities to account for the non-stop nature of crypto markets.

4.7.2 Data Preprocessing and Split

We obtained high-frequency Bitcoin price data from the Binance exchange for the period from January 1, 2023, to October 4, 2024, using (McHardy, 2023) which constitute 612 observations in total. The data was preprocessed to calculate daily realized volatility using 5-minute returns. Specifically, we computed the realized volatility as:

$$RV_t = \sum_{i=1}^{n} r_{t,i}^2,$$

where RV_t is the realized volatility on day t, n is the number of 5-minute intervals in a trading day ($n = 288$), and $r_{t,i}$ is the 5-minute log return.

Additionally, we obtain daily trade volume and daily closing prices (P_t) of Bitcoin from the Binance exchange for the same time period. We then calculate the daily logarithmic returns series r_t from the daily historical prices ($r_t = \ln P_t - \ln P_{t-1}$). Trade volume and returns are used as features to train the Decision Tree within the CEDT framework. The feature space for this particular example consists of the first lag of each of the following five variables: trade volume, returns, and daily, weekly, and monthly volatilities.

Additionally, we divide the time series dataset into three consecutive parts, each containing approximately 30% of the observations: (i) training, (ii) validation, and (iii) test

sets. The training set is used to fit the model, the validation set is employed to tune hyperparameters of the CEDT algorithm using unseen data, and the test set is utilized to generate volatility predictions and compare them to the benchmark.

It is worth mentioning that, in this study, we use a limited set of features to learn and predict the regime switching mechanism. While a larger feature space could potentially enhance the learning of hidden state dynamics, particularly for the highly volatile Bitcoin market, our primary objective is not to compete with other volatility models. Instead, we aim to demonstrate that even in this simplified setting — assuming a two-state OLTAR model and a small feature space — the CEDT algorithm can effectively learn time series dynamics and generate useful results.

4.7.3 Estimation Procedure using the CEDT algorithm

Stage I - Classification Stage

We use the smooth CE algorithm (Algo 4.1) to estimate the state variables for the training set. The algorithm's parameters are tuned as described in Section 4.6 with trim parameter $\lambda = 0.5$. Subsequently, model parameters are estimated using the selected subset of data using the OLS as described in Algo 4.2.

Stage II - threshold function estimation

After, we first fit a maximal decision tree to the subset of the training set previously denoted as $\mathcal{S}_{\text{retain}}$. Then, to select the final tree, we use the validation set. Note that we don't use the whole validation set in this application. Indeed, we apply the CE algorithm to the whole validation set in order to estimate the states and then we retain only the subset including the less uncertain observations with the same trim parameter value as used for the training set in Stage I. The reason is that, unlike experiments conducted on synthetic data, in this real-world application on the bitcoin market, we don't know the true states. Therefore, it seems more relevant to validate the decision tree parameters using the less uncertain state estimates from the validation set rather than the entire validation set.

In addition, feature importance is calculated on the basis of the final decision tree. As explained above, these indices, when non-zero, allow features to be ranked according to their discriminating power. The feature with the highest importance is generally considered to be the most discriminating feature in the decision tree and therefore in the

threshold function. Features with zero importance are considered as features not included in the definition of the threshold function.

In the next part of the work, we present some descriptive statistics and the estimation results.

Results and Interpretation

We applied the smooth CE method, as described in Algorithm 4.1, to estimate a two-state digression model for Bitcoin daily realized volatility data. This approach helps identify regime-switching patterns within the dataset. The CE estimates of the digression model regimes are illustrated in (4.5). Figure 4.5 displays the daily realized volatility of

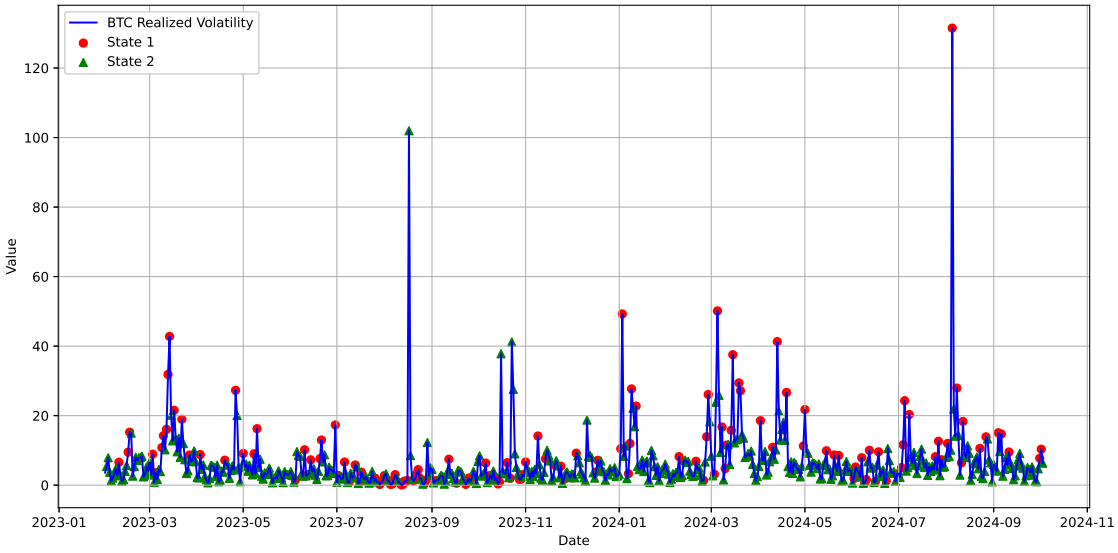


Figure 4.5 – Cross Entropy Estimation of Digression Model States for Bitcoin

Bitcoin over a specified period, depicted by the blue line. The red dots and green triangles represent two distinct regimes predicted by the digression model. Notably, the digression model attributed mainly high values of volatility to one regime and small values to the other one with a few exceptions. This is an indication of the multi-state nature of volatility documented by the related literature (Bauwens et al., 2012).

Summary statistics for the two regimes estimated by the CE method are provided in Table 4.4. The statistics further confirm the presence of distinct regimes, as evidenced by

significant differences in descriptive measures. Initial evidence about the distinct regimes

State	Value count	mean	std	min	25%	50%	75%	max
State 1	467.0	5.46	6.39	0.29	2.58	4.05	6.53	101.96
State 2	144.0	10.14	14.28	0.08	1.70	6.89	12.16	131.49

Table 4.4 – Summary Statistics of Bitcoin Volatility in Two Regimes

is further enhanced by the estimation results.

First, Table 4.4 highlights that the states estimated using the CE algorithm are imbalanced. More precisely, high volatility states seems rarer ($\approx$ 20%) compared to low volatility states ($\approx$ 80%).

Table 4.5 shows the OLS estimation output. As we have already discussed in previous sections, the OLS estimators of the model parameters within the CEDT algorithm are consistent, but their standard errors are inefficient due to the fact that a portion of observations are dropped. Therefore, making inferences about statistical significance is somewhat difficult.

As explained previously, the threshold variable is approximated using a decision tree. Figure 6.56 displays the resulting tree. Moreover, Figure 4.6 shows the importance indices for the features included in the decision tree and so the features driving the regime switches.

4.7.4 Comparison between OLTAR-HAR model and HAR model

We fit a simple HAR model to the retained observations in the training set used to OLTAR-HAR model. This is to have a fair comparison between the two. Then, we use both models to predict the daily realized volatilities in the test set. For OLTAR-HAR model, we first use the validated decision tree to predict the states of the test set, and based on those state predictions we predict the realized volatility in the test set. The regression outputs for both models are given side-by-side in Table 4.6. The results in Table 4.5 and Table 6.6 are copied to Table 4.6 to help the comparison.

We compare the predictive performance of the two models using both Mean Squared Error (MSE) and Mean Absolute Error (MAE) as metrics. The OLTAR-HAR model

Dep. Variable:	$RV_{t+1}^{(d)}$	**R-squared:**	0.722
Model:	OLS	**Adj. R-squared:**	0.701
Method:	Least Squares	**F-statistic:**	34.11
Date:	Mon, 07 Oct 2024	**Prob (F-statistic):**	5.85e-23
Time:	03:31:22	**Log-Likelihood:**	-273.62
No. Observations:	100	**AIC:**	563.2
Df Residuals:	92	**BIC:**	584.1
Df Model:	7		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
$\boldsymbol{RV_t^{(d)}\mathbb{1}_{\{x_t=1\}}}$	0.0822	0.041	2.016	0.047	0.001	0.163
$\boldsymbol{RV_t^{(d)}\mathbb{1}_{\{x_t=2\}}}$	1.2301	0.151	8.149	0.000	0.930	1.530
$\boldsymbol{RV_t^{(w)}\mathbb{1}_{\{x_t=1\}}}$	0.4270	0.120	3.545	0.001	0.188	0.666
$\boldsymbol{RV_t^{(w)}\mathbb{1}_{\{x_t=2\}}}$	-0.0816	0.345	-0.236	0.814	-0.767	0.604
$\boldsymbol{RV_t^{(m)}\mathbb{1}_{\{x_t=1\}}}$	0.0153	0.210	0.073	0.942	-0.403	0.433
$\boldsymbol{RV_t^{(m)}\mathbb{1}_{\{x_t=2\}}}$	1.0899	0.633	1.722	0.088	-0.167	2.347
$\mathbb{1}_{\{x_t=1\}}$	1.2216	1.381	0.885	0.379	-1.521	3.964
$\mathbb{1}_{\{x_t=2\}}$	-0.7433	2.109	-0.352	0.725	-4.932	3.445

Omnibus:	54.111	**Durbin-Watson:**	1.723
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	191.934
Skew:	1.863	**Prob(JB):**	2.10e-42
Kurtosis:	8.673	**Cond. No.**	83.6

Table 4.5 – Consistent Estimates of OLTAR-HAR Parameters Based on The Most Informative States

achieved an MSE of 73.98, outperforming the HAR model's MSE of 111.52. Additionally, the MAE for the OLTAR-HAR model was 3.24, compared to 4.77 for the HAR model.

The predictions from both models against actual realized volatility in the test set are illustrated in Figure 4.7. From the figure, it seems that the OLTAR-HAR model mimics the behavior of realized daily volatility much better.

To analyze model performance in more detail, we examine the prediction errors for both models. Figure 4.8 shows that on the Bitcoin data the OLTAR-HAR model not only has a lower median error, but the errors seem also more stable compared to the HAR model. Here, the OLTAR-HAR model seems to far outperform the HAR model. This application suggests that the OLTAR-HAR model seems more adapted, than the HAR-

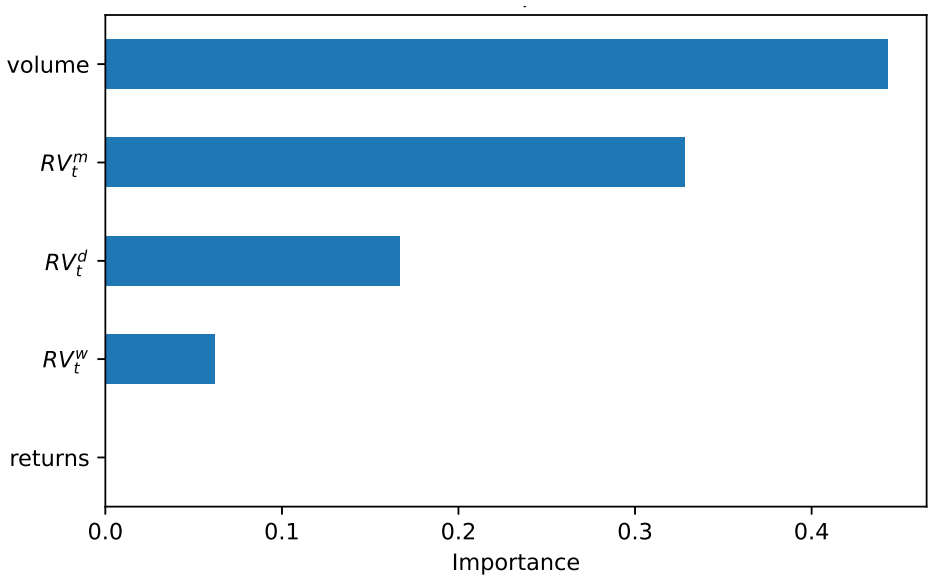


Figure 4.6 – Feature importance scores from the decision tree.

model, to capture non-linear dynamics in Bitcoin's realized volatility. Moreover, the use of decision tree to approximate the threshold function offers interpretability by identifying key features that influence regime changes.

4.8 Conclusion

In this chapter, we introduce a novel approach for estimating Open-Loop Threshold Autoregressive (OLTAR) models. This original method, that we call CEDT algorithm, uses both cross-entropy algorithm and decision trees. Our algorithm addresses key challenges in OLTAR modeling, namely the estimation of state variables and the selection of relevant threshold variables from a potentially large feature space.

This methodology combines the global optimization capabilities of the cross-entropy method for state estimation with the non-parametric nature of decision trees for capturing complex variable relationships. This combination allows for a more comprehensive exploration of the model space, potentially leading to more accurate and interpretable learning of OLTAR models.

Through numerical experiments, we empirically demonstrate the effectiveness of the CEDT algorithm in accurately estimating state variables and identifying true threshold variables across various model specifications. The algorithm shows good performance,

Table 4.6 – Comparison of a HAR and OLTAR-HAR Model On the Test Set

	Dependent variable: $RV_{t+1}^{(d)}$	
	(1)	(2)
$\mathbb{1}_{\{x_t=1\}}$		1.222
		(1.381)
$\mathbb{1}_{\{x_t=2\}}$		-0.743
		(2.109)
const	8.065**	
	(3.159)	
$RV_t^{(d)}$	0.062	
	(0.124)	
$\mathbb{1}_{\{x_t=1\}} RV_t^{(d)}$		0.082**
		(0.041)
$\mathbb{1}_{\{x_t=2\}} RV_t^{(d)}$		1.230***
		(0.151)
$RV_t^{(m)}$	-0.816	
	(0.574)	
$\mathbb{1}_{\{x_t=1\}} RV_t^{(m)}$		0.015
		(0.210)
$\mathbb{1}_{\{x_t=2\}} RV_t^{(m)}$		1.090*
		(0.633)
$RV_t^{(w)}$	0.617*	
	(0.364)	
$\mathbb{1}_{\{x_t=1\}} RV_t^{(w)}$		0.427***
		(0.120)
$\mathbb{1}_{\{x_t=2\}} RV_t^{(w)}$		-0.082
		(0.345)
Observations	100	100
R^2	0.058	0.722
Adjusted R^2	0.028	0.701
Residual Std. Error	11.691 (df=96)	3.892 (df=92)
F Statistic	1.960 (df=3; 96)	34.110*** (df=7; 92)
Note:		*p<0.1; **p<0.05; ***p<0.01

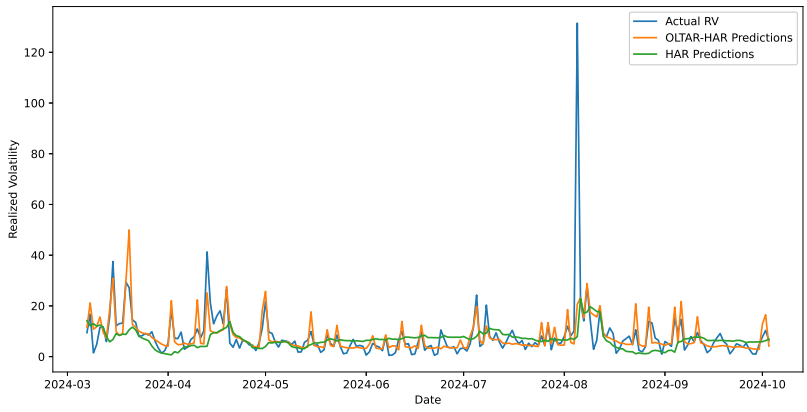


Figure 4.7 – Comparison of actual vs predicted realized volatility using OLTAR-HAR and HAR models.

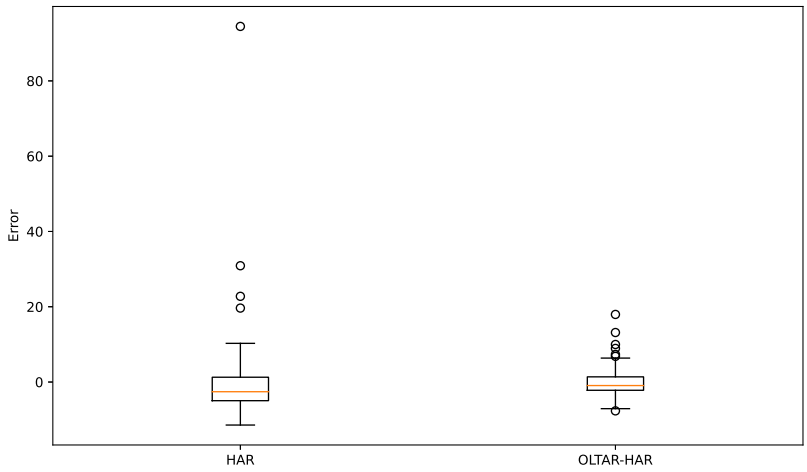


Figure 4.8 – Comparison of Errors of HAR and OLTAR-HAR Models

particularly for larger sample sizes and appropriate trim parameter settings.

As a practical application, we introduce the OLTAR-HAR model for volatility modeling, blending the TAR model framework with the popular Heterogeneous Autoregressive (HAR) model. When applied to Bitcoin realized volatility data, the OLTAR-HAR model outperforms the traditional HAR model in terms of predictive accuracy, as evidenced by lower Mean Squared Error (MSE) and Mean Absolute Error (MAE) metrics. The decision tree component of our algorithm provides valuable insights into the factors driving regime switches in Bitcoin volatility, identifying trade volume as the most important threshold variable. This interpretability, combined with improved predictive performance, highlights

the potential of the OLTAR-HAR model for capturing non-linear dynamics in financial time series.

While our study focused on a two-state OLTAR model with a limited feature space, future research could explore extensions to multiple regimes and larger feature sets. Additionally, investigating the theoretical properties of the CEDT algorithm, such as consistency and convergence rates, could further establish its place in the econometric toolkit for non-linear time series analysis.

In conclusion, the CEDT algorithm and the OLTAR-HAR model offer promising frameworks for modeling complex, regime-switching dynamics in time series data. Their ability to balance interpretability with predictive power makes them valuable tools for researchers and practitioners in finance, economics, and related fields dealing with non-linear time series processes.

Chapter 5

Conclusion

This doctoral thesis contributes to the field of financial modeling by developing advanced methodologies for conditional volatility forecasting, integrating machine learning techniques, and improving numerical optimization strategies. Through novel algorithms and the extension of existing models, the research offers tools for discrete-time volatility modeling, crucial for applications in risk management, hedging, and portfolio optimization in dynamic financial markets.

Enhancing GARCH Models with Variable Selection

Chapter 2 addresses the limitations of traditional GARCH models, particularly their rigidity in incorporating exogenous variables that may contain valuable information about asset volatility. To tackle the challenge of overfitting while preserving model flexibility, the thesis introduces the Variable Selection Log-TGARCHX (VS-LTGARCHX) algorithm. By integrating variable selection techniques such as LASSO, ABESS, and Boruta, this approach efficiently identifies the most relevant exogenous factors driving volatility. Applied to the volatile Bitcoin market, with an extensive set of 42 conditioning variables, the VS-LTGARCHX models demonstrated superior predictive performance compared to standard benchmarks. Robustness checks and the Model Confidence Set (MCS) procedure further validate the practical applicability of these models.

Advancing Clustering Techniques with CESBKM Algorithm

Building upon the work in Chapter 2, Chapter 3 explores clustering methodologies to capture regime-switching behaviors in financial time series. The thesis introduces the Cross Entropy with Symmetry Breaking and K-Means (CESBKM) algorithm, a hybrid approach combining the strengths of Cross Entropy and K-means algorithms. This novel algorithm addresses challenges in clustering problems related to symmetry and enhances convergence speed. Empirical results demonstrate that CESBKM outperforms the generic

Cross Entropy method in accuracy and efficiency. When applied to the Heterogeneous Autoregressive (HAR) model for Bitcoin volatility, CESBKM significantly improved model fit, showcasing its robustness and relevance for financial time series applications.

Innovating OLTAR Model Estimation with CEDT Algorithm

In Chapter 4, the thesis introduces the Cross Entropy with Decision Trees (CEDT) algorithm, proposing a new framework for estimating Open-Loop Threshold Autoregressive (OLTAR) models. By integrating the global optimization capabilities of the Cross Entropy method with the interpretability of Decision Trees, CEDT enables consistent state estimation and effective threshold variable selection from large feature spaces. Applied to the OLTAR-HAR model for Bitcoin realized volatility, CEDT demonstrated superior predictive accuracy and provided valuable insights into regime-switching factors. This balance of model interpretability and predictive performance underlines the potential of CEDT for nonlinear time series analysis.

Future research directions

While this thesis has introduced several novel algorithms and demonstrated their effectiveness in various contexts, there remain important avenues for further research. One key direction lies in expanding the set of exogenous variables used in the models. While the VS-LTGARCHX algorithm was applied with 42 conditioning variables, future work could explore larger, more diverse sets of predictors, including macroeconomic indicators, sentiment analysis, or market activity data. Expanding the pool of potential predictors could further enhance the model's predictive accuracy and extend its applicability to a broader range of financial assets.

Extending the proposed algorithms to multivariate settings, such as multivariate GARCH or Vector Autoregressive (VAR) models, presents another important future direction. A multivariate extension would allow for the modeling of interdependencies between multiple assets or markets, thereby enabling joint modeling of cross-market correlations and volatility spillovers. This would significantly improve portfolio optimization and risk management strategies, especially in highly interconnected markets.

In terms of theoretical contributions, the thesis already provides a detailed study of the convergence properties of the CESBKM algorithm in Chapter 3. However, further work could focus on establishing the consistency of the final estimator from the CEDT when using the CE-based point selection method (Algo 4.2 in Chapter 4). While convergence

has been extensively examined for CESBKM, proving the consistency of the regression coefficient estimator would further solidify the theoretical foundation of the approach. This would ensure that the estimator remains robust as the sample size grows, adding to the long-term reliability of the model. Such an investigation would enhance the practical and theoretical contributions of the CE-based selection method.

Beyond the current Bitcoin-focused applications, there is significant scope for applying the proposed methodologies to other financial markets, including equities, bonds, and commodities. Each market comes with its own unique volatility patterns, and applying these methods across various asset classes could lead to a broader generalization of the results and reveal market-specific dynamics. Furthermore, real-time applications of these models would be a valuable area for future development, particularly in high-frequency trading environments. Improving the computational efficiency and scalability of the proposed algorithms could enable their use in real-time forecasting and decision-making in fast-moving markets, where speed and accuracy are crucial for financial practitioners and algorithmic traders.

Lastly, the integration of advanced machine learning techniques, such as neural networks, or reinforcement learning, with traditional econometric models could further enhance the predictive accuracy and interpretability of the models. This hybrid approach could leverage the strengths of both data-driven methodologies and established econometric frameworks, offering powerful new tools for financial market analysis.

Chapter 6

Appendix

6.1 Appendix of Chapter 2

6.1.1 Proof of Proposition 2.1

Proof. Let us denote the estimators of ARMA parameters obtained from solving (2.10) as $\hat{\phi}_i^{\text{1-step}}$ for $i = 1, \ldots, p_1 + p_2$. Due to the orthogonality conditions and the fact that ARMA parameters are not penalized in (2.10), the following should hold:

$$\hat{\phi}_i^{(\text{1-step})} \xrightarrow{plim} \phi_i, \text{ for } i = 1, \ldots, p_1 + p_2.$$

Moreover, due to the orthogonality conditions the estimators of ARMA parameters obtained from *Step 1* of the three-step methodology ($\hat{\phi}_i^{\text{3-step}}$) are also consistent:

$$\hat{\phi}_i^{(\text{3-step})} \xrightarrow{plim} \phi_i, \text{ for } i = 1, \ldots, p_1 + p_2,$$

which implies that running penalized regression on the model in (2.8) results in asymptotically equivalent coefficient estimates as the ones obtained from the minimization of (2.10). This completes the proof. □

6.1.2 Abbreviations

1	ATRCT	Bitcoin Median Transaction Confirmation Time in Minutes
2	AVBLS	Bitcoin Average Block Size
3	BLCHS	Bitcoin api.blockchain Size
4	CPTRA	Bitcoin Cost Per Transaction
5	CPTRV	Bitcoin Cost % of Transaction Volume
6	DIFF	Bitcoin Difficulty
7	ETRAV	Bitcoin Estimated Transaction Volume
8	ETRVU	Bitcoin Estimated Transaction Volume USD
9	HRATE	Bitcoin Hash Rate
10	MIREV	Bitcoin Miners Revenue
11	MKPRU	Bitcoin Market Price USD
12	MKTCP	Bitcoin Market Capitalization
13	MWNTD	Bitcoin My Wallet Number of Transaction Per Day
14	MWNUS	Bitcoin My Wallet Number of Users
15	MWTRV	Bitcoin My Wallet Transaction Volume
16	NADDU	Bitcoin Number of Unique Bitcoin Addresses Used
17	NTRAN	Bitcoin Number of Transactions
18	NTRAT	Bitcoin Total Number of Transactions
19	NTRBL	Bitcoin Number of Transaction per Block
20	NTREP	Bitcoin Number of Transactions Excluding Popular Addresses
21	TOTBC	Total Bitcoins
22	TOUTV	Bitcoin Total Output Volume
23	TRFEE	Bitcoin Total Transaction Fees
24	TRFUS	Bitcoin Total Transaction Fees USD
25	TRVOU	Bitcoin USD Exchange Trade Volume

Table 6.1 – Abbreviations for Blockchain Technology Variables

Public Opinion (www.bitinfocharts.com)
1. Number of tweets (tweet)
2. Google trends (gtrends)
Risks and Uncertainties
3. Geopolitical risks (GPRD) (www.matteoiacoviello.com)
4. Daily China economic policy uncertainty (CNEPU) Index (www.economicpolicyuncertaintyinchina.weebly.com)
Financials (www.finance.yahoo.com)
5. US Federal Funds Effective Rate (FFER) (www.fred.stlouisfed.org)
6. Ripple-USD Exchange Rate (XRP-USD)
7. Bitcoin Futures price (BTC=F)
8. Chinese Yuan to dollar exchange rate (CNYUSD=X)
Macroeconomic development (www.finance.yahoo.com)
9. Crude Oil (CL=F)
10. Nasdaq (NDAQ)
11. S&P500 (SPY)
12. DOW30 (DJI)
13. The Chicago Board Options Exchange Volatility Index (VIX)
14. The Chicago Board Options Exchange Gold Volatility Index (GVZ)

Table 6.2 – Abbreviations for exogenous variables in Public Opinion, Risks and Uncertainties, Financials and Macroeconomic Development group

1	rv_d	Daily realized variance
2	rv_w	Weekly realized variance
3	rv_m	Monthly realized variance
4	d_	Prefix in variable names, denotes for the first difference (d_$X := X_t - X_{t-1}$)
5	dl_	Prefix in variable names, denotes for the first logarithmic difference (dl_$X := \ln X_t - \ln X_{t-1}$)

Table 6.3 – Abbreviations for Realized Volatility and Prefixes

6.1.3 Time Series CV For LASSO or ABESS

The following algorithm outlines the methodology we follow for data partitioning to perform the cross validation for LASSO and ABESS.

Algorithm 6.1 Time Series CV For Variable Selection With LASSO or ABESS

Input: $\mathcal{D}_t = \{(z_t, \boldsymbol{x}_{t-1}, \boldsymbol{\sigma}_t | t = 1, ..., T\}$ where $\boldsymbol{\sigma_t}$ is the sequence of the realized standard deviation, $\boldsymbol{x}_{t-1}$ are regressors including the asymmetries, z_t are residuals within (2.8), Π is the parameter grid

function BESTSUBSETSELECTOR($\mathcal{D}_t$, Π)

Data Split: Training Sets (Ω_k) and Validation Sets (Υ_k), $\forall k = 1, 2, ..., \lfloor 0.2N \rfloor$, where cardinalities $\#(\Omega_1) = \lfloor 0.6N \rfloor = \tau_1$ and $\#(\Upsilon_1) = \lceil 0.2N \rceil = \tau_2$. Then, the collection of train and validation sets are:

$$
\begin{aligned}
\Omega_1 &:= \{\mathcal{D}_t | t = 1, 2, ..., \tau_1\} \\
\Upsilon_1 &:= \{\mathcal{D}_t | t = \tau_1 + 1\} \\
\Omega_2 &:= \{\mathcal{D}_t | t = 1, 2, ..., \tau_1 + 1\} \\
\Upsilon_2 &:= \{D_t | t = \tau_1 + 2\} \\
&\vdots \\
\Omega_{\tau_2} &:= \{\mathcal{D}_t | t = 1, 2, ..., \tau_1 + \tau_2 - 1\} \\
\Upsilon_{\tau_2} &:= \{\mathcal{D}_t | t = \tau_1 + \tau_2\}
\end{aligned}
$$

Define the set of parameters to be tuned with a search grid Π.
for each parameter set $\pi \in \Pi$ **do**
  for each $(\Omega_k \subseteq \Omega, \Upsilon_k \subset \Upsilon), \quad \forall k = 1, ..., \tau_2$ **do**
    Fit LASSO or ABESS
    Generate Predictions for Υ_k using LASSO or ABESS, respectively
  end for
  Calculate average performance (RMSE) across Υ and $\forall \pi \in \Pi$.

$$RMSE_\pi = \sqrt{\frac{1}{\tau_2} \sum_{t=\tau_1}^{\tau_1+\tau_2} (\boldsymbol{\sigma}_t - \hat{\boldsymbol{\sigma}}_t)^2}$$

end for
Determine the optimal set of tuning parameters based on the lowest RMSE
Fit the ABESS or LASSO with respective optimal tuning parameters to $\Omega_{\tau_2} \cup \Upsilon_{\tau_2}$
Get The Best Subset of Covariates, $\mathbf{x}^*$
end function

Output: Selected subset of covariates ($\mathbf{x}^*$) either by LASSO or ABESS.

6.1.4 Time Series CV For Random Forest (Boruta)

The following algorithm describes the technique of data partitioning to perform the time series cross validation for the random forest (Boruta).

Algorithm 6.2 Time Series CV For Random Forest Behind The Boruta

Input: $\mathcal{D}_t = \{(z_t, \boldsymbol{x}_{t-1}, \boldsymbol{\sigma}_t | t = 1, ..., T\}$ where $\boldsymbol{\sigma_t}$ is the sequence of the realized standard deviation, $\boldsymbol{x}_{t-1}$ are regressors including the asymmetries, z_t are residuals within (2.8), number of trees (n_{tree}), minimum node size (n_{min}), block size (l_n), number of split variables (m), $\Pi := (n_{tree}, n_{min}, l_n, m)'$ is four dimensional the parameter grid

Data Split: Training Set $\mathcal{T} = \{\mathcal{D}_t | 1 < t < \lfloor 0.8T \rfloor\}$.
Define the set of parameters to be tuned with a search grid Π.
for $\pi \in \Pi$ **do**
  for each b = 1 to n_{tree} **do**
    Draw block bootstrap sample $Z_\star$ of size $\#(Z_\star) \leq \lfloor 0.8T \rfloor$ with parameter l_n.
    while node size $n > n_{min}$ **do**
      Chose a random set of m variables among the p variables and apply the CART criterion on this subset.
      Cut on the best split.
    end while
  end for
  Make out-of-bag (OOB) predictions $\hat{\boldsymbol{\sigma}}_t$
  Calculate average OOB performance (RMSE) across $\forall \pi \in \Pi$.
end for
Determine the optimal set of tuning parameters based on the lowest RMSE
Apply Boruta with the optimal parameters
Get The Best Subset of Covariates, $\mathbf{x}^*$

Output: Selected subset of covariates ($\mathbf{x}^*$) by Boruta.

6.1.5 Evaluation of the Predictive Performance of VS-LTGARCHX variants and The Benchmark Models

The following algorithm summarizes the strategy we follow to evaluate the predictive performance of the different variants of VS-LTGARCHX algorithm.

Algorithm 6.3 Evaluation of the Predictive Performance of VS-LTGARCHX variants and The Benchmark Models

Inputs: $\mathcal{D}_t = \{(r_t, \boldsymbol{x}^*_{t-1}, \boldsymbol{\sigma}_t)|t = 1, ..., T\}$, r_t are log-returns, x^*_{t-1} is deemed to be the selected variables from Algorithm 6.1 or 6.2 for VS-LTGARCHX variants, the full set of the variables at hand for the benchmark log-TGARCHX model, and an empty set for the benchmark log-GARCH(1,1).

1: **Data Split**: Define set of indices $\mathcal{S} = \{\tau : \lfloor 0.8T \rfloor \leq \tau \leq T\}$ for the test data, with cardinality $\#(\mathcal{S}) = \lceil 0.2T \rceil$
2: **for** $\tau \in \mathcal{S}$ **do**
3: Fit the intended model using $\mathcal{D}_\tau = \{\mathcal{D}_t|\tau - \lfloor 0.8T \rfloor + 1 \leq t \leq \tau\}$
4: Predict $\sigma_{\tau+1}$:

$$\mathbb{E}(\sigma_{\tau+1}|\mathcal{D}_\tau) = f(r_\tau, \boldsymbol{x}^*_{\tau-1})$$

5: Calculate the prediction errors, $\bar{d}_\tau = \sigma_\tau - \hat{\sigma_\tau}$
6: **end for**
7: Evaluate predictive performance using the following:

$$\begin{aligned} ME &= \frac{1}{\#(\mathcal{S})} \sum_{\tau \in \mathcal{S}} \bar{d}_\tau, \quad RMSE = \sqrt{\frac{1}{\#(\mathcal{S})} \sum_{\tau \in \mathcal{S}} \bar{d}^2_\tau} \\ MAE &= \frac{1}{\#(\mathcal{S})} \sum_{\tau \in \mathcal{S}} |\bar{d}_\tau|, \quad MAPE = \frac{1}{\#(\mathcal{S})} \sum_{\tau \in \mathcal{S}} \left|\frac{\bar{d}_\tau}{\sigma_\tau}\right| \\ MPE &= \frac{1}{\#(\mathcal{S})} \sum_{\tau \in \mathcal{S}} \frac{\bar{d}_\tau}{\sigma_\tau}, \quad QLIKE = \frac{1}{\#(\mathcal{S})} \sum_{\tau \in \mathcal{S}} (\ln \hat{\sigma_\tau}^2 + \frac{\sigma^2_\tau}{\hat{\sigma_\tau}^2}) \end{aligned} \tag{6.1}$$

Output: The table of predictive performance scores of the models considered.

6.1.6 Study of Robustness

The below table (Table 6.4) is the extended version of Table 2.2. In Table 6.4, together with RMSE and QLIKE metrics, we report mean error (ME), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Mean Percentage Error (MPE) for rolling window forecasts (see (6.1) and Hyndman and Athanasopoulos (2018) for details).

	ME	RMSE	MAE	MPE	MAPE	QLIKE
Log-GARCH	1.20	2.00	1.70	24.78	36.04	3.74
Log-TGARCHX	**0.42**	1.85	1.39	**3.02**	36.66	3.82
LASSO	0.65	1.60	**1.23**	15.47	29.66	**3.64**
ABESS	0.67	**1.59**	1.24	16.11	**29.53**	**3.64**
BORUTA	0.47	1.66	1.25	9.94	31.61	3.69

Table 6.4 – Predictive accuracy of models in the test set.

Although we are aware of the potential problems posed by the recursive forecasting scheme when evaluating nested models, we follow Hansen et al. (2011) and test our algorithm using the recursive scheme, also for the sake of robustness. Interestingly, our conclusions about the superiority of forecasts generated by the VS-LTGARCHX models over those generated by the benchmark models remain valid. Table 6.5 describes the performance metrics of the models in the test set (20% of the overall data). The training window is expanding by one observation at each forecasting origin. The initial training set comprises 1314 (80% of the data) observations. The results of the MCS procedure are exactly the same as those in the rolling window forecasting scheme, and thus, are omitted.

	ME	RMSE	MAE	MPE	MAPE	QLIKE
Log-GARCH	1.22	2.03	1.74	24.45	36.60	3.75
Log-TGARCHX	**0.07**	1.78	1.31	**7.91**	40.34	3.93
LASSO	0.59	1.62	1.24	14.07	30.29	**3.65**
ABESS	0.61	**1.61**	1.24	14.66	**30.02**	**3.65**
BORUTA	0.45	1.64	**1.22**	9.65	31.15	3.68

Table 6.5 – Recursive window forecasts.

6.1.7 Volatility Clustering

Volatility clustering, or volatility persistence, is a phenomenon well known in finance literature (Cont, 2001). This feature of financial markets implies that high volatility periods follow high volatility periods and low volatility periods follow low volatility periods. In Figure 6.1, the autocorrelation function (ACF) of daily realized volatility and the ACF of volatility predictions by respective models are depicted. The figure supports the volatility clustering phenomenon in the BTC market. Interestingly, ACFs of volatility predictions from all models are overestimating the true underlying ACF. Nevertheless, the volatility predictions from LASSO and ABESS suggest a faster decaying ACF structure, which is more consistent with the ACF of the daily realized volatility.

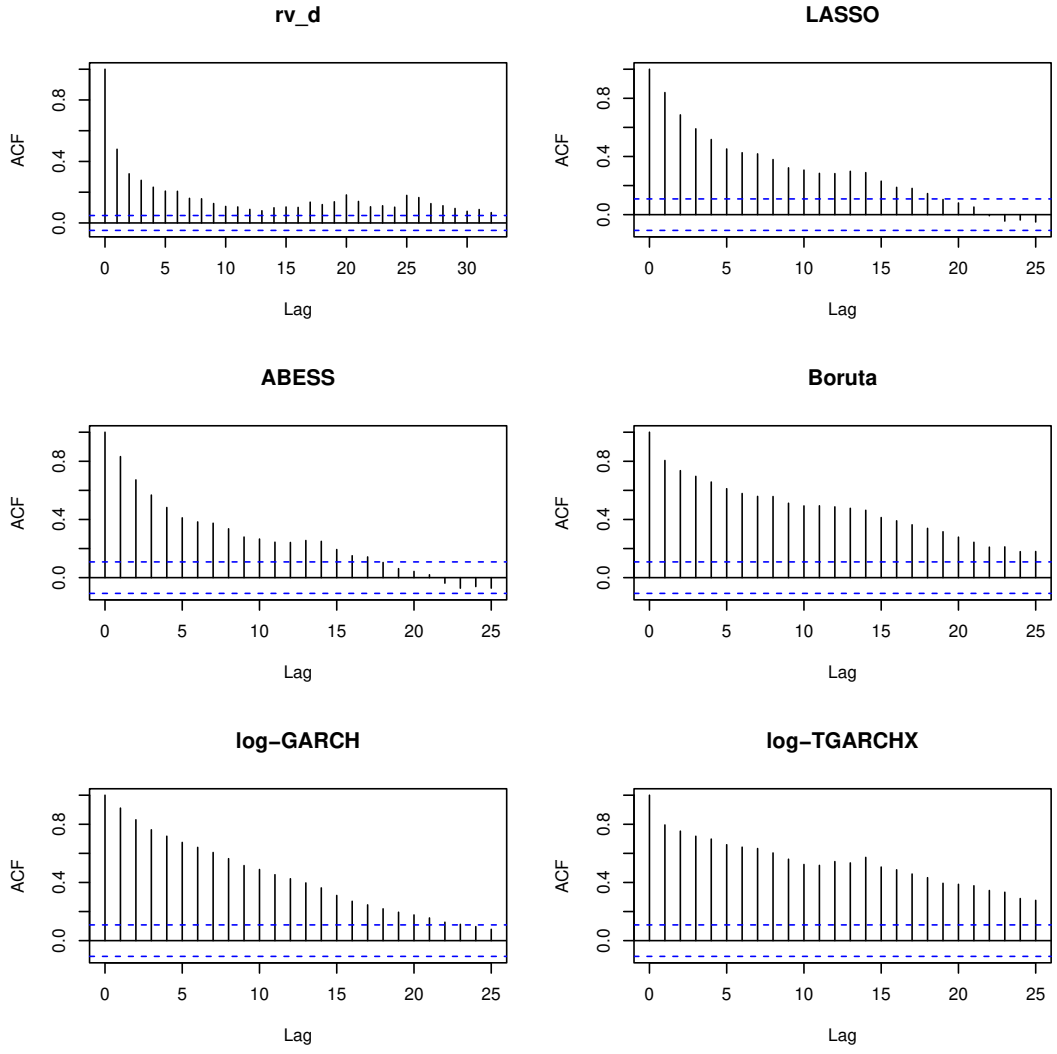


Figure 6.1 – Autocorrelation function (ACF) of daily realized volatility and ACF of volatility predictions by respective models.

6.2 Appendix of Chapter 3

6.2.1 Stochastic Version of the CE Method for Minimization

Algorithm 6.4 General CE for Minimization (Stochastic Version)

Input: N number of samples, $f(\cdot; \mathbf{u}_0)$ - unimodal reference distribution with an arbitrary parameter vector $\mathbf{u}_0$ and with a support containing the optimum.

1: Initialize $\hat{\boldsymbol{v}}_0 := \hat{\boldsymbol{u}}_0$, ρ and $t := 1$ (iteration counter).
2: **while** Stopping Criterion **do**
3: **Adaptive updating of** $\hat{\gamma}^{(t)}$:
Draw a sample $\boldsymbol{X}_1, ..., \boldsymbol{X}_N$ from $f(\cdot; \hat{\boldsymbol{v}}_{t-1})$;
Evaluat the corresponding performances $\ell_i = L(\boldsymbol{X}_i) : i = 1, 2, ..., N$ in an ascending order, and denote the ordered sequence as $\ell_{(1)} \leq ... \leq \ell_{(N)}$;
Update $\hat{\gamma^{(t)}} := \ell_{(\lceil \rho N \rceil)}$, where $\hat{\gamma^{(t)}}$ (sample quantile) is an estimator of $\gamma^{(t)}$, which is $\rho 100\%$ population quantile.
4: **Adaptive updating of** $\hat{\boldsymbol{v}}_t$: Solve the stochastic counterpart for a fixed $\hat{\gamma}_t$ and $\hat{\boldsymbol{v}}_{t-1}$:

$$\mathbf{v}_t = \underset{\mathbf{v}}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{I}_{\{L(\mathbf{X_i}) \leq \hat{\gamma}_t\}} \ln \left(f(\mathbf{X}_i; \mathbf{v}) \right)$$

5: $t := t + 1$
6: **end while**

6.2.2 Boxplot of Losses, Number of Iterations and Computation Time For Different Variants of the CESBKM Algorithm

Threshold Model With SBC (Symmetry Breaking Constraint) On The Fitted Values

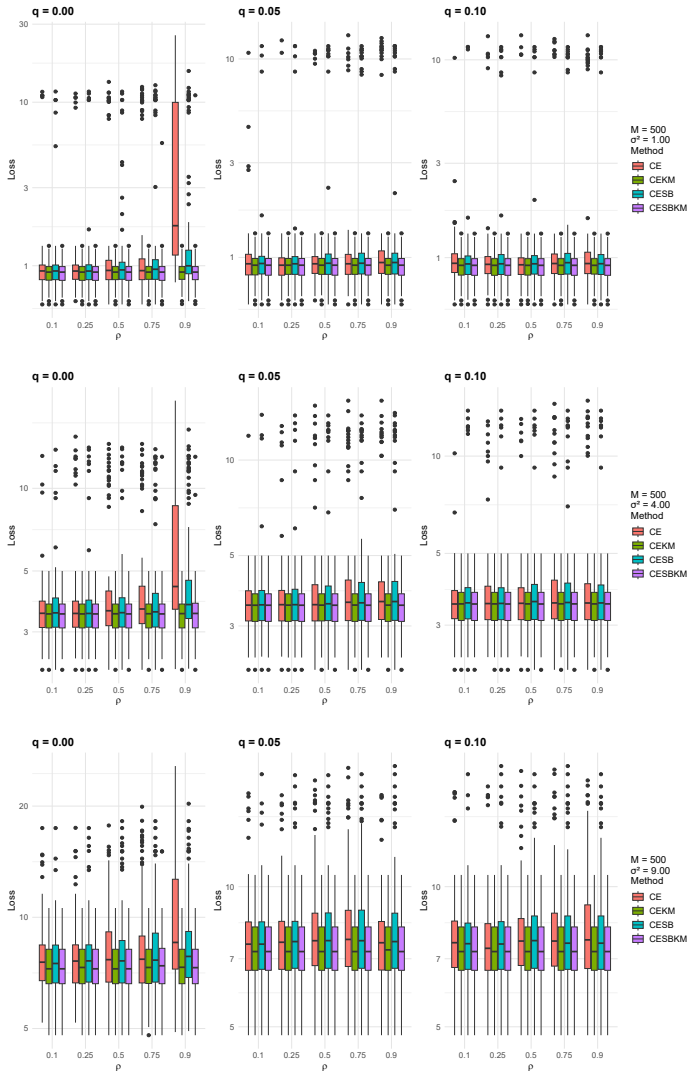


Figure 6.2 – Loss values (y-axis in log-scale) for Threshold model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

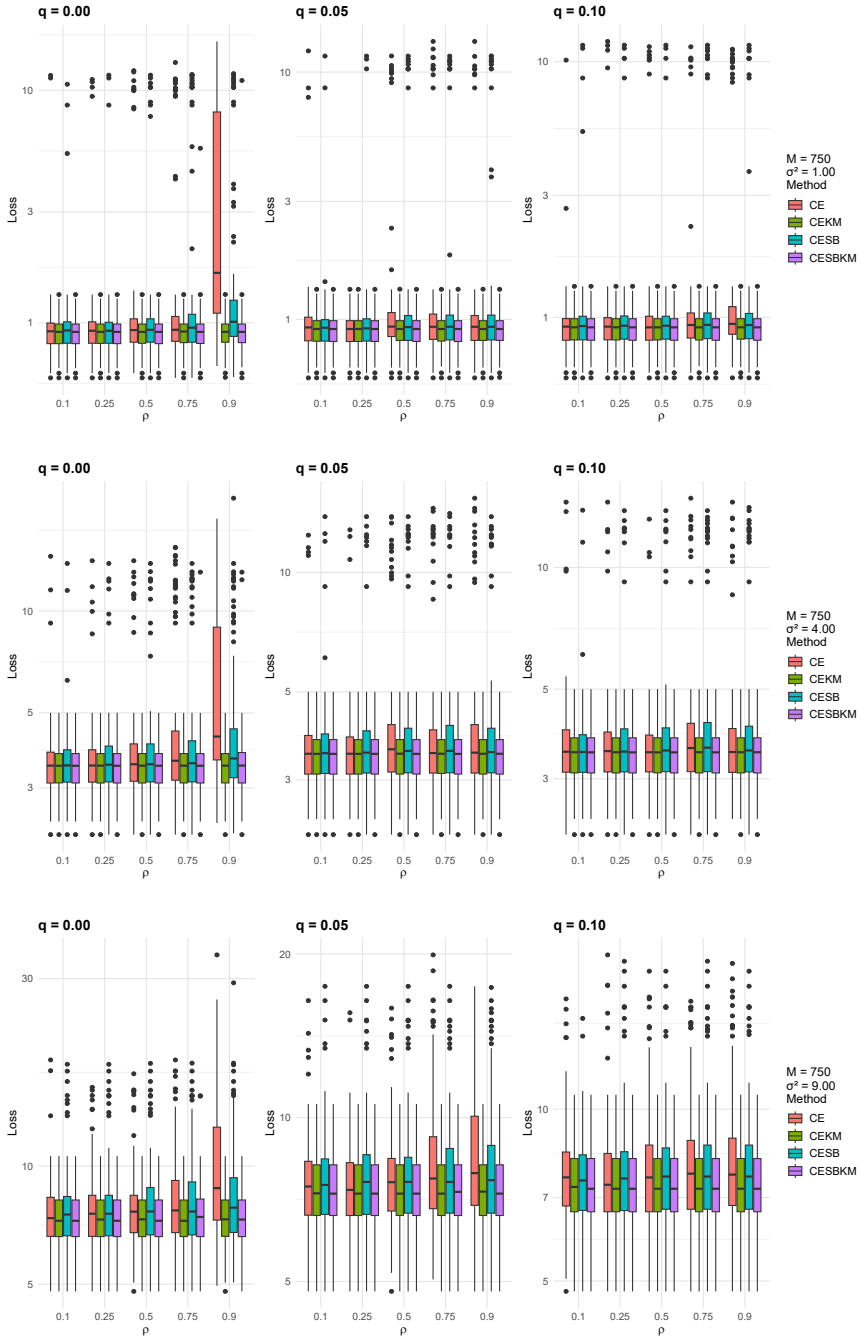


Figure 6.3 – Loss values (y-axis in log-scale) for Threshold model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

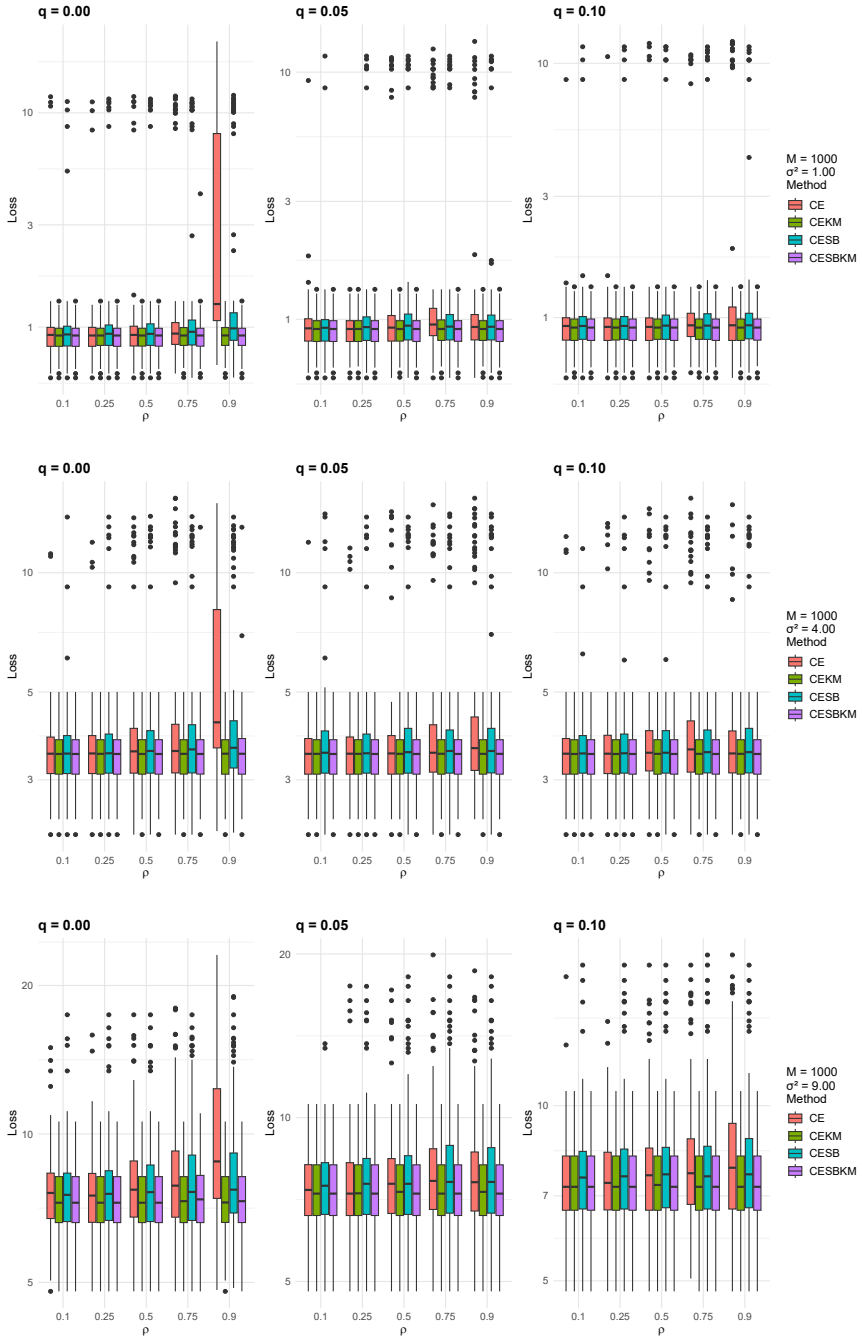


Figure 6.4 – Loss values (y-axis in log-scale) for Threshold model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

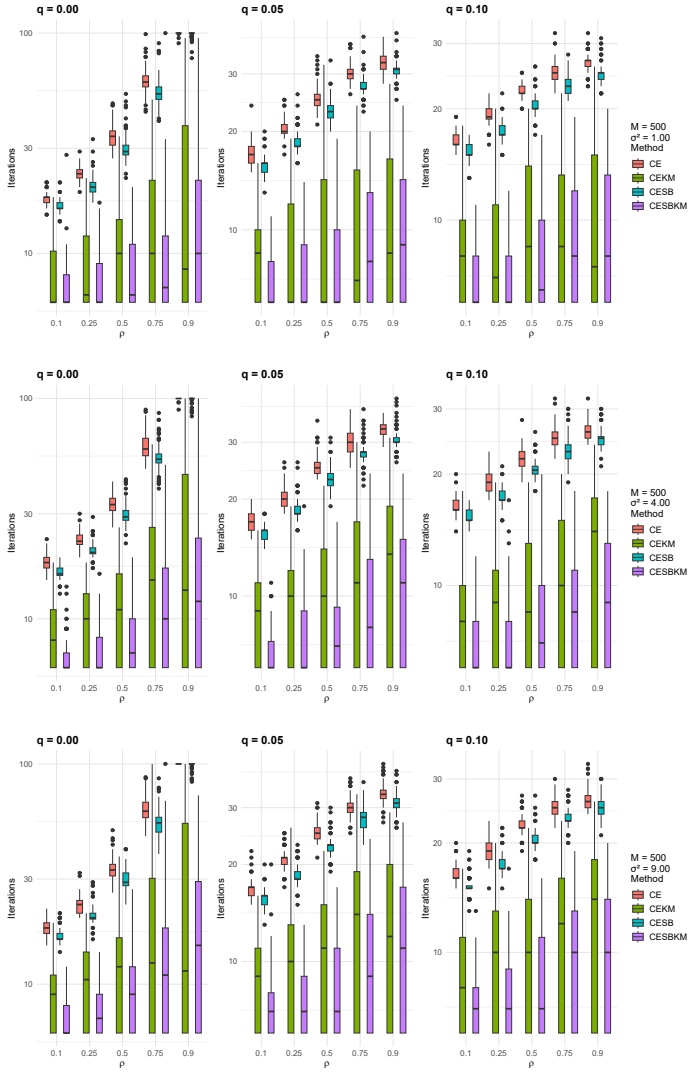


Figure 6.5 – Number of Iterations (y-axis in log-scale) for Threshold model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

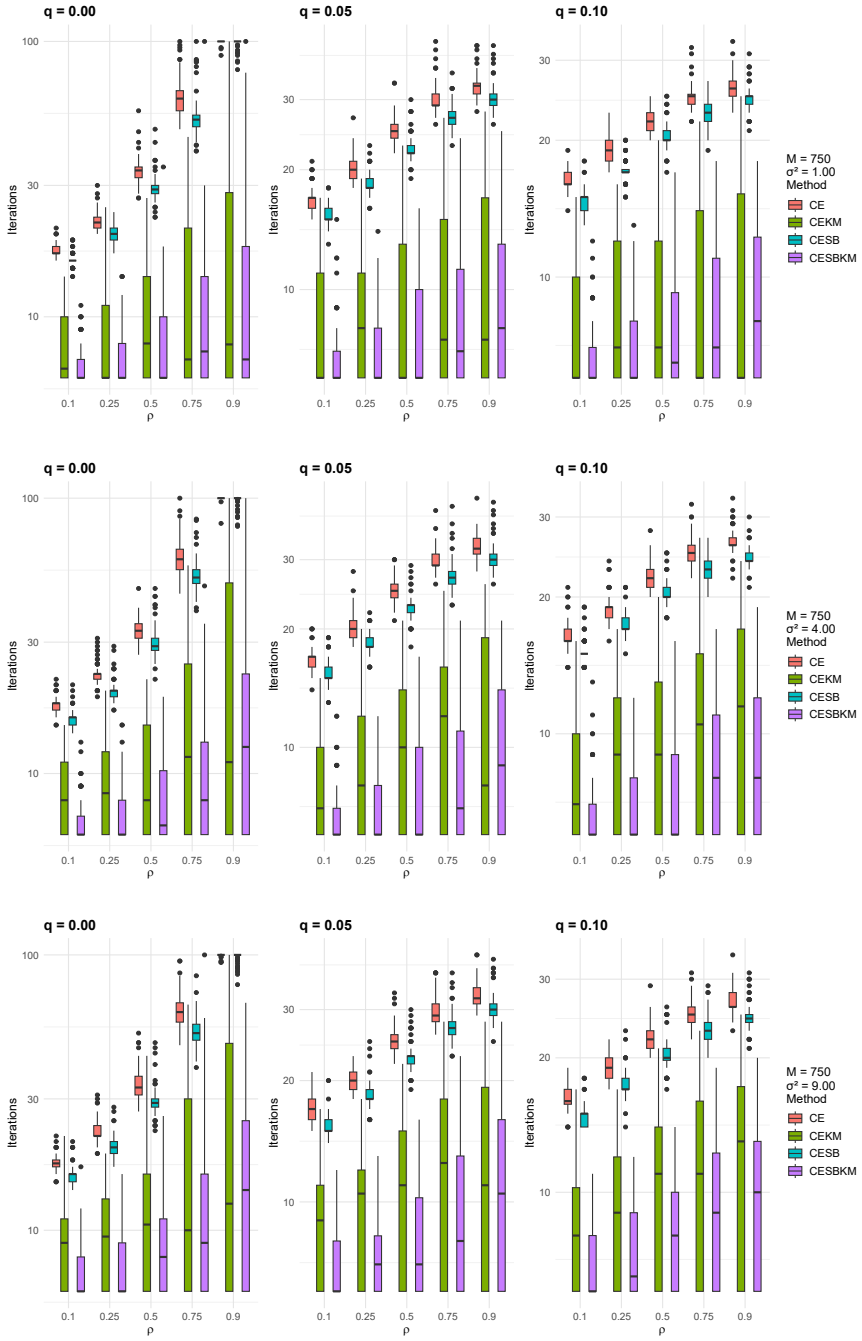


Figure 6.6 – Number of Iterations (y-axis in log-scale) for Threshold model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

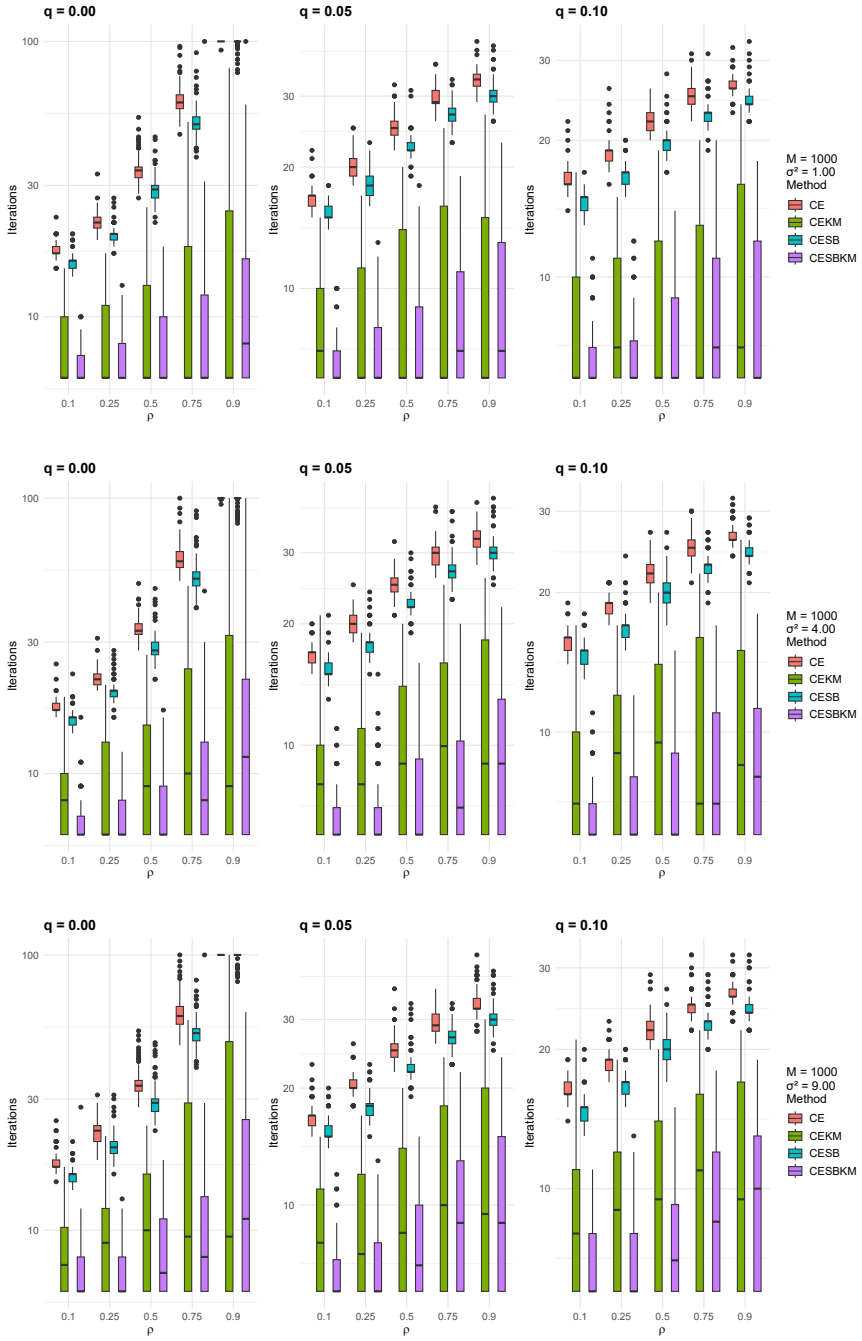


Figure 6.7 – Number of Iterations (y-axis in log-scale) for Threshold model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

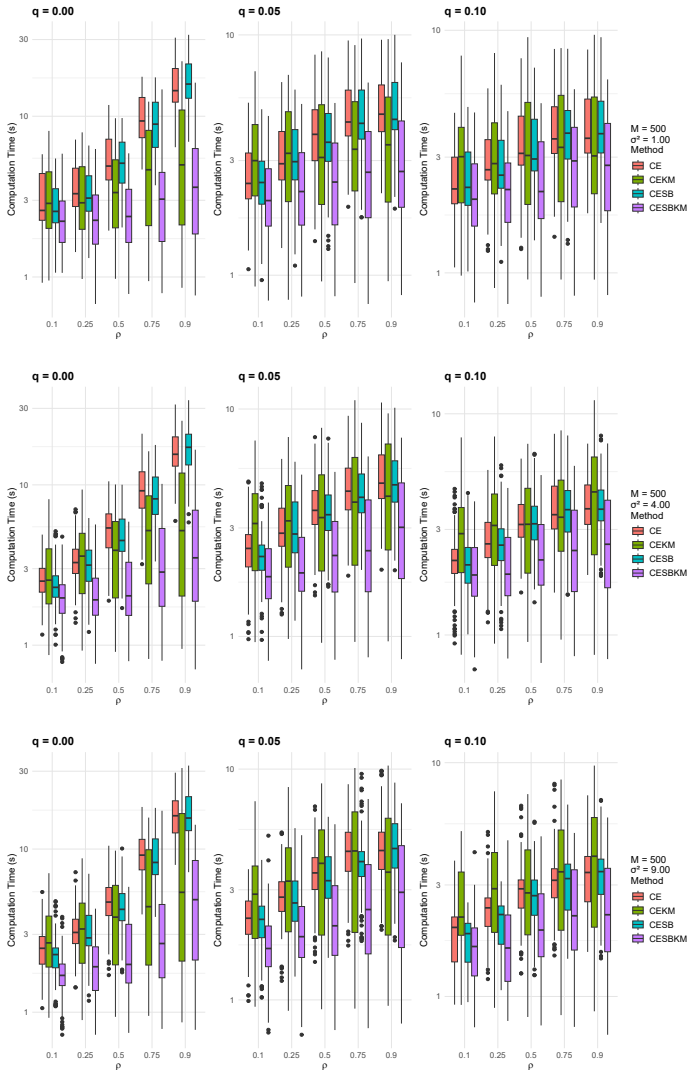


Figure 6.8 – Computation Time (y-axis in log-scale) for Threshold model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

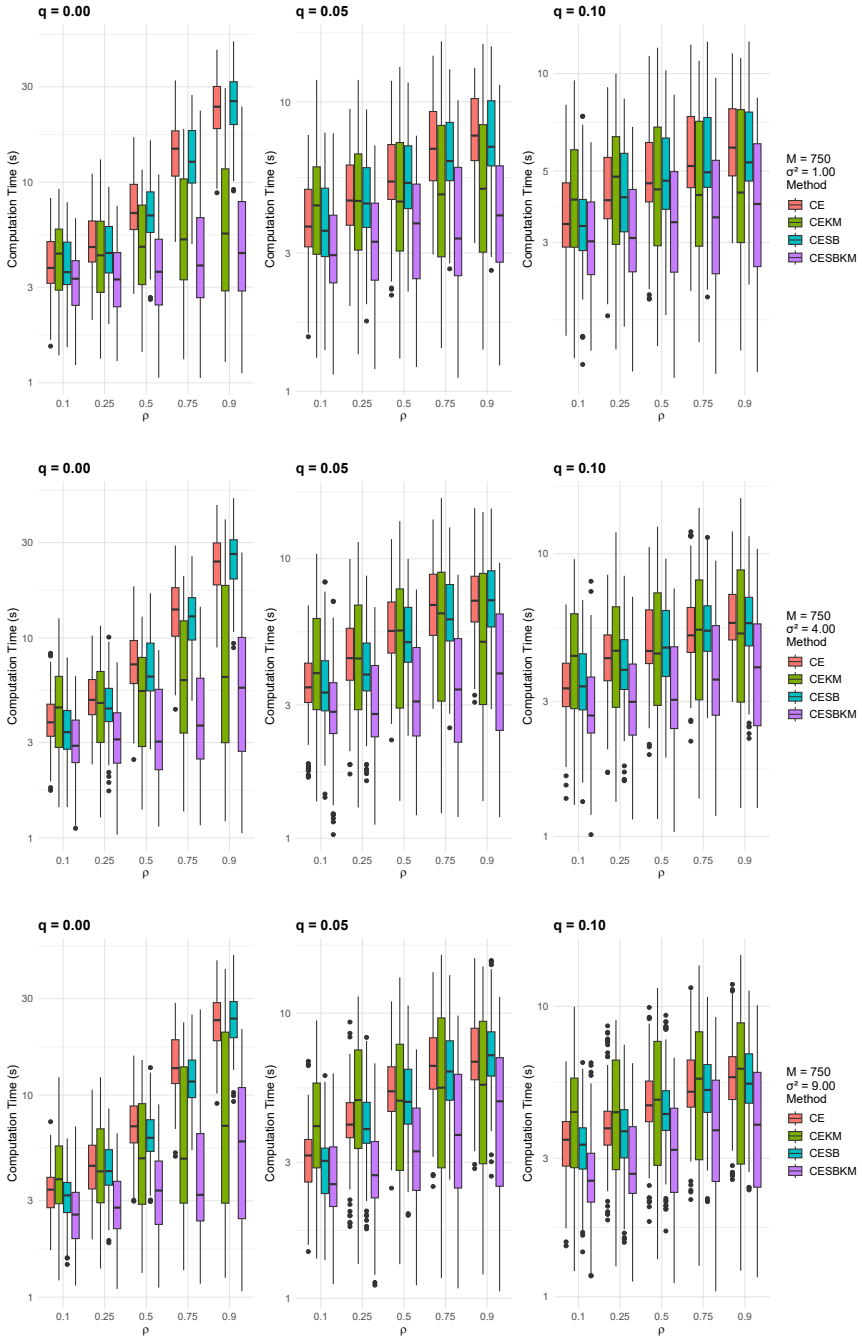


Figure 6.9 – Computation Time (y-axis in log-scale) for Threshold model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

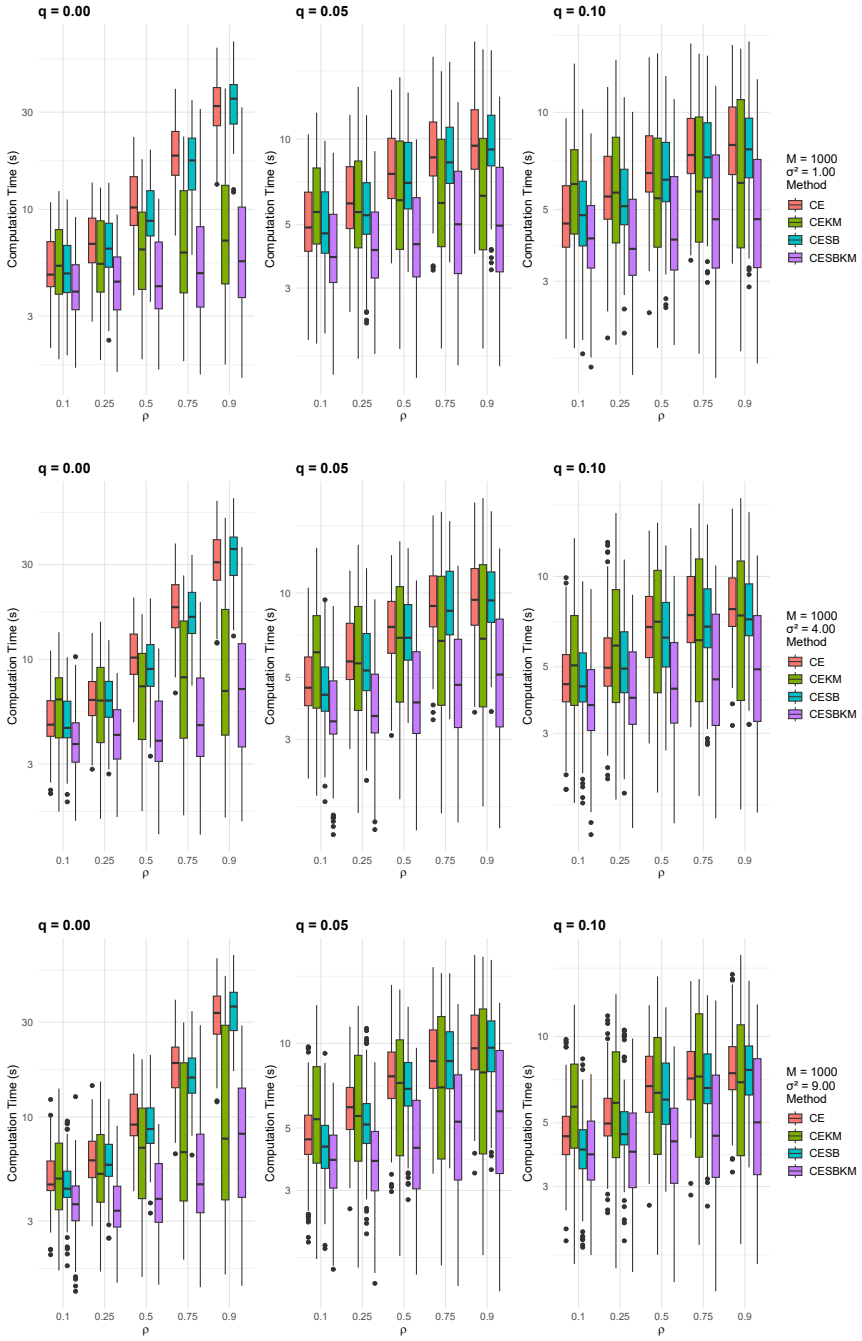


Figure 6.10 – Computation Time (y-axis in log-scale) for Threshold model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

Threshold Model With SBC On Centroids

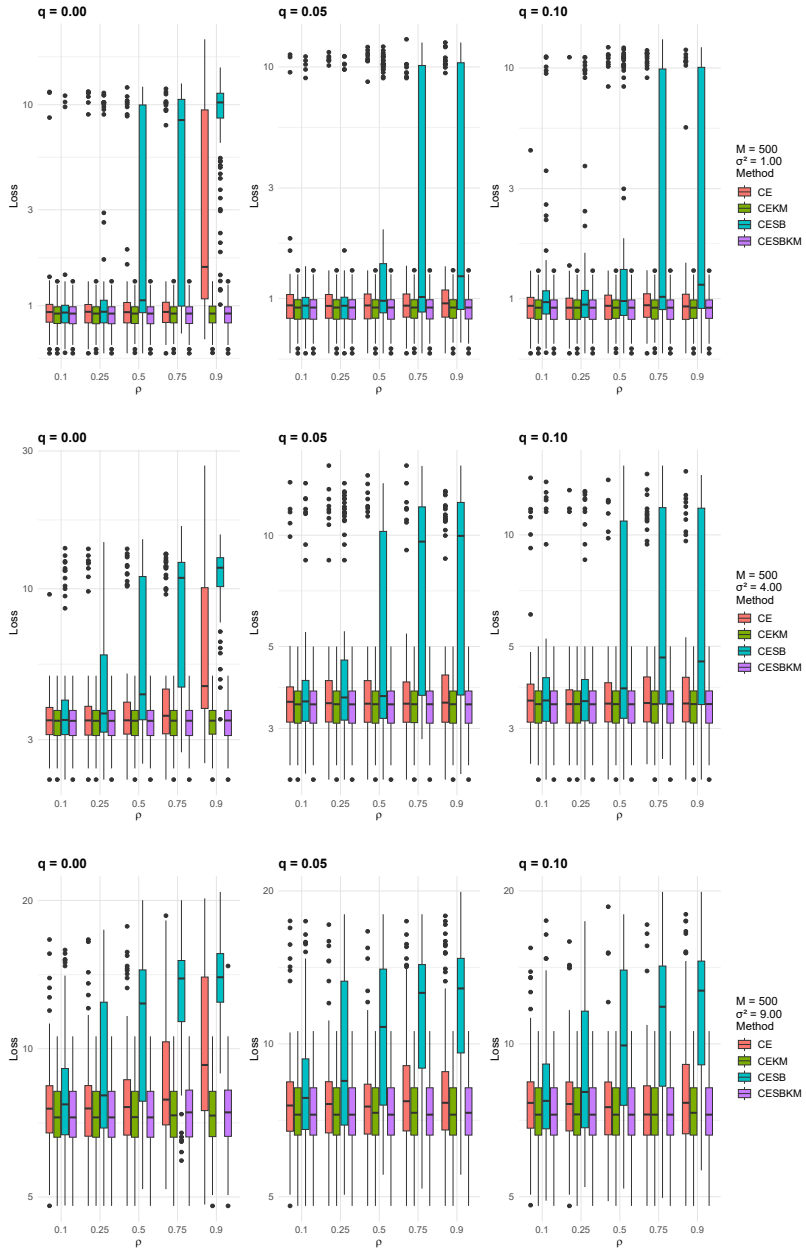


Figure 6.11 – Loss values (y-axis in log-scale) for Threshold model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

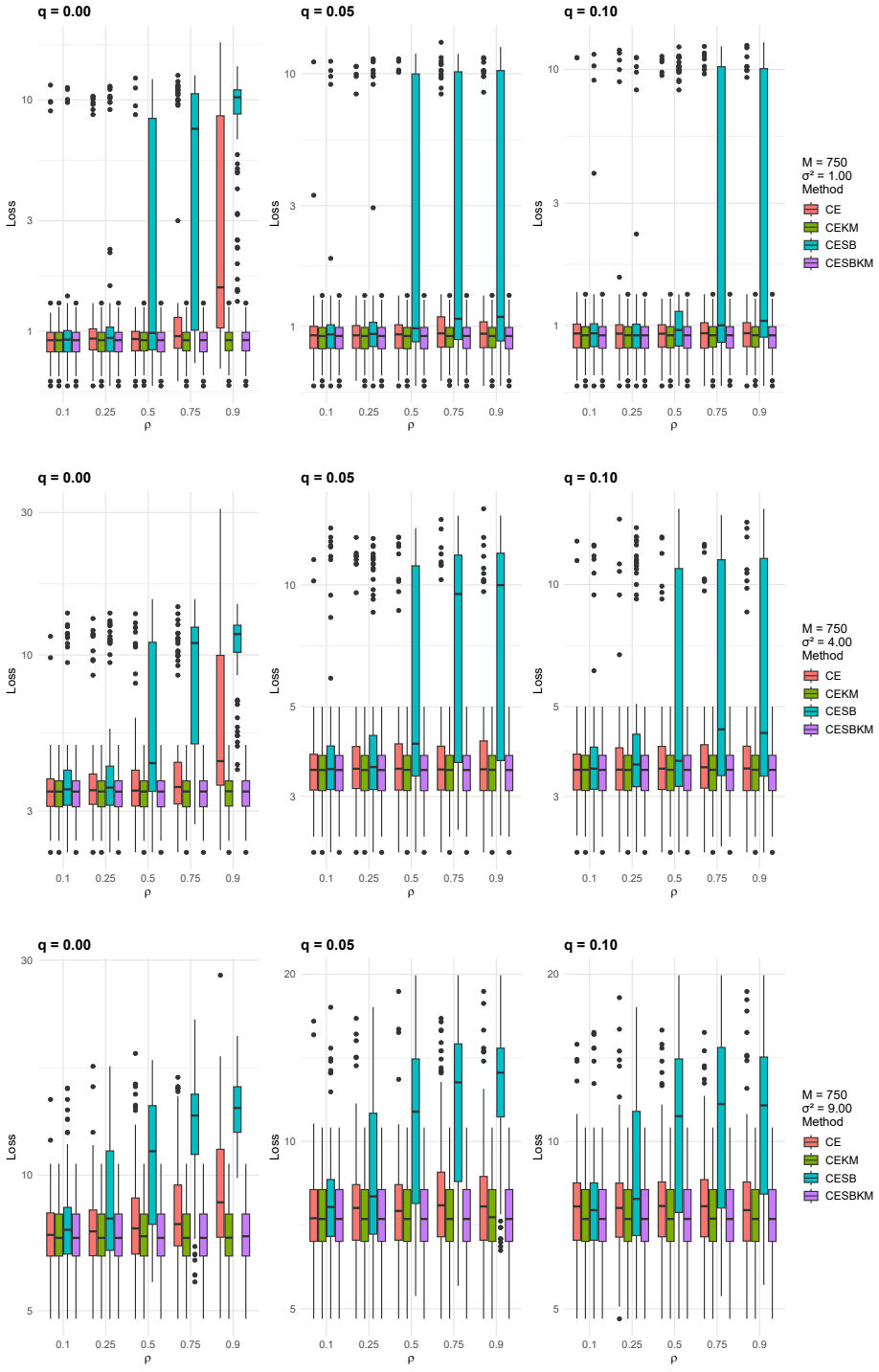


Figure 6.12 – Loss values (y-axis in log-scale) for Threshold model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

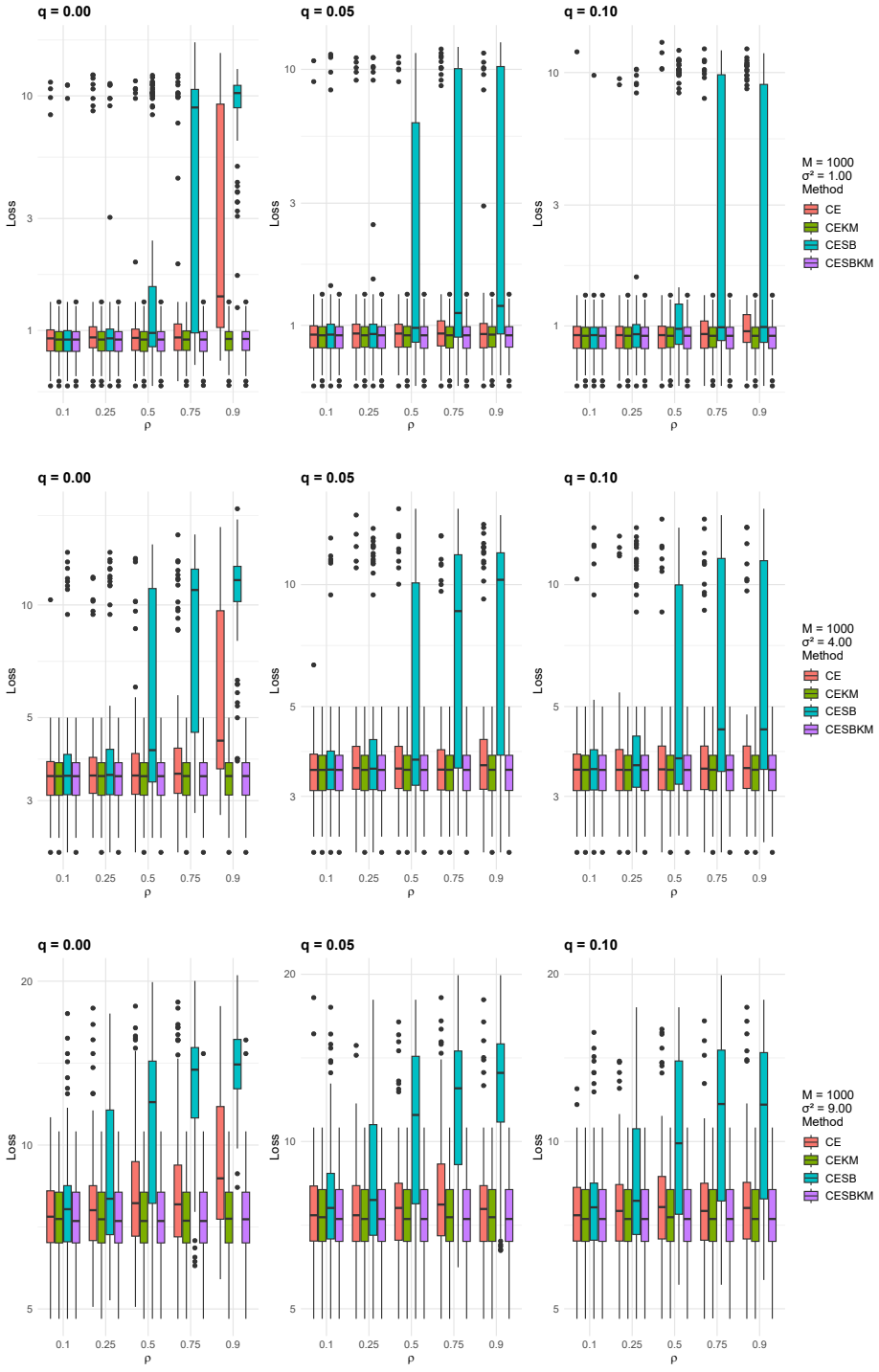


Figure 6.13 – Loss values (y-axis in log-scale) for Threshold model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

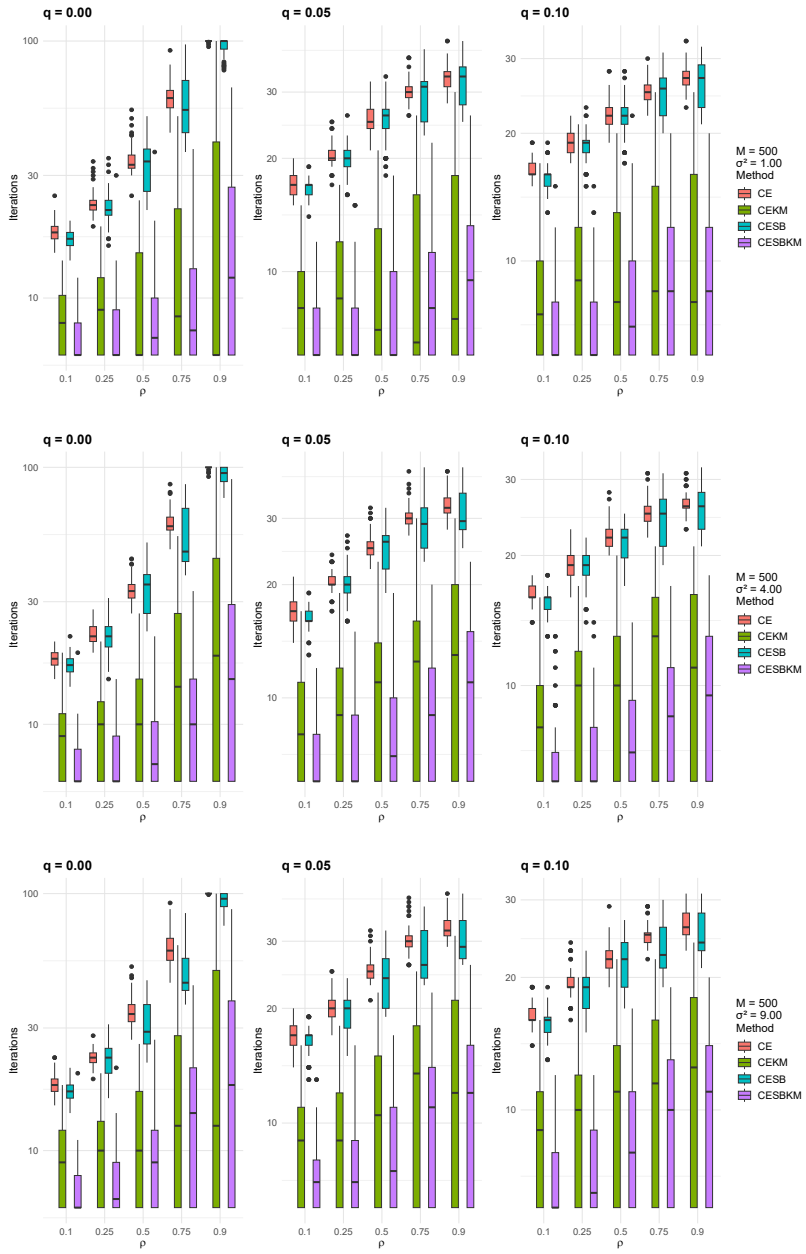


Figure 6.14 – Number of Iterations (y-axis in log-scale) for Threshold model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

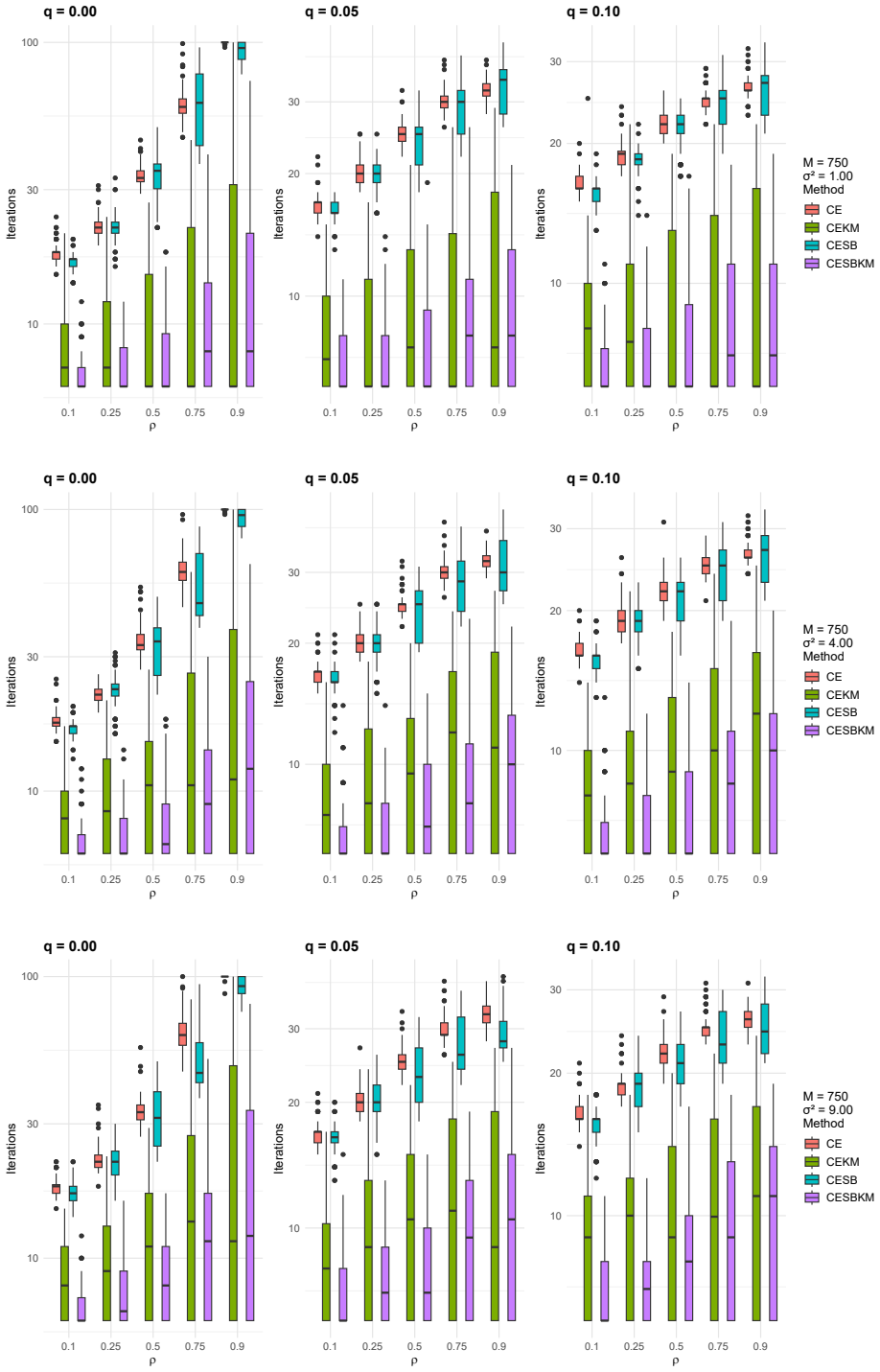


Figure 6.15 – Number of Iterations (y-axis in log-scale) for Threshold model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

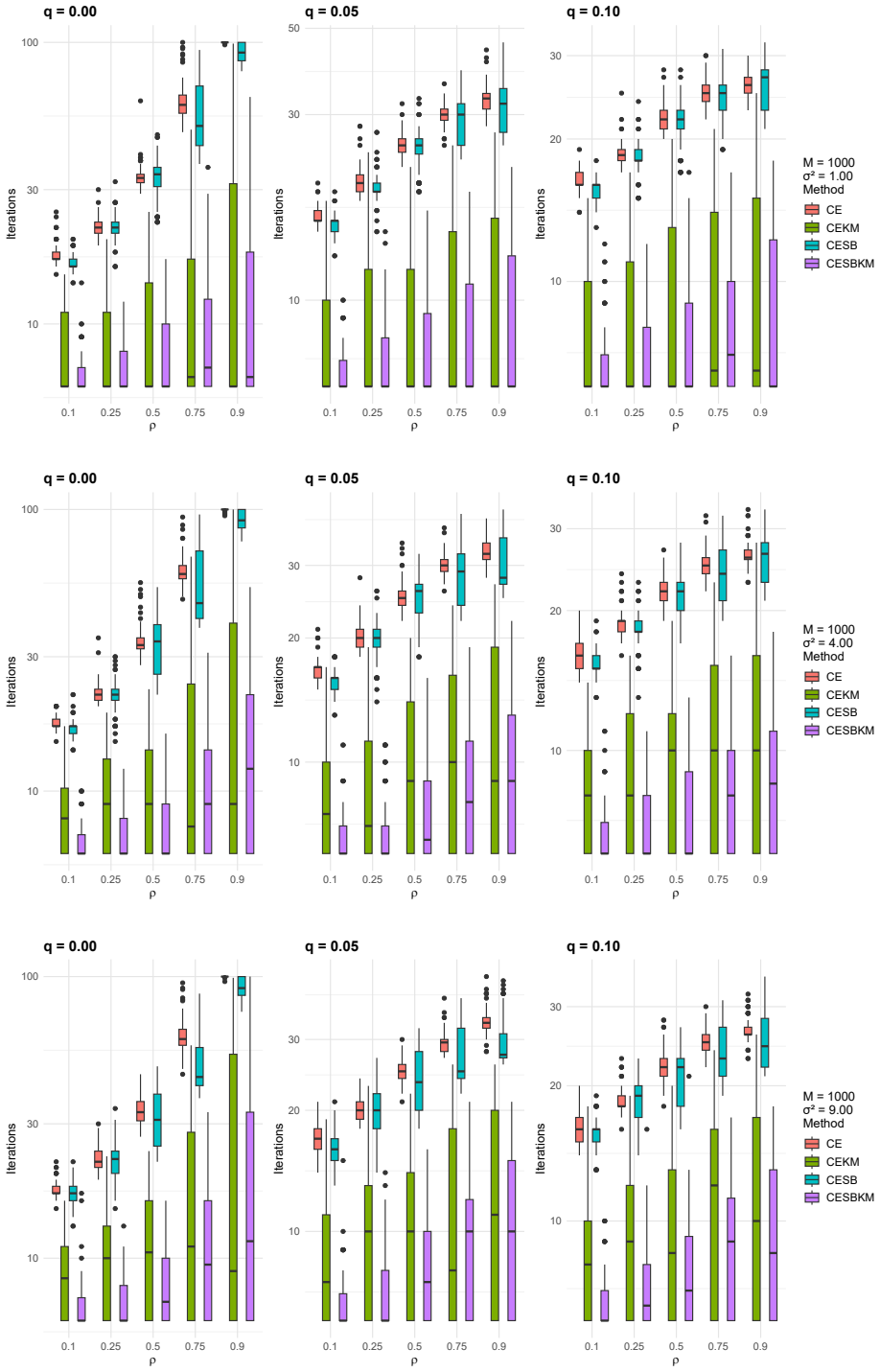


Figure 6.16 – Number of Iterations (y-axis in log-scale) for Threshold model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

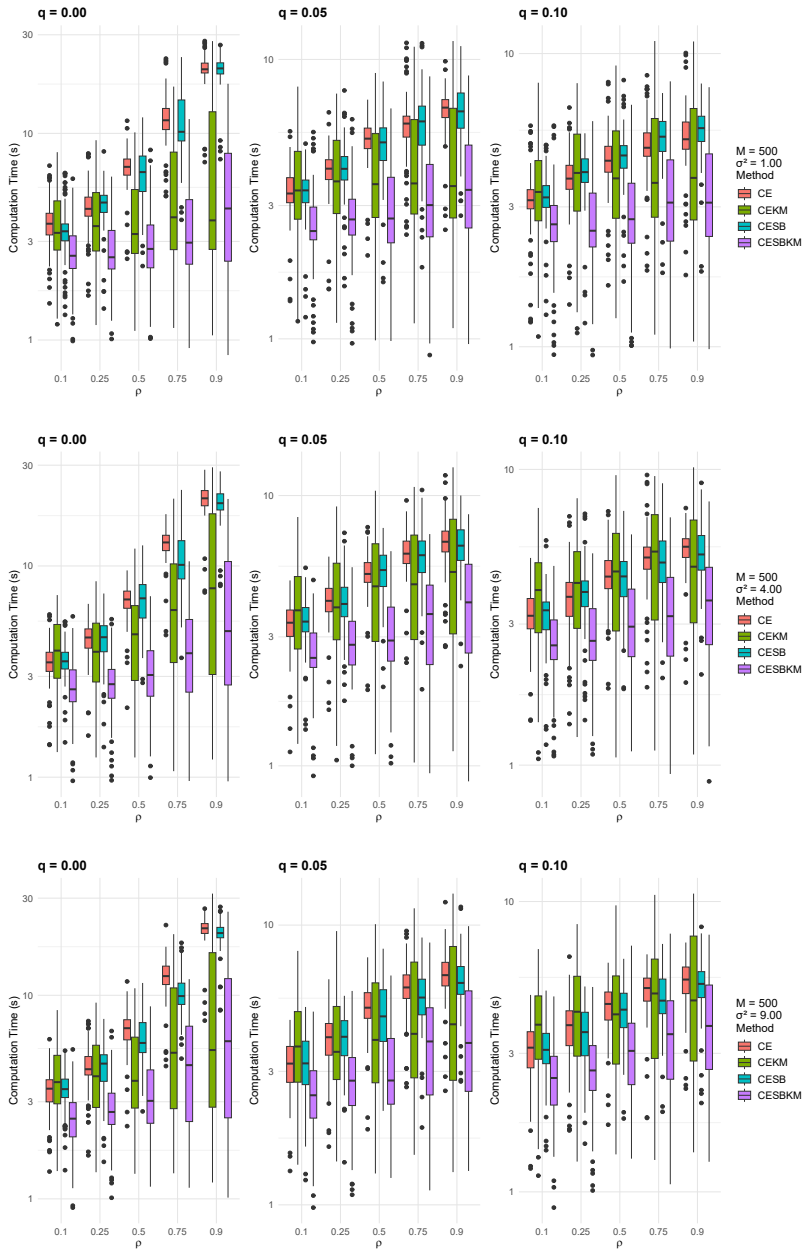


Figure 6.17 – Computation Time (y-axis in log-scale) for Threshold model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

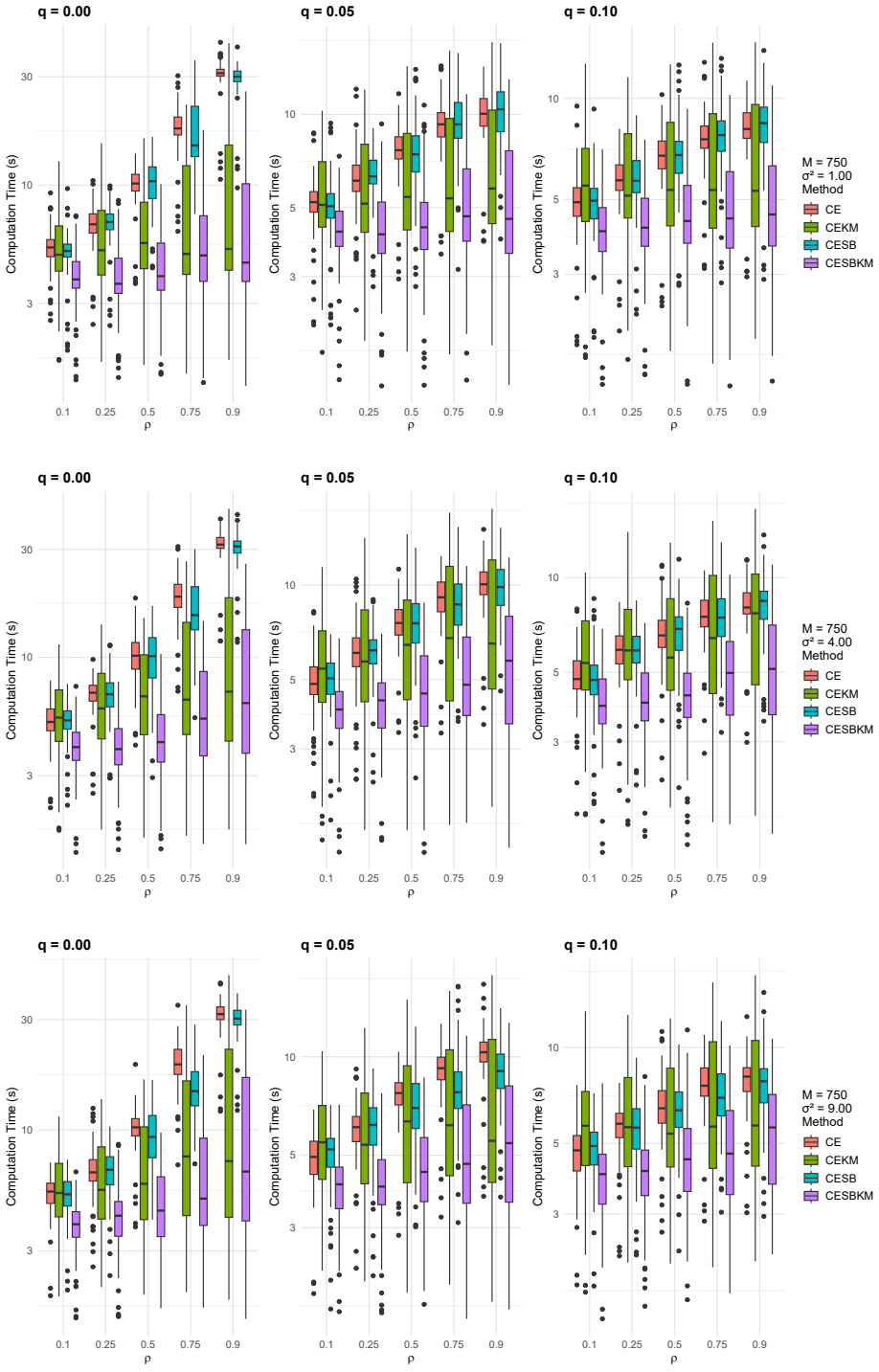


Figure 6.18 – Computation Time (y-axis in log-scale) for Threshold model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

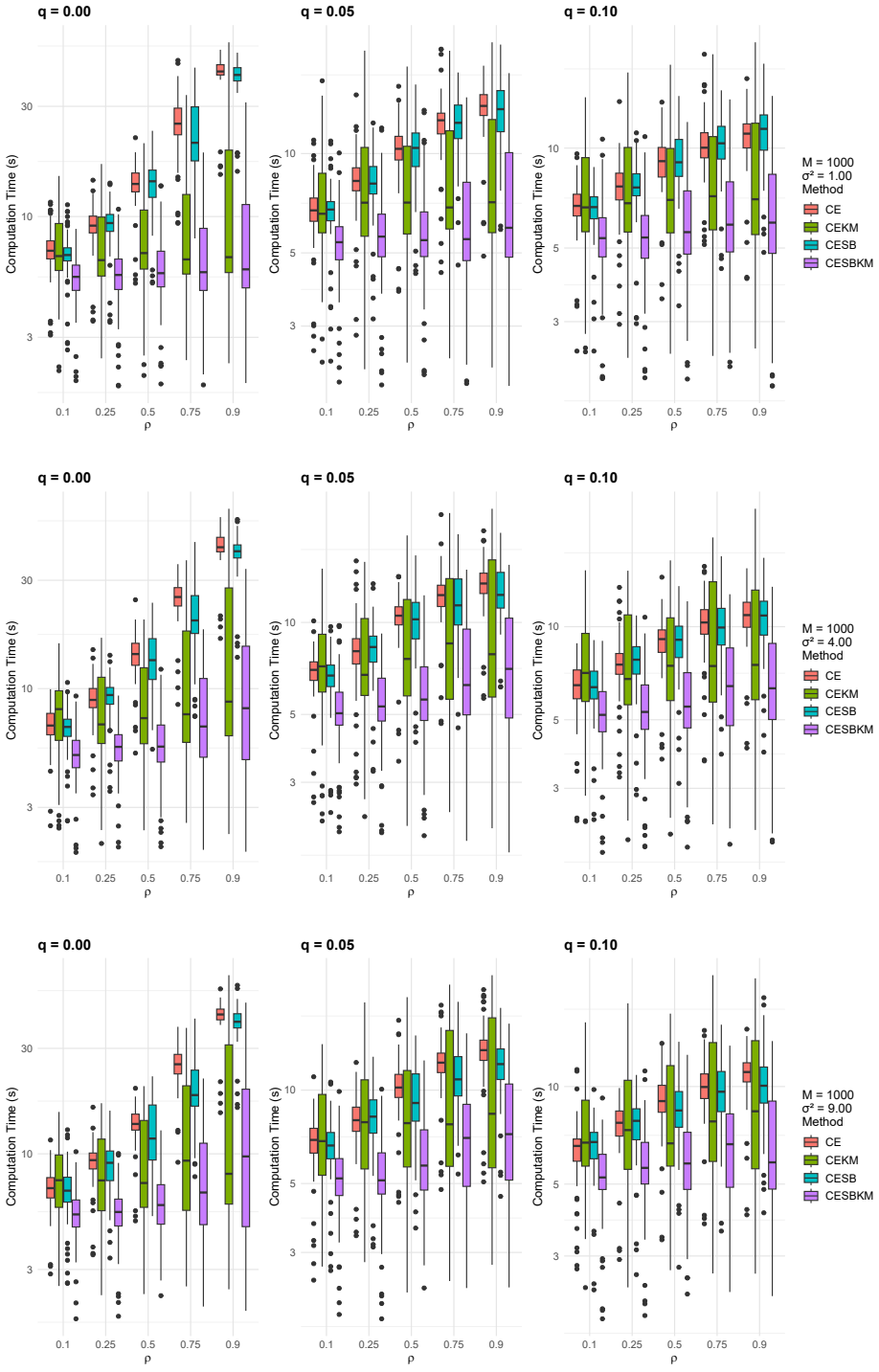


Figure 6.19 – Computation Time (y-axis in log-scale) for Threshold model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

Threshold Model With SBC On The First Element of The Cluster Vector

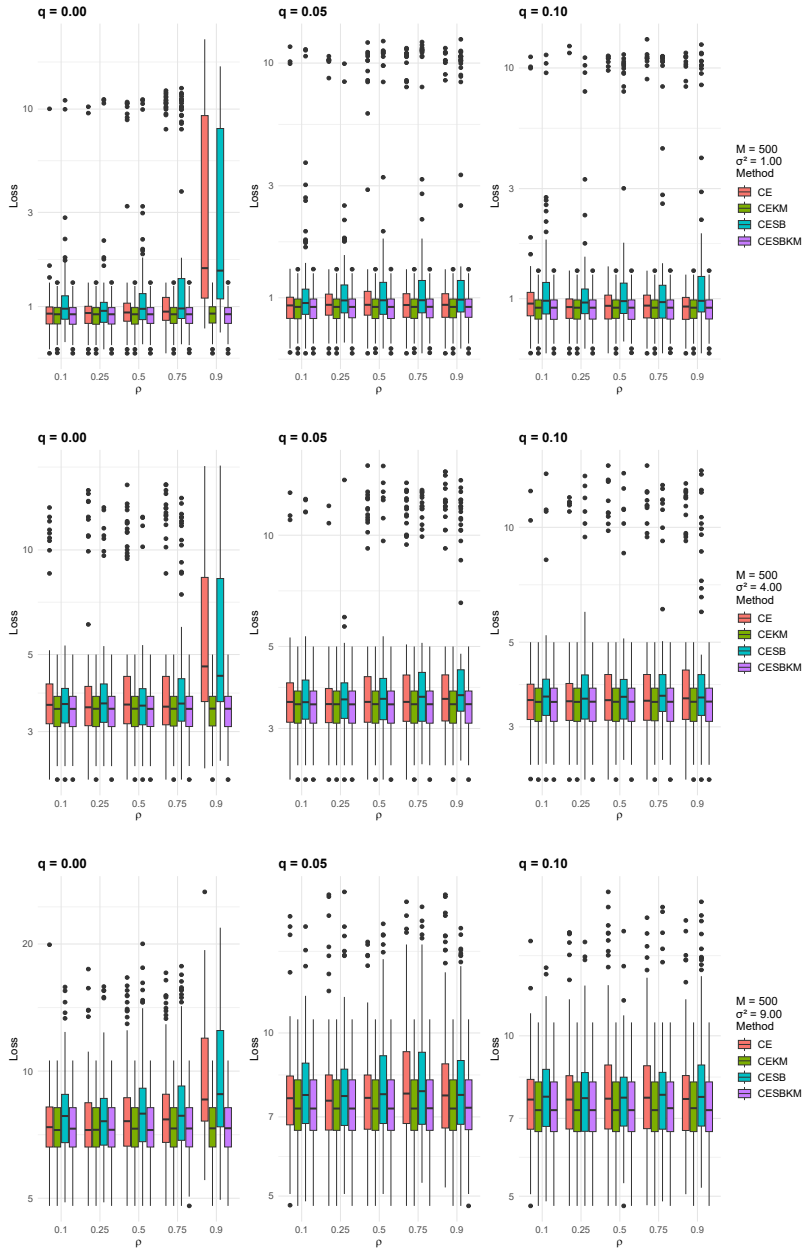


Figure 6.20 – Loss values (y-axis in log-scale) for Threshold model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

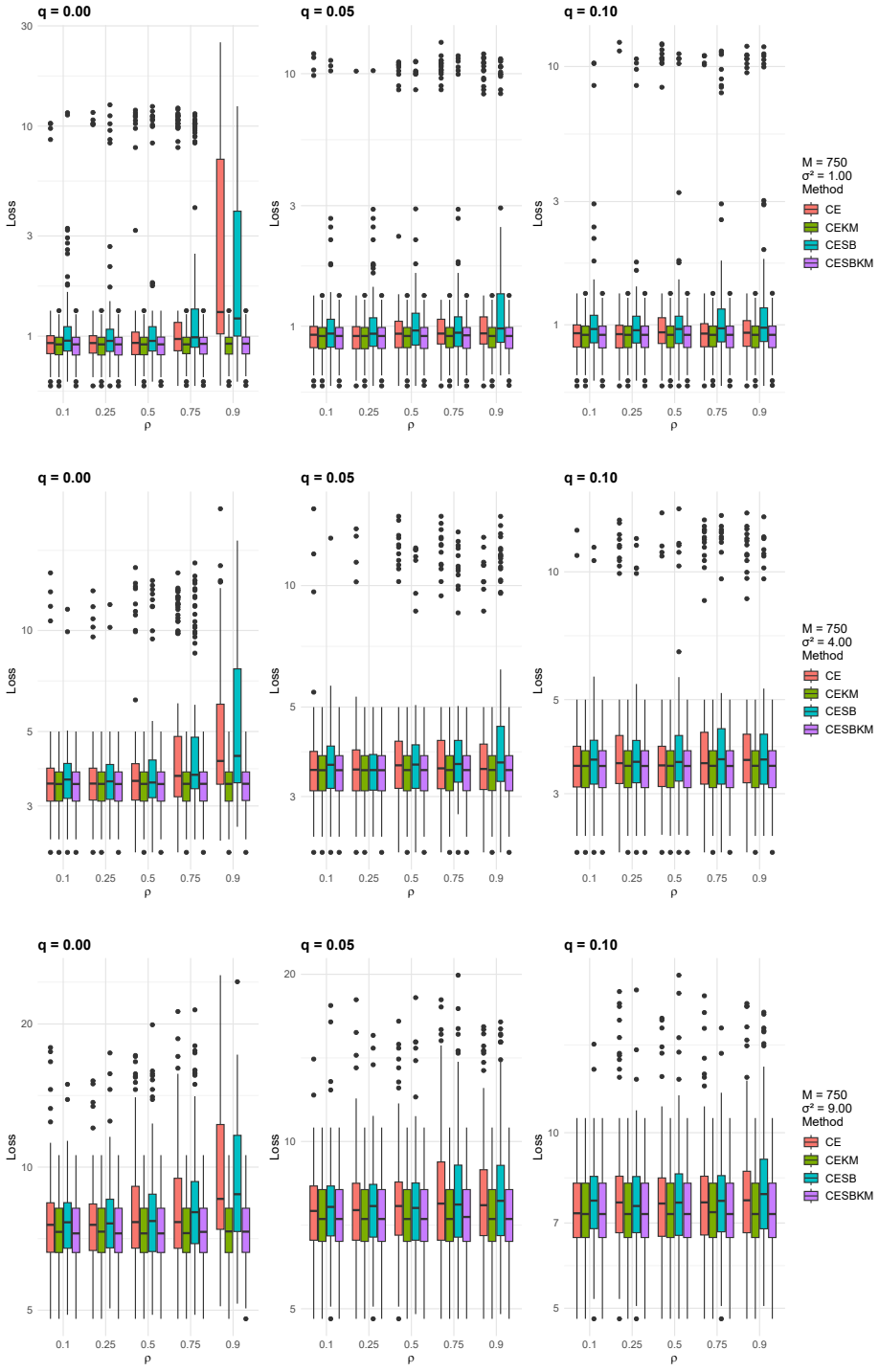


Figure 6.21 – Loss values (y-axis in log-scale) for Threshold model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

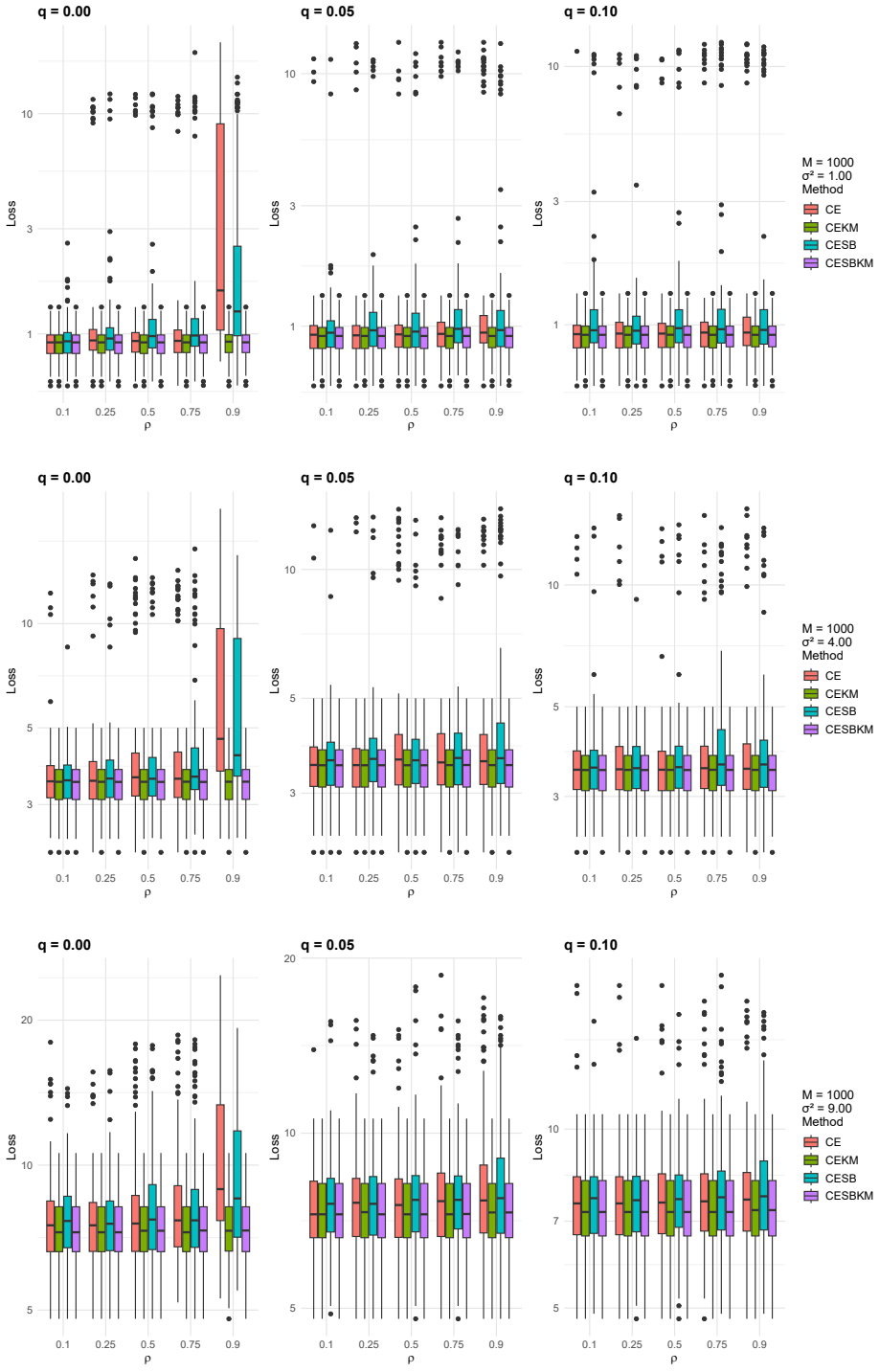


Figure 6.22 – Loss values (y-axis in log-scale) for Threshold model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

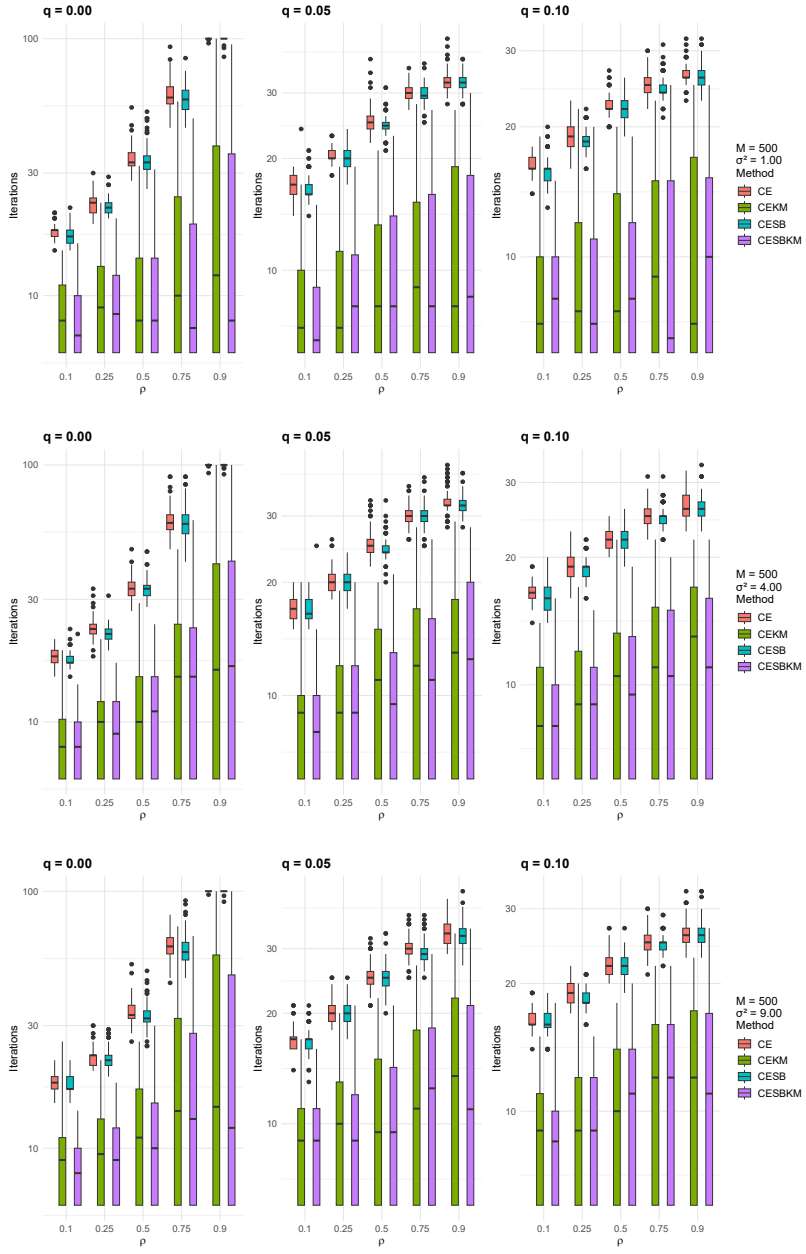


Figure 6.23 – Number of Iterations (y-axis in log-scale) for Threshold model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

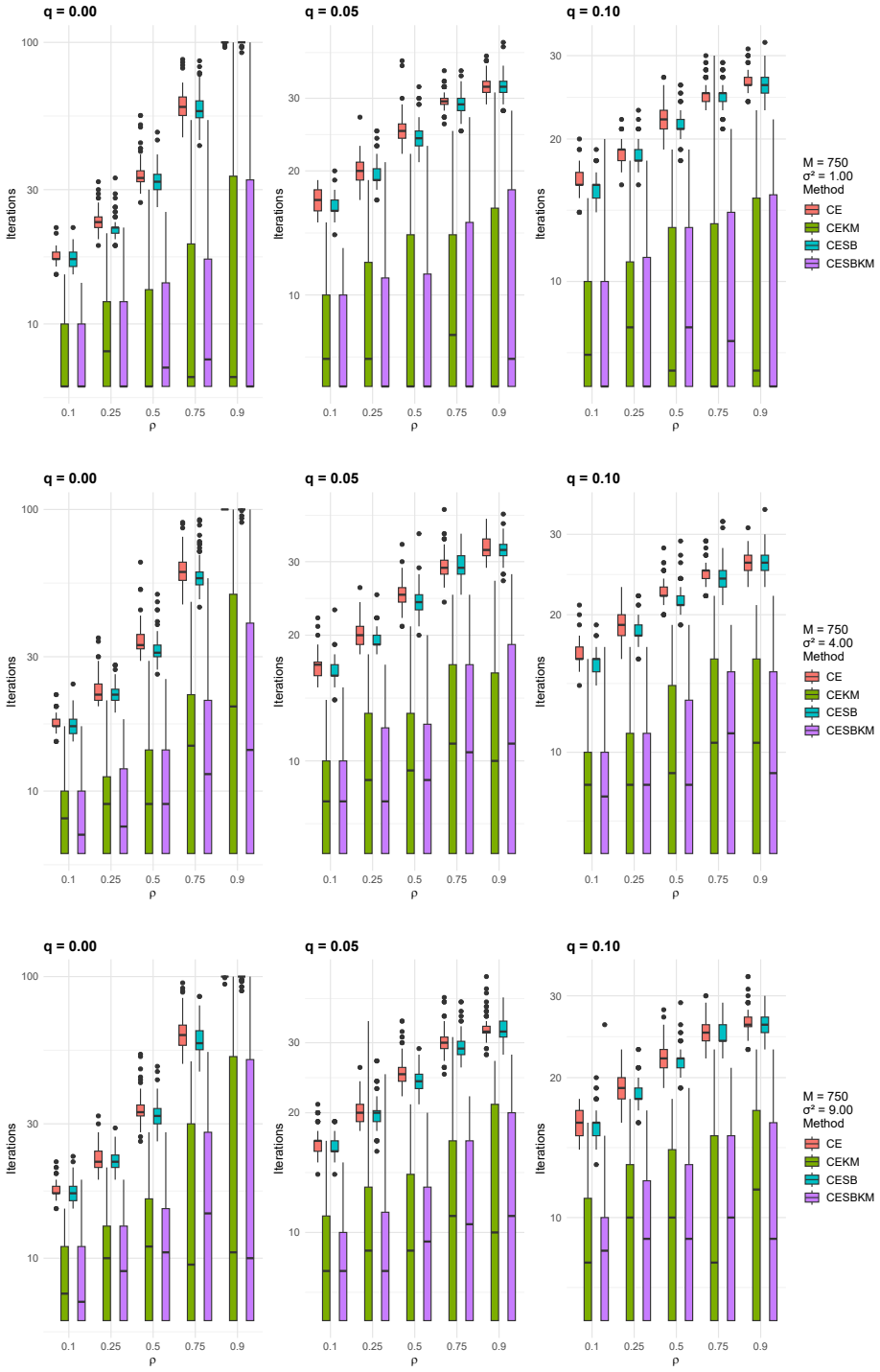


Figure 6.24 – Number of Iterations (y-axis in log-scale) for Threshold model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

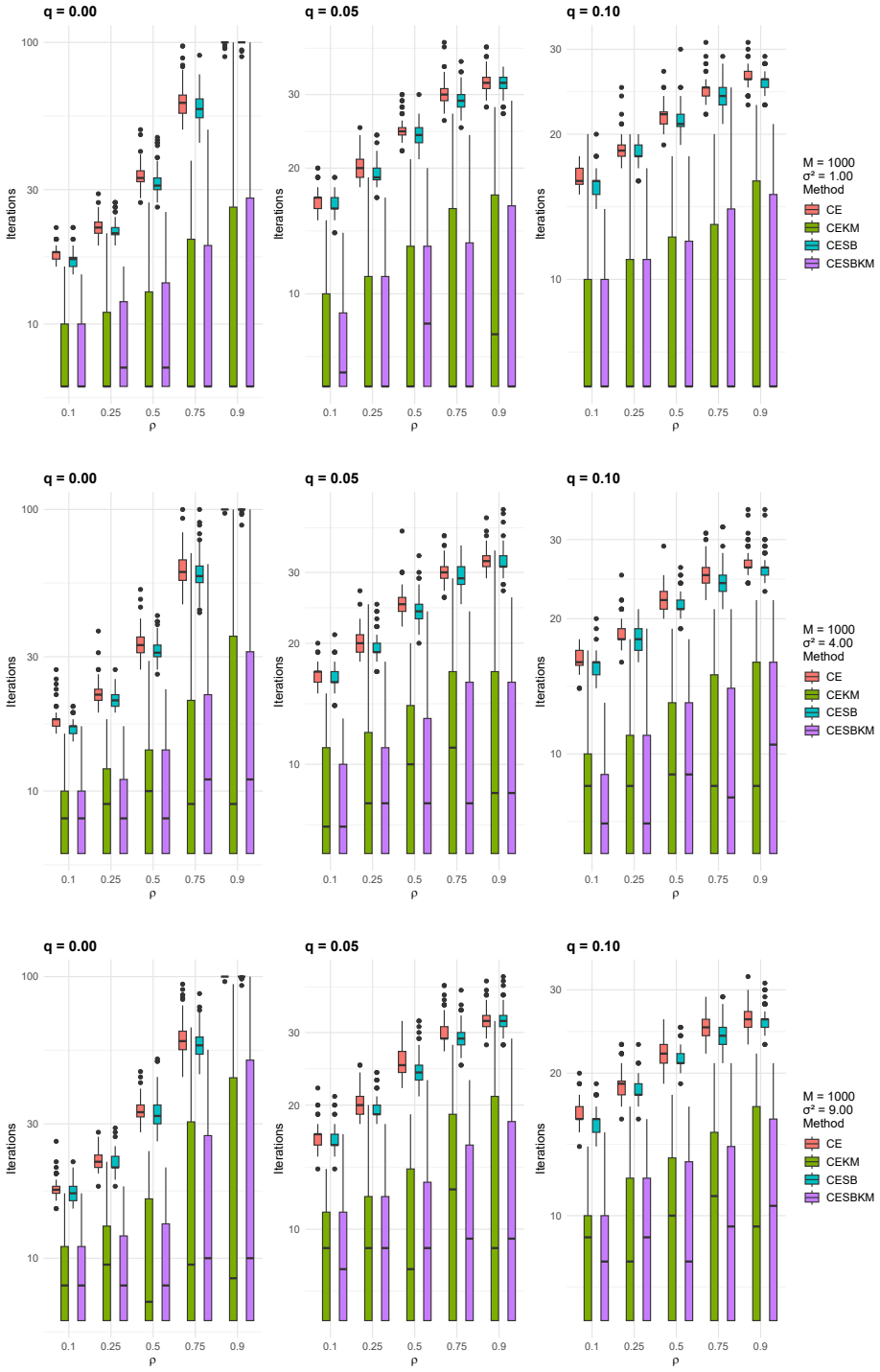


Figure 6.25 – Number of Iterations (y-axis in log-scale) for Threshold model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

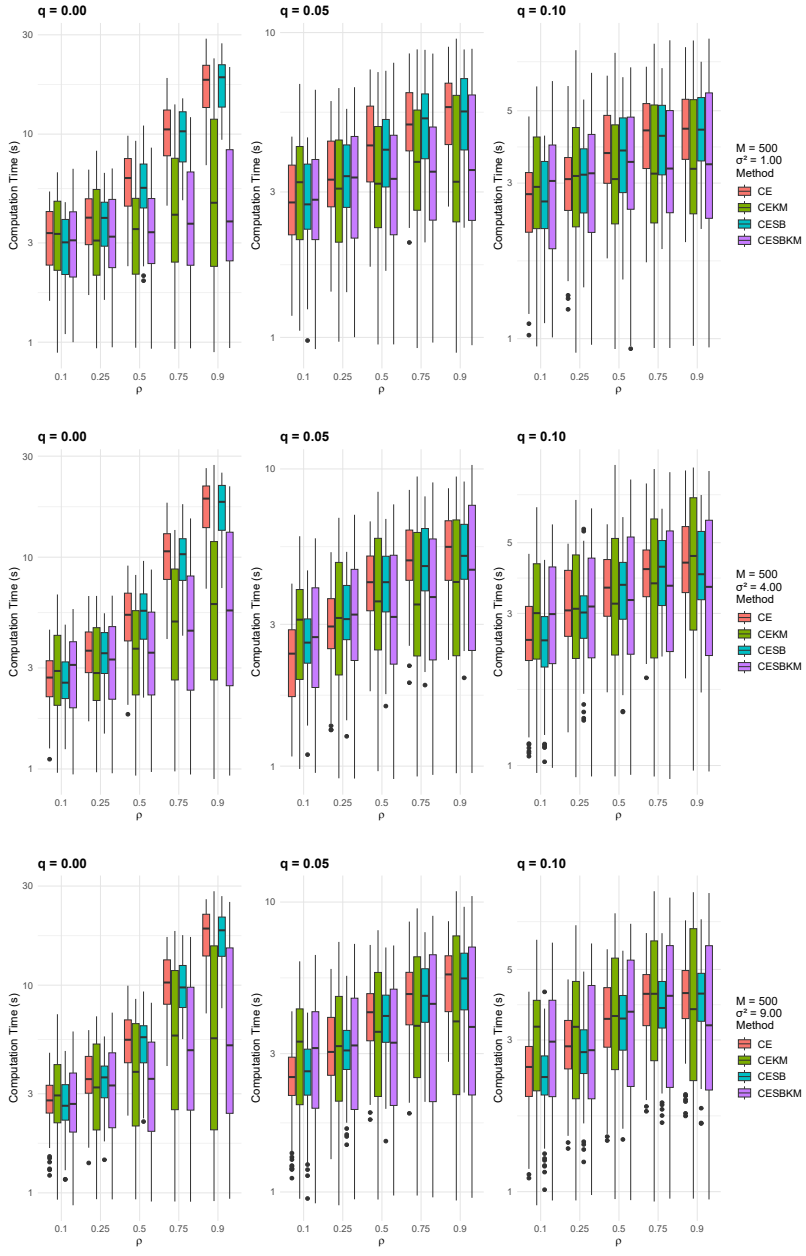


Figure 6.26 – Computation Time (y-axis in log-scale) for Threshold model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

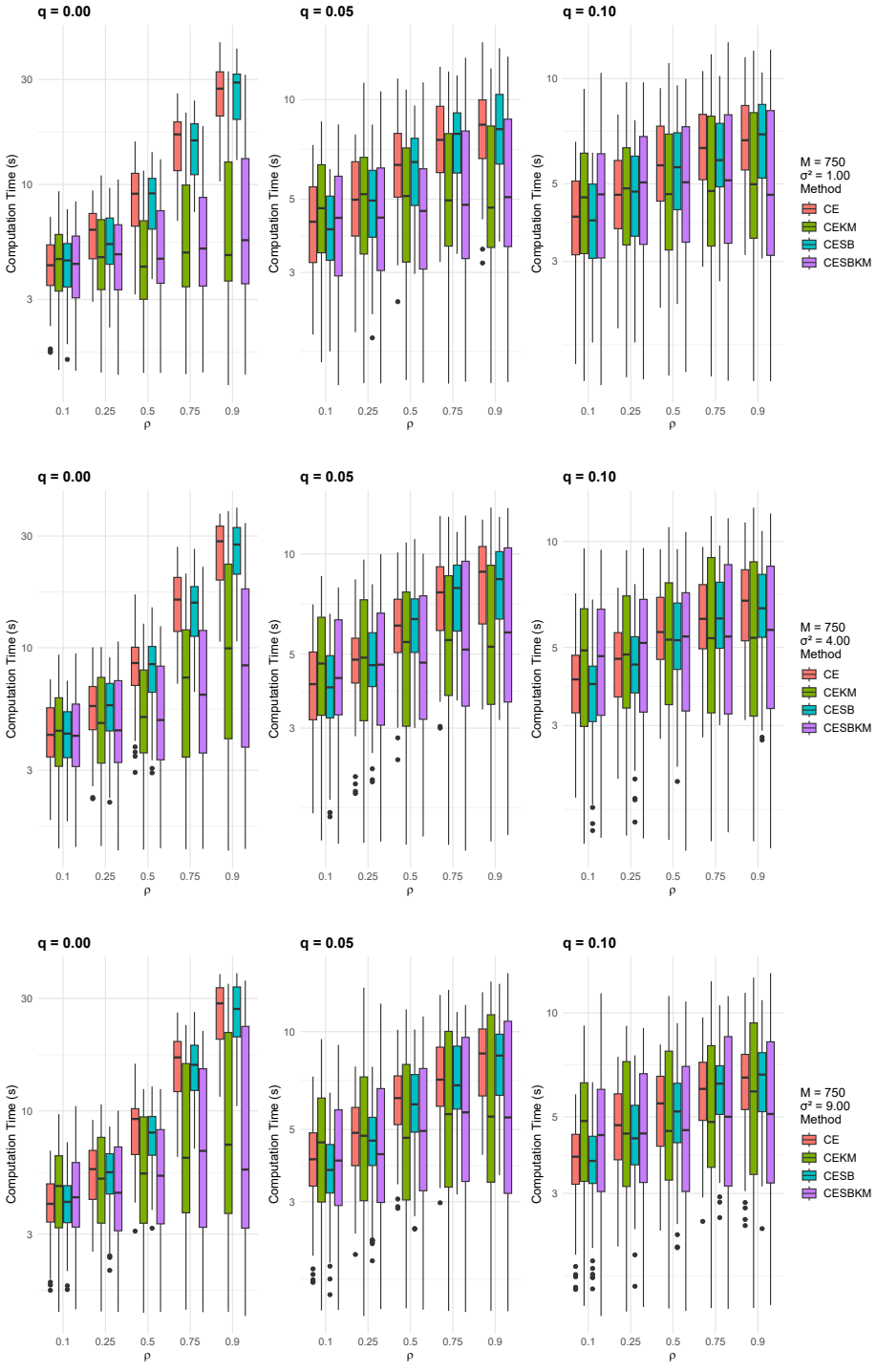


Figure 6.27 – Computation Time (y-axis in log-scale) for Threshold model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

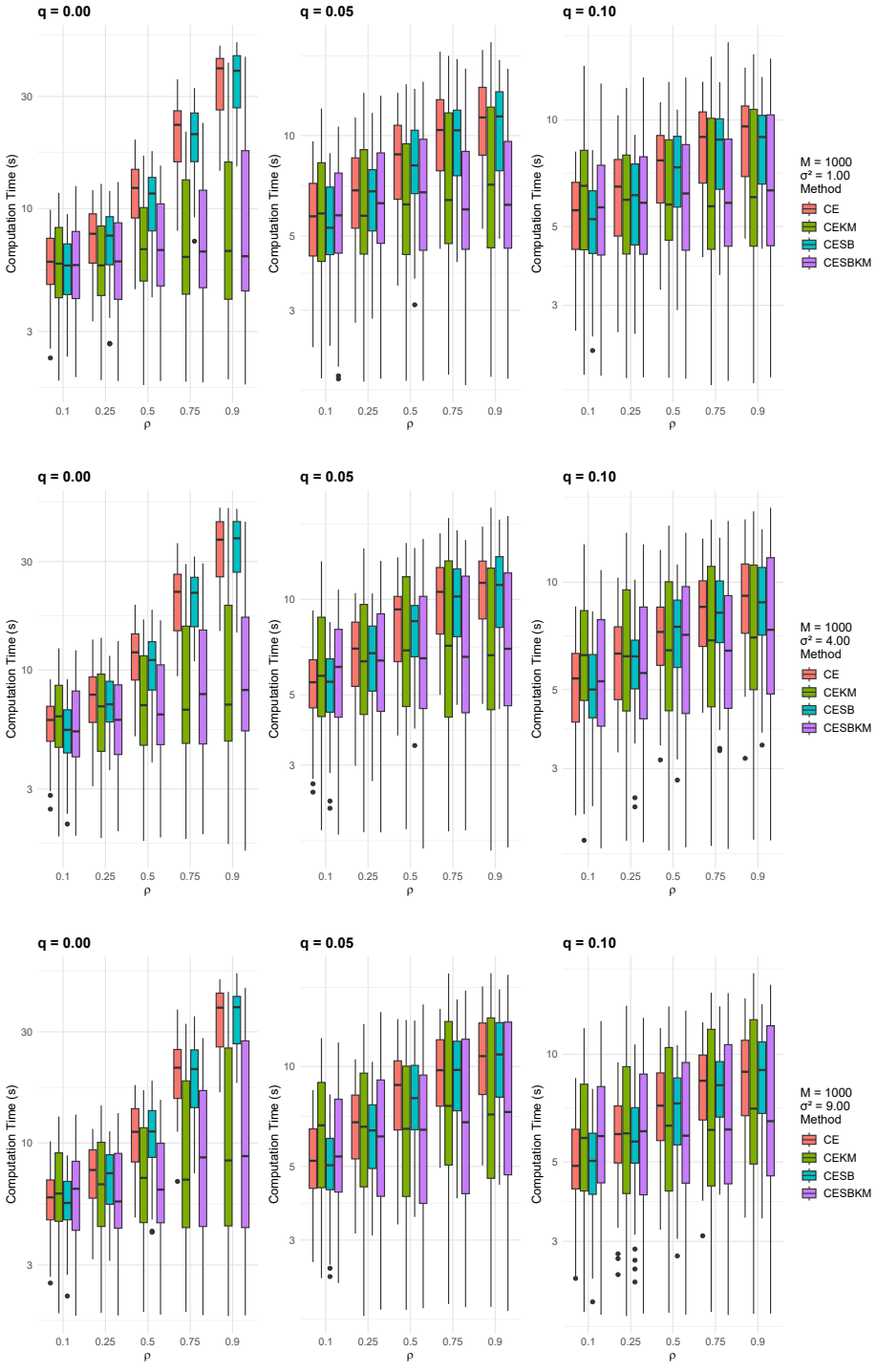


Figure 6.28 – Computation Time (y-axis in log-scale) for Threshold model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

Pure Noise Model With SBC On Fitted Values

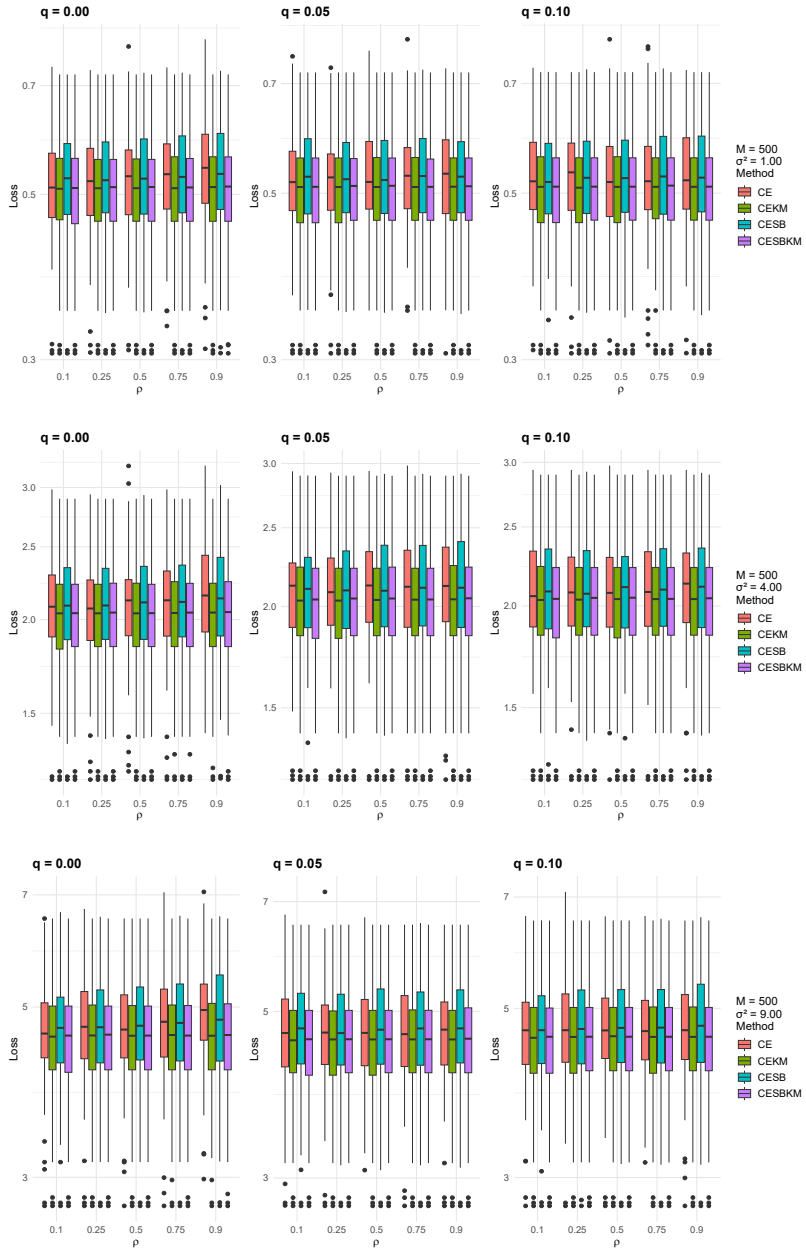


Figure 6.29 – Loss values (y-axis in log-scale) for Noise model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

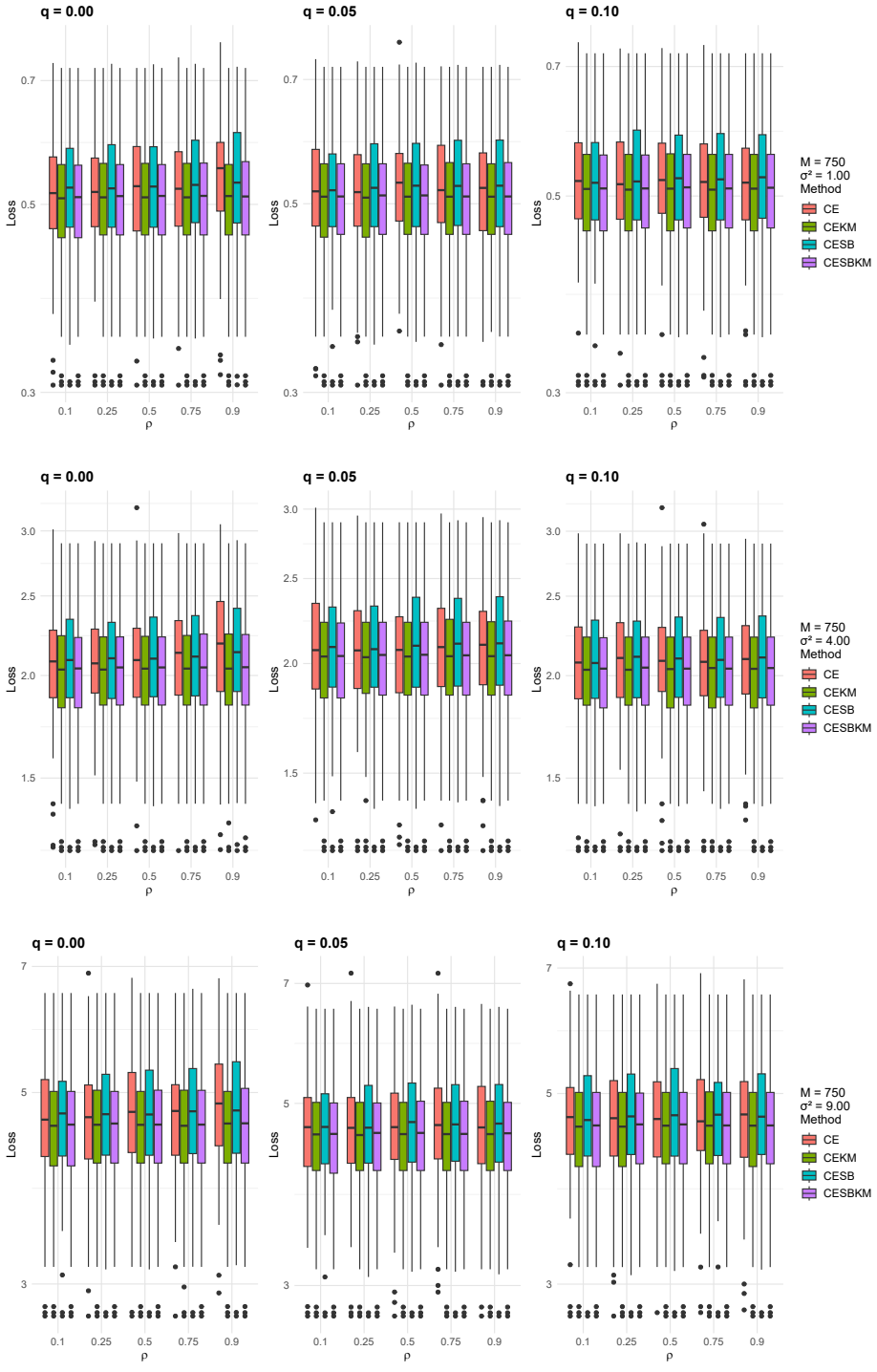


Figure 6.30 – Loss values (y-axis in log-scale) for Noise model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

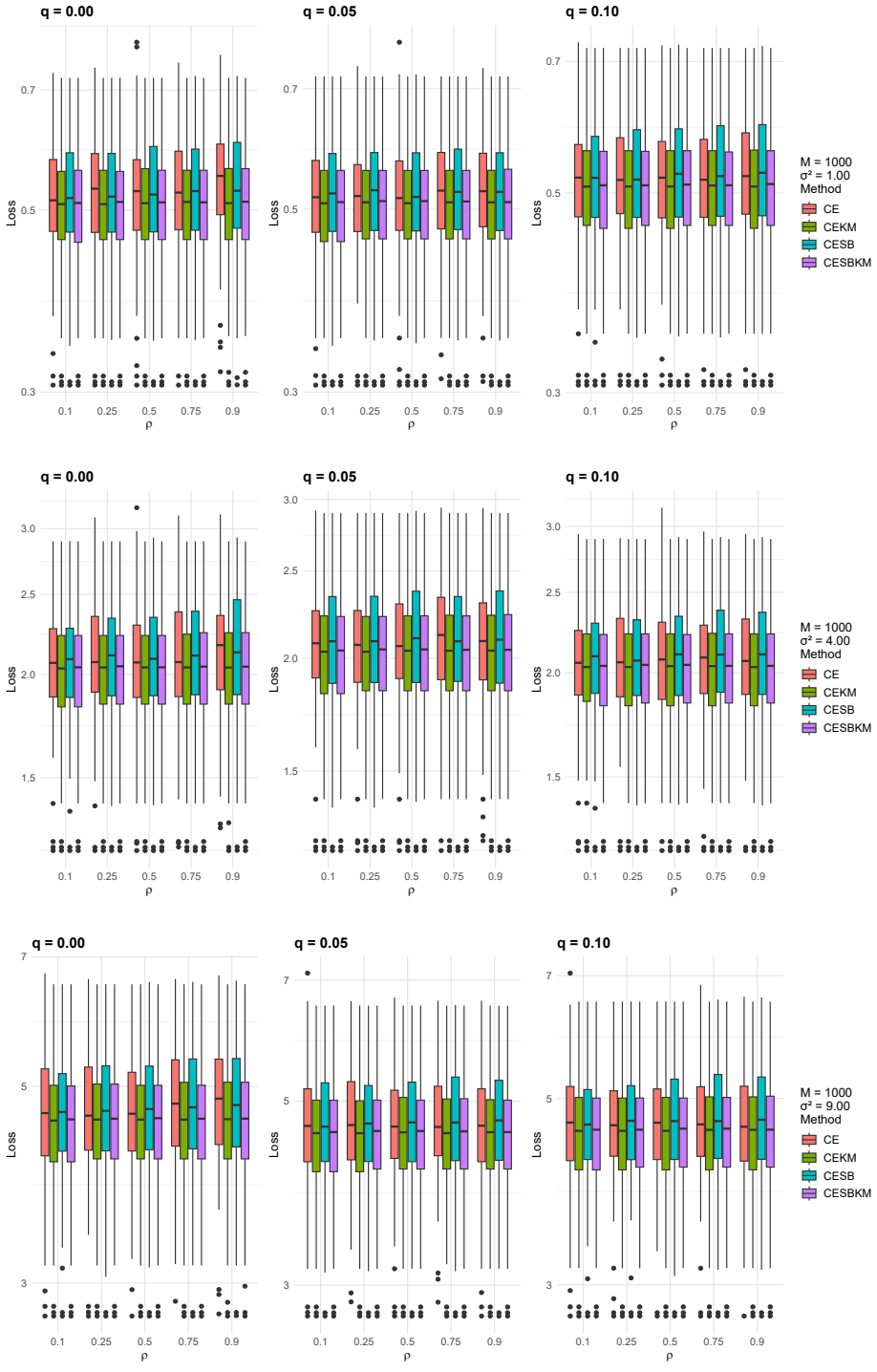


Figure 6.31 – Loss values (y-axis in log-scale) for Noise model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

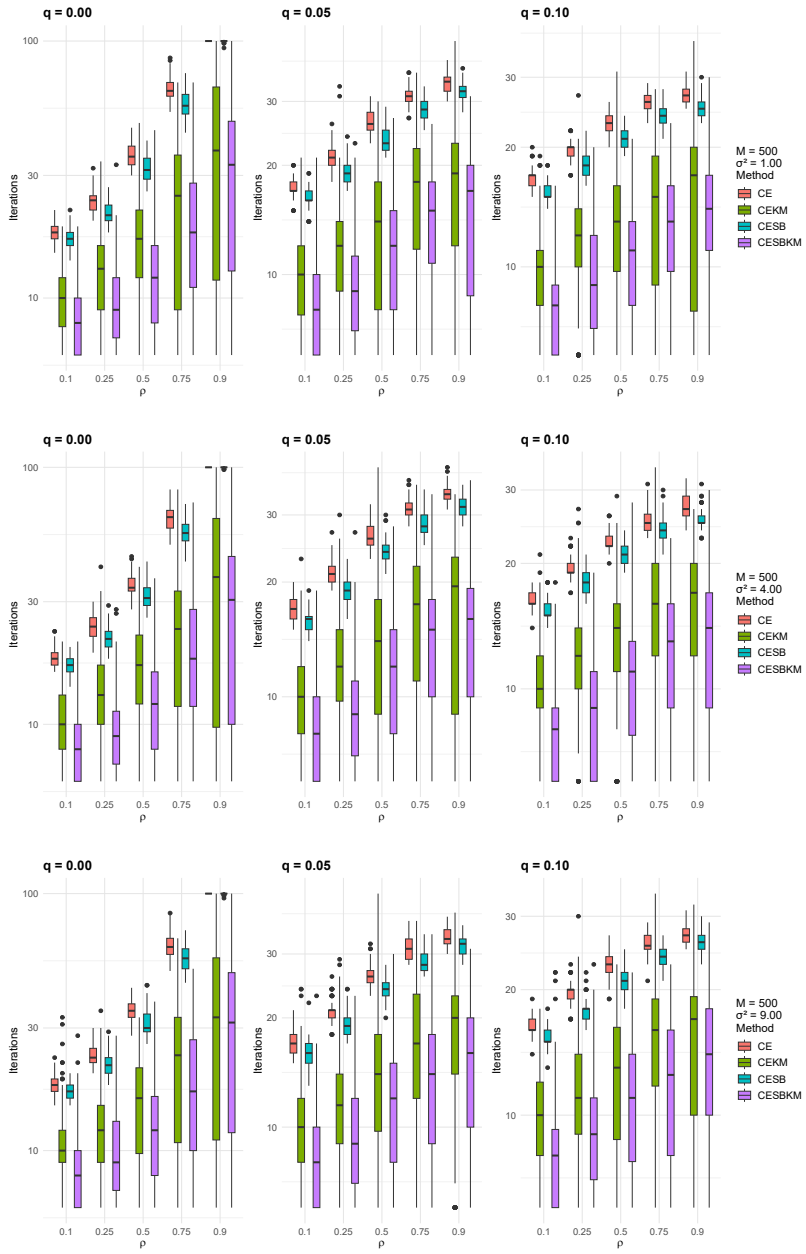


Figure 6.32 – Number of Iterations (y-axis in log-scale) for Noise model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

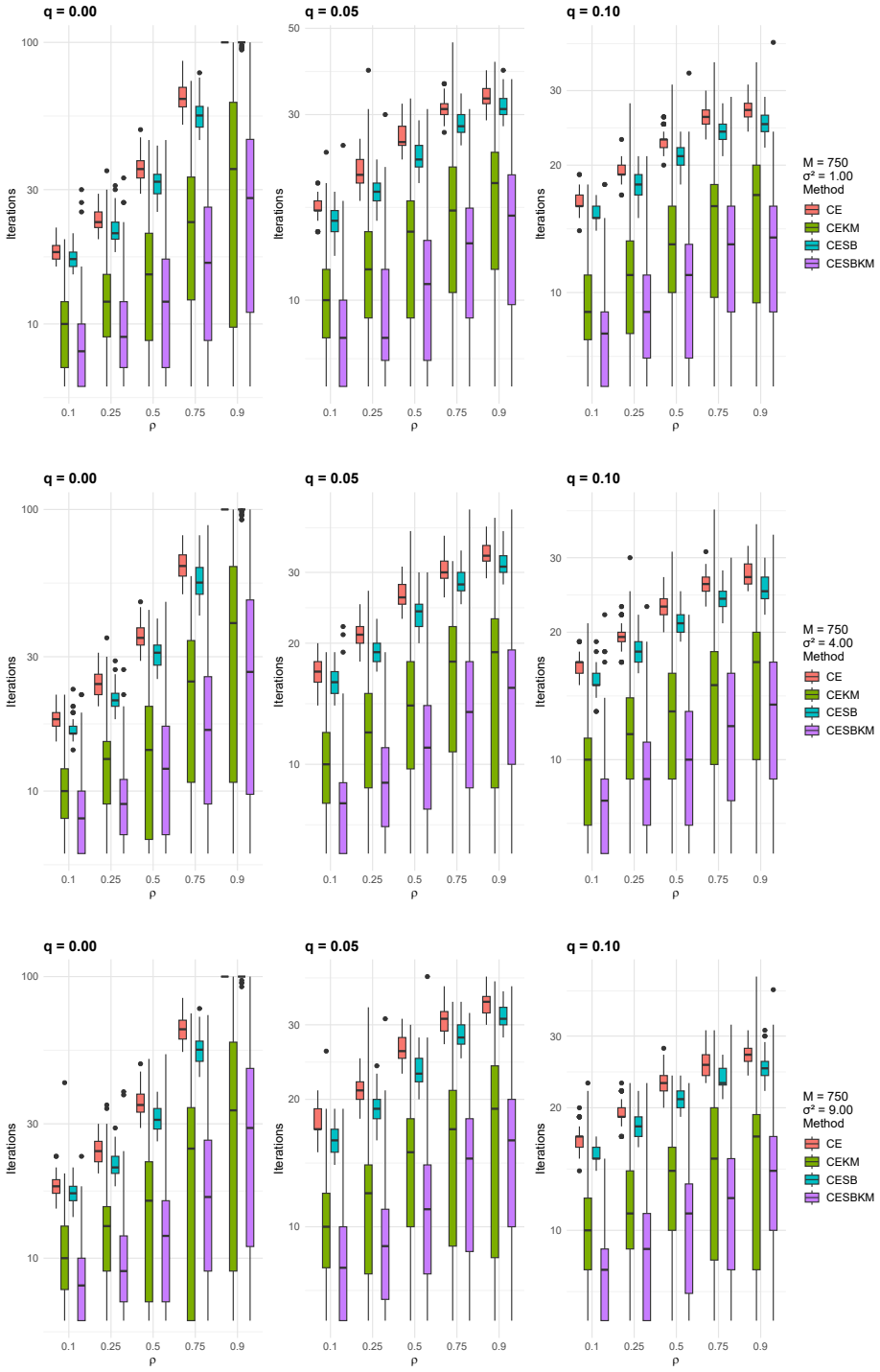


Figure 6.33 – Number of Iterations (y-axis in log-scale) for Noise model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

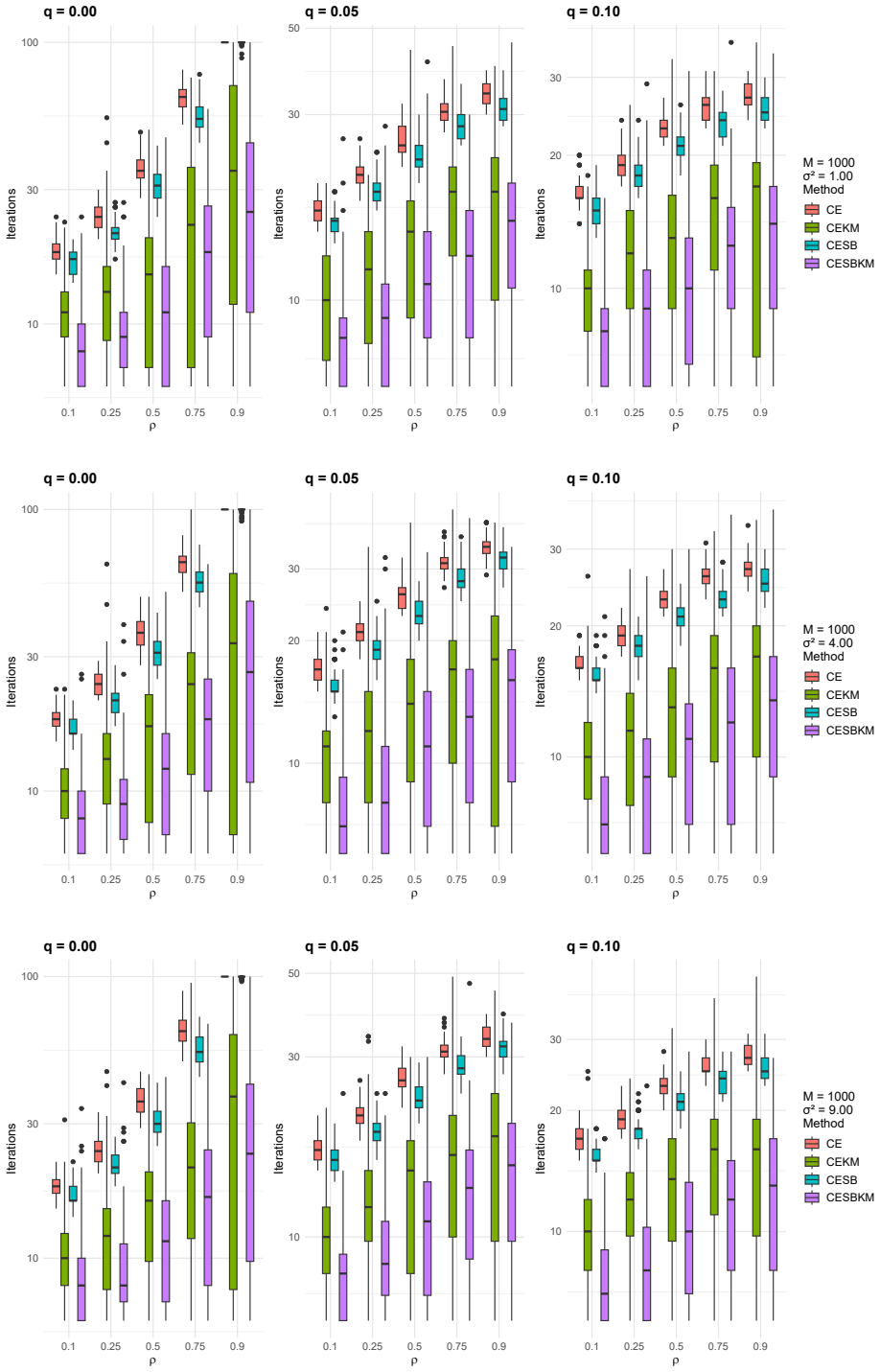


Figure 6.34 – Number of Iterations (y-axis in log-scale) for Noise model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

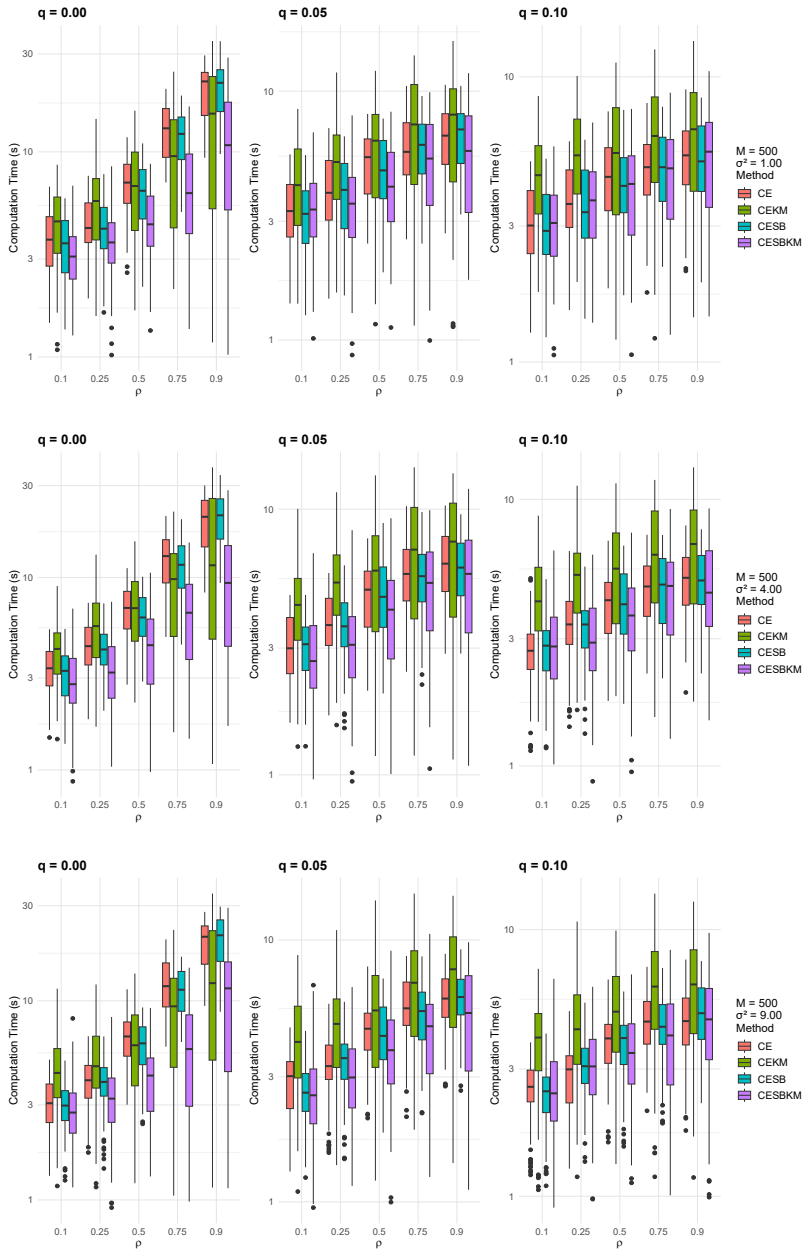


Figure 6.35 – Computation Time (y-axis in log-scale) for Noise model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

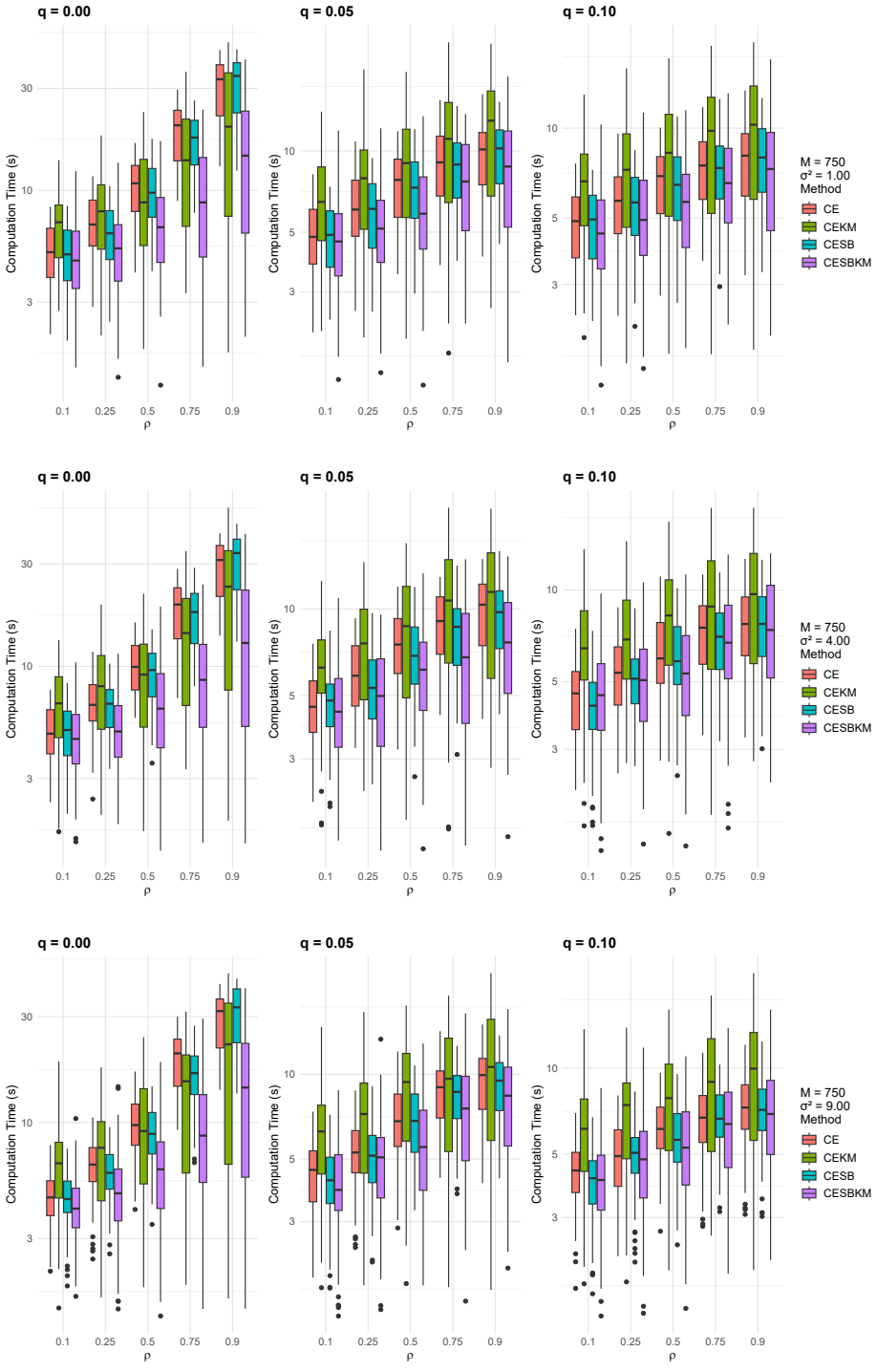


Figure 6.36 – Computation Time (y-axis in log-scale) for Noise model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

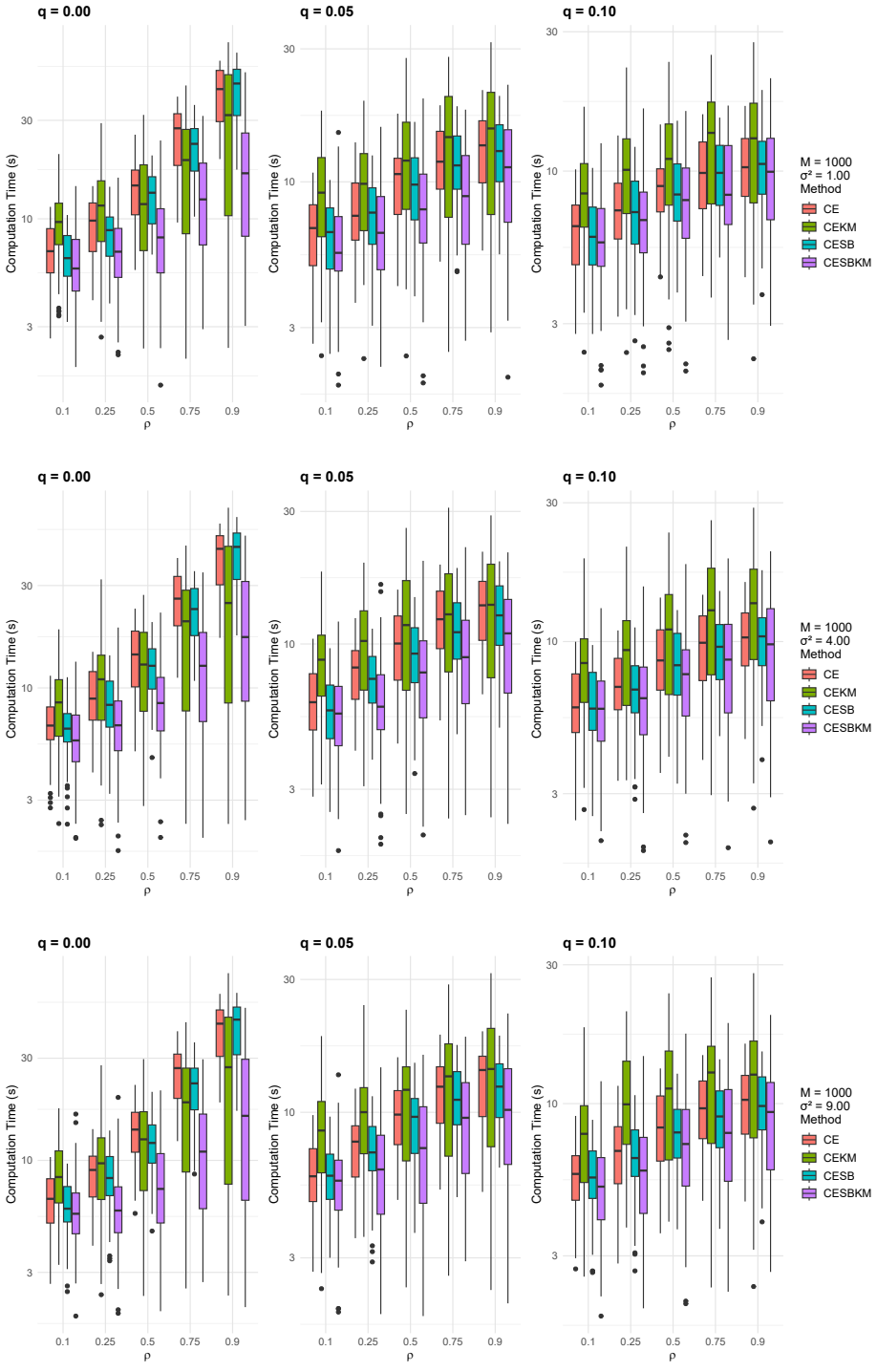


Figure 6.37 – Computation Time (y-axis in log-scale) for Noise model with SBC on fitted values: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

Pure Noise Model With SBC On Centroids

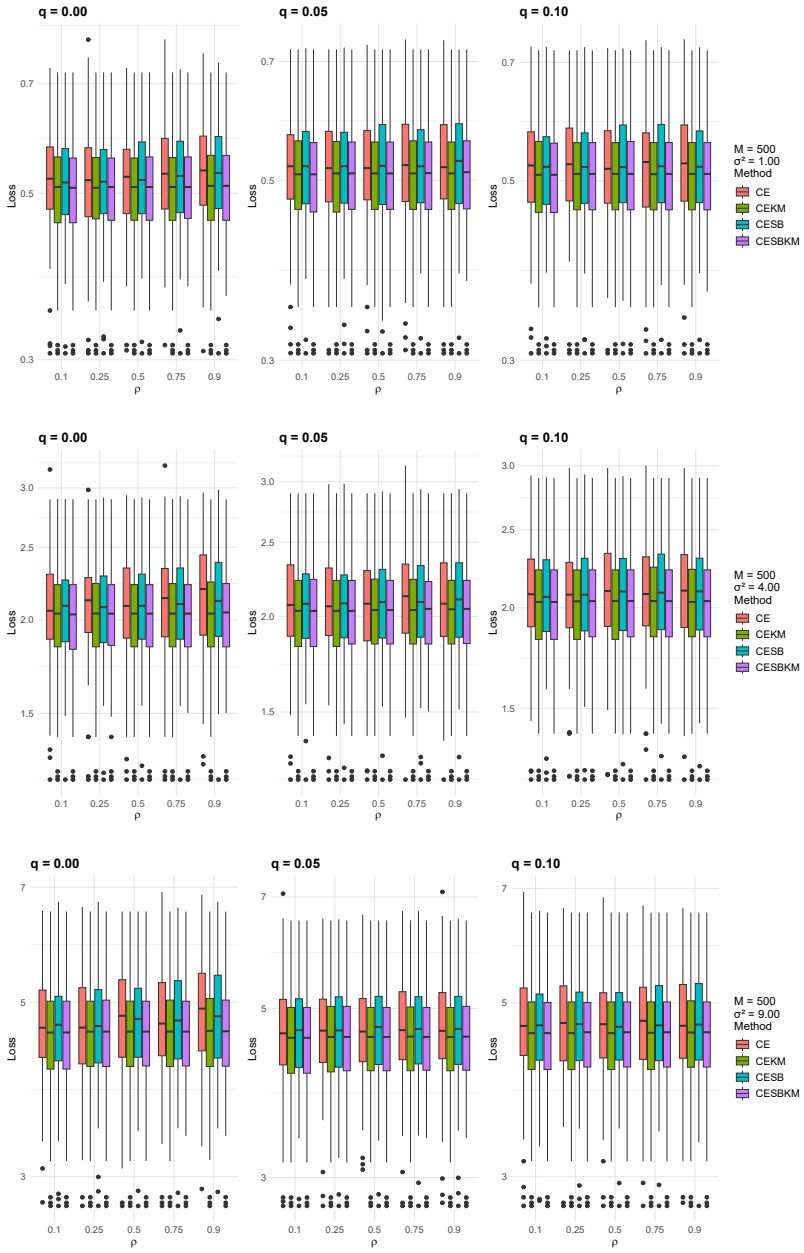


Figure 6.38 – Loss values (y-axis in log-scale) for Noise model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

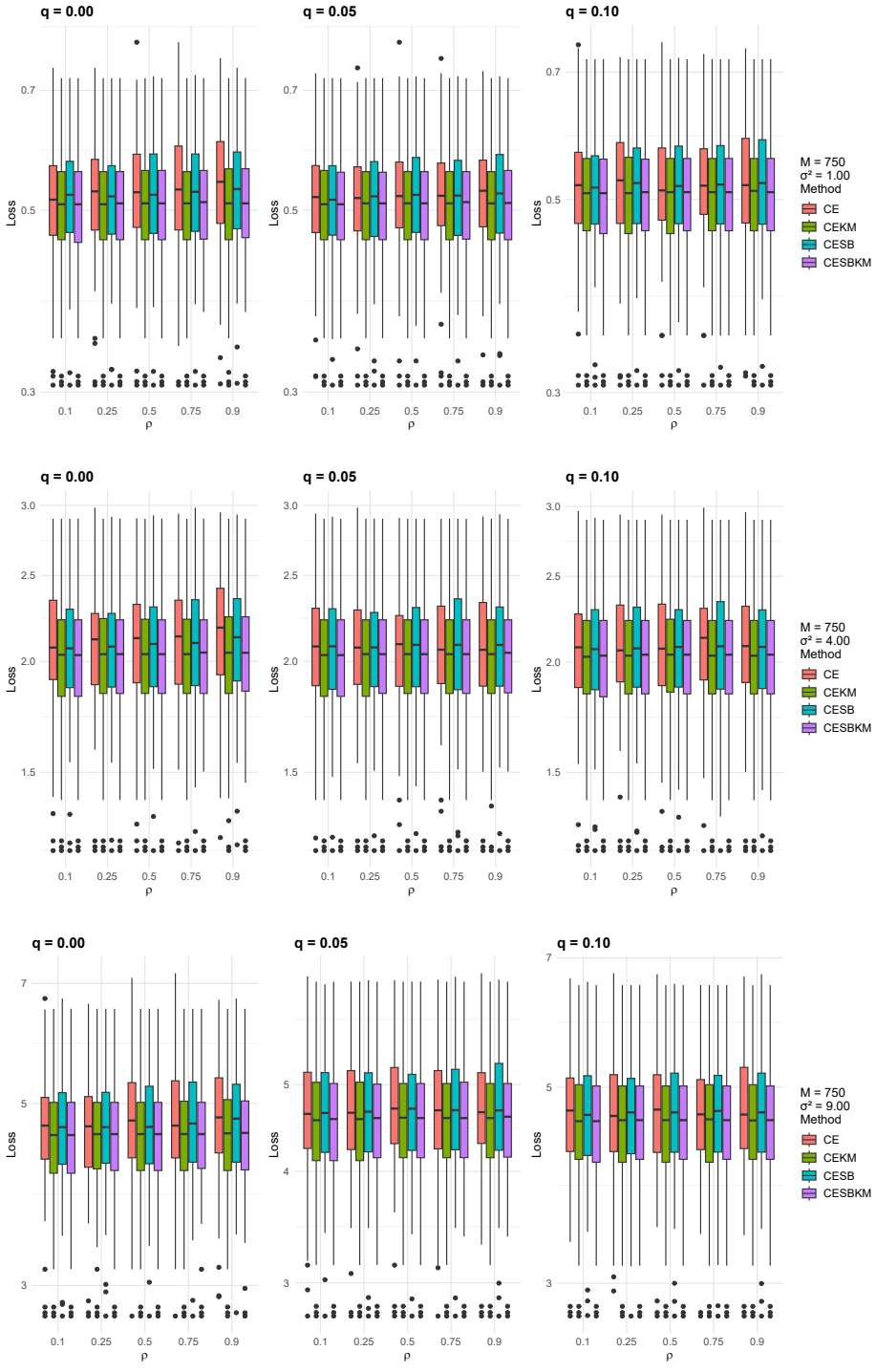


Figure 6.39 – Loss values (y-axis in log-scale) for Noise model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

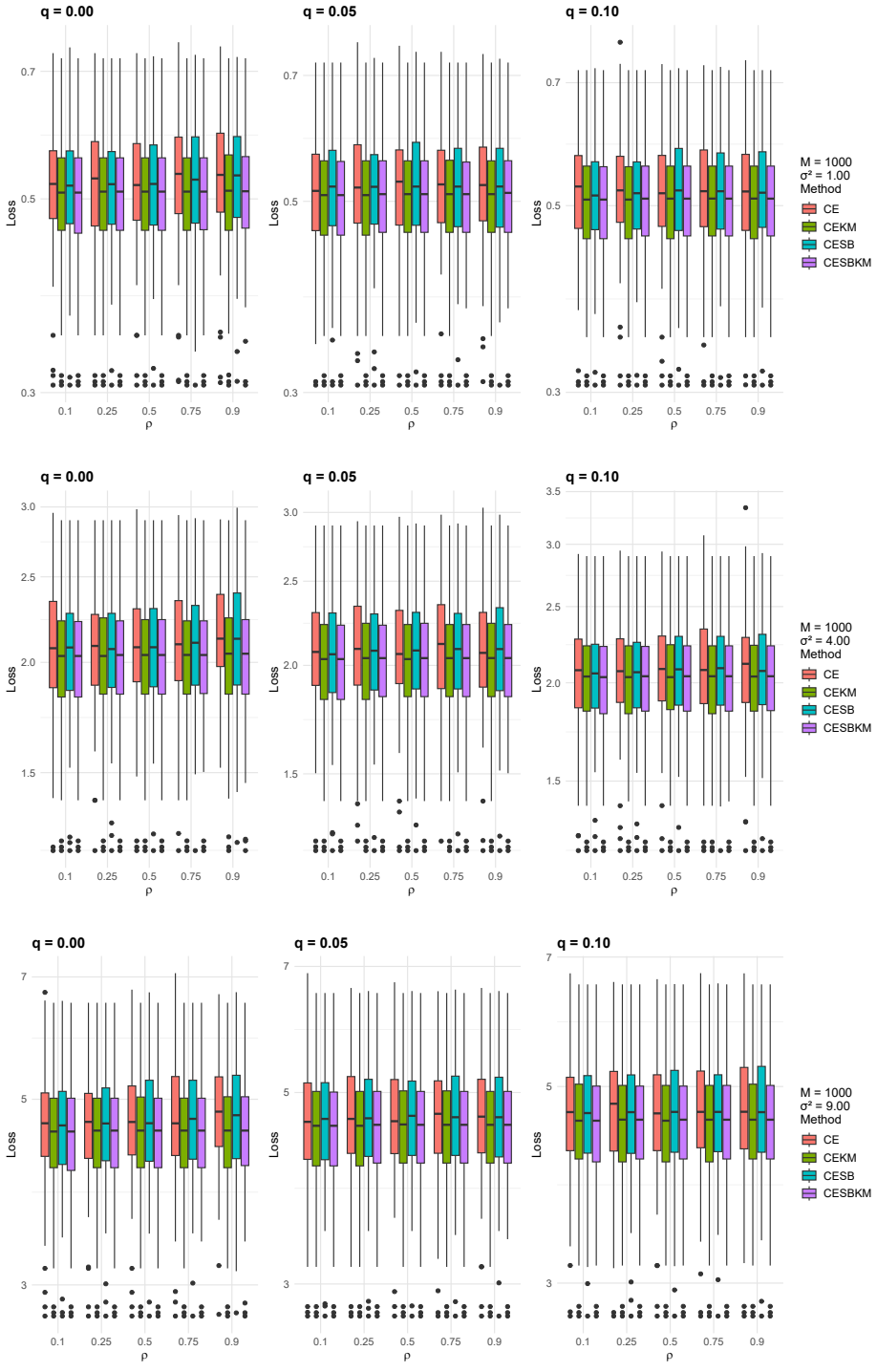


Figure 6.40 – Loss values (y-axis in log-scale) for Noise model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

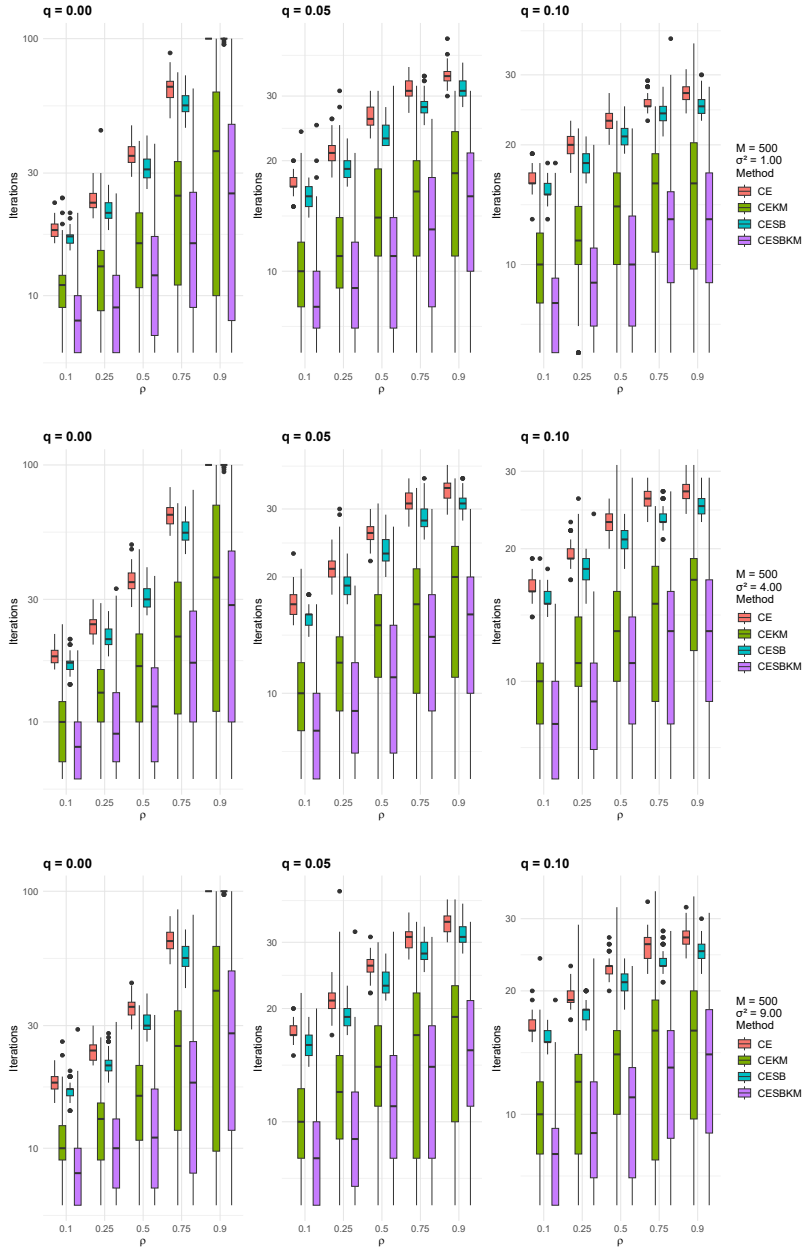


Figure 6.41 – Number of Iterations (y-axis in log-scale) for Noise model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

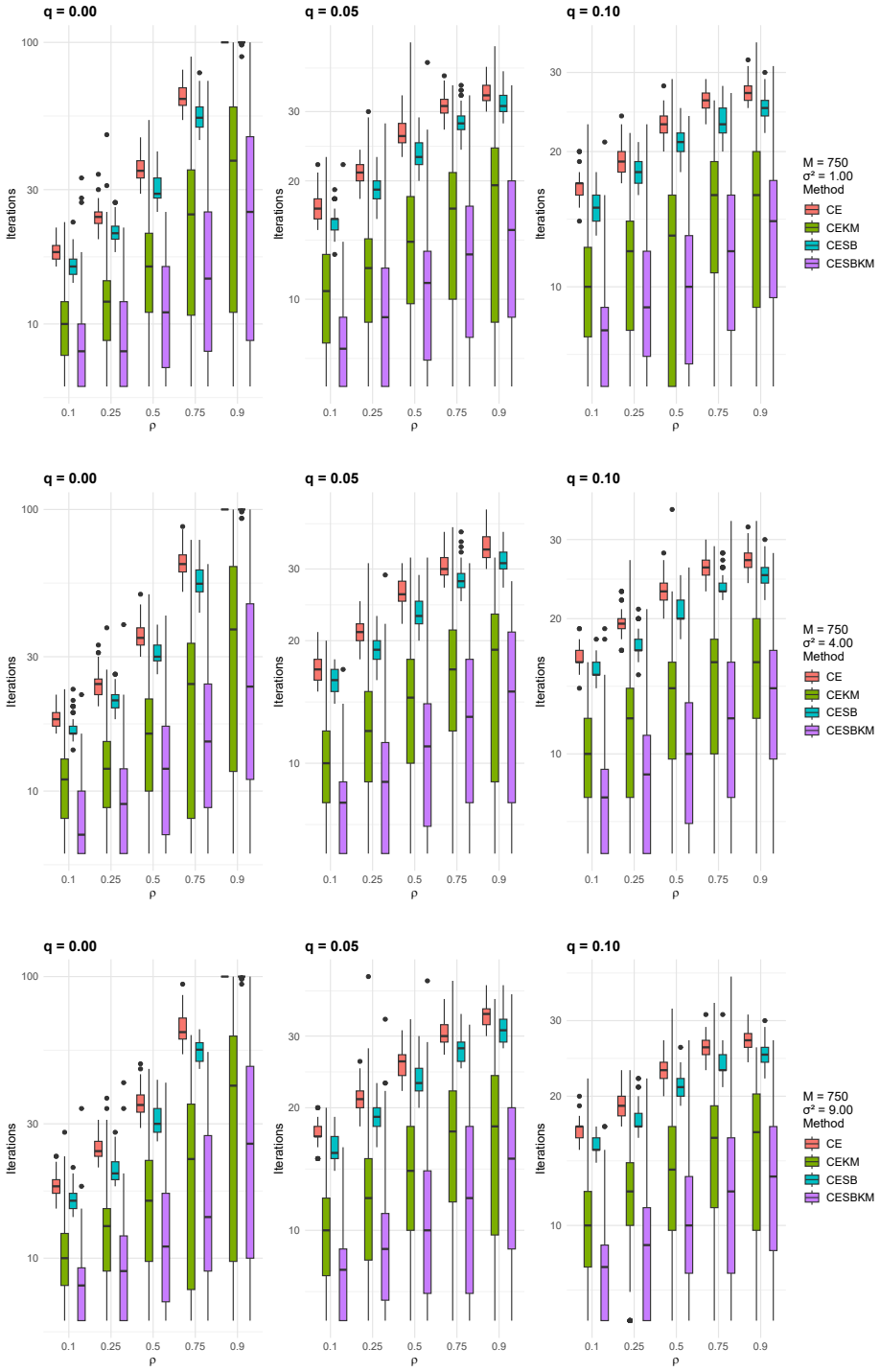


Figure 6.42 – Number of Iterations (y-axis in log-scale) for Noise model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

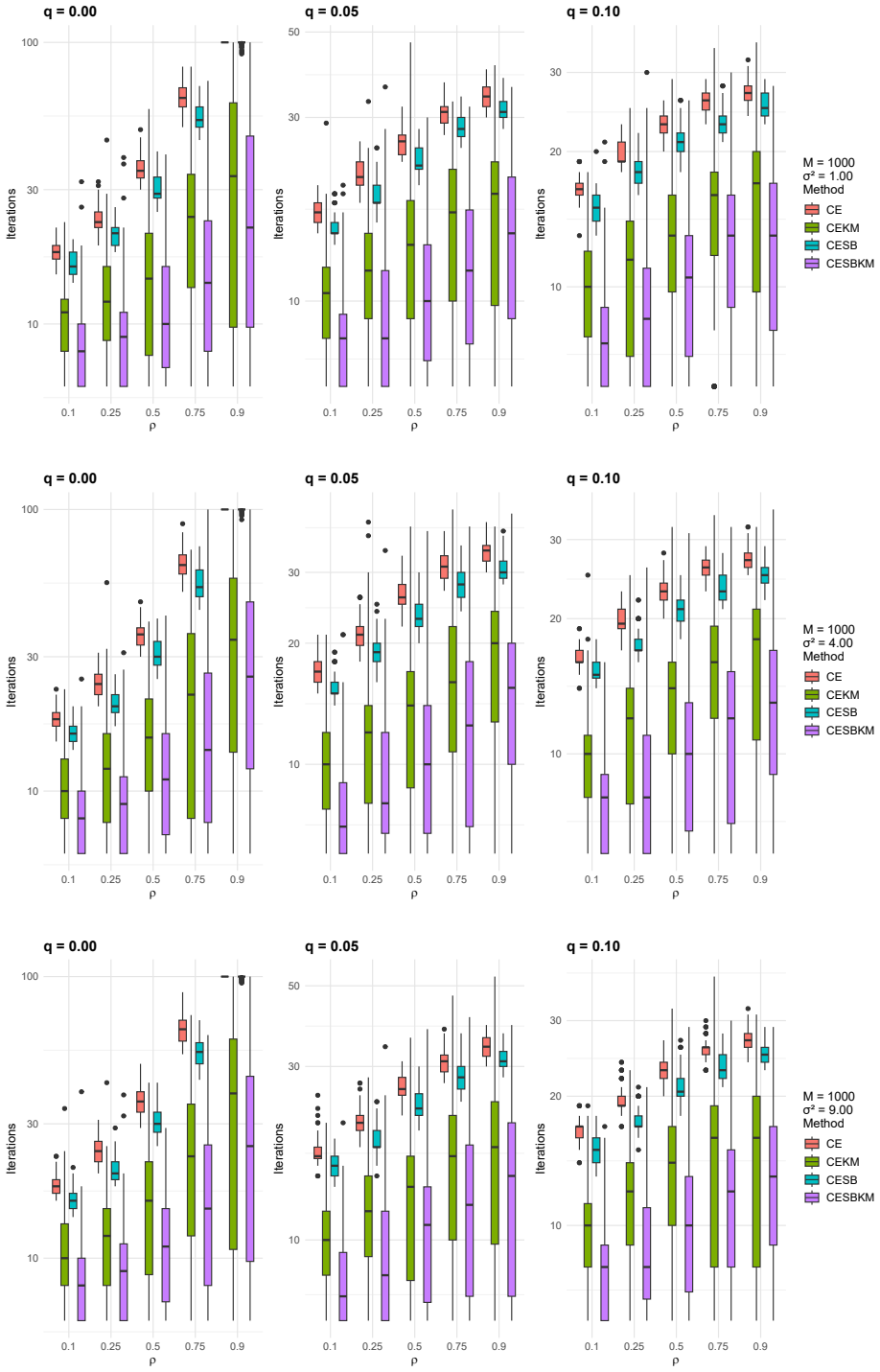


Figure 6.43 – Number of Iterations (y-axis in log-scale) for Noise model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

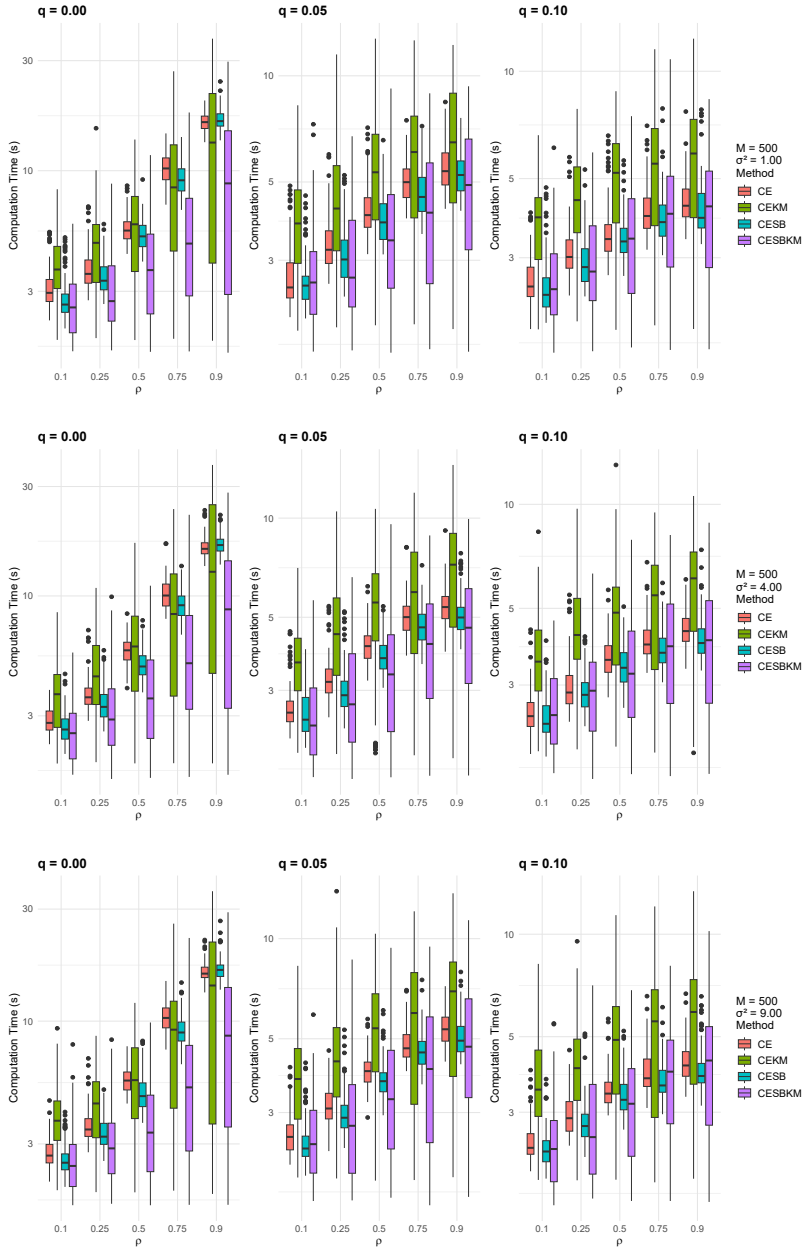


Figure 6.44 – Computation Time (y-axis in log-scale) for Noise model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

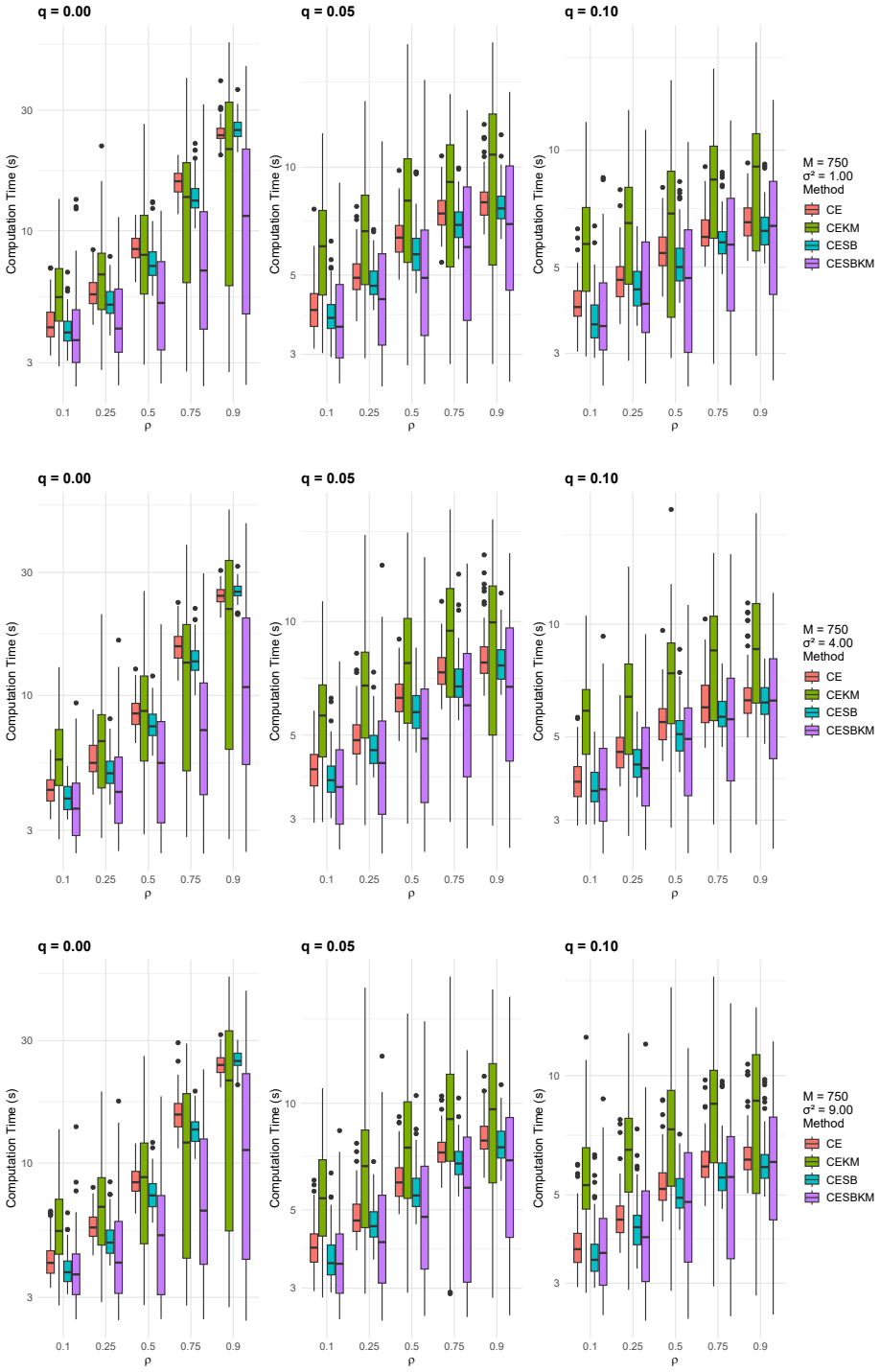


Figure 6.45 – Computation Time (y-axis in log-scale) for Noise model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

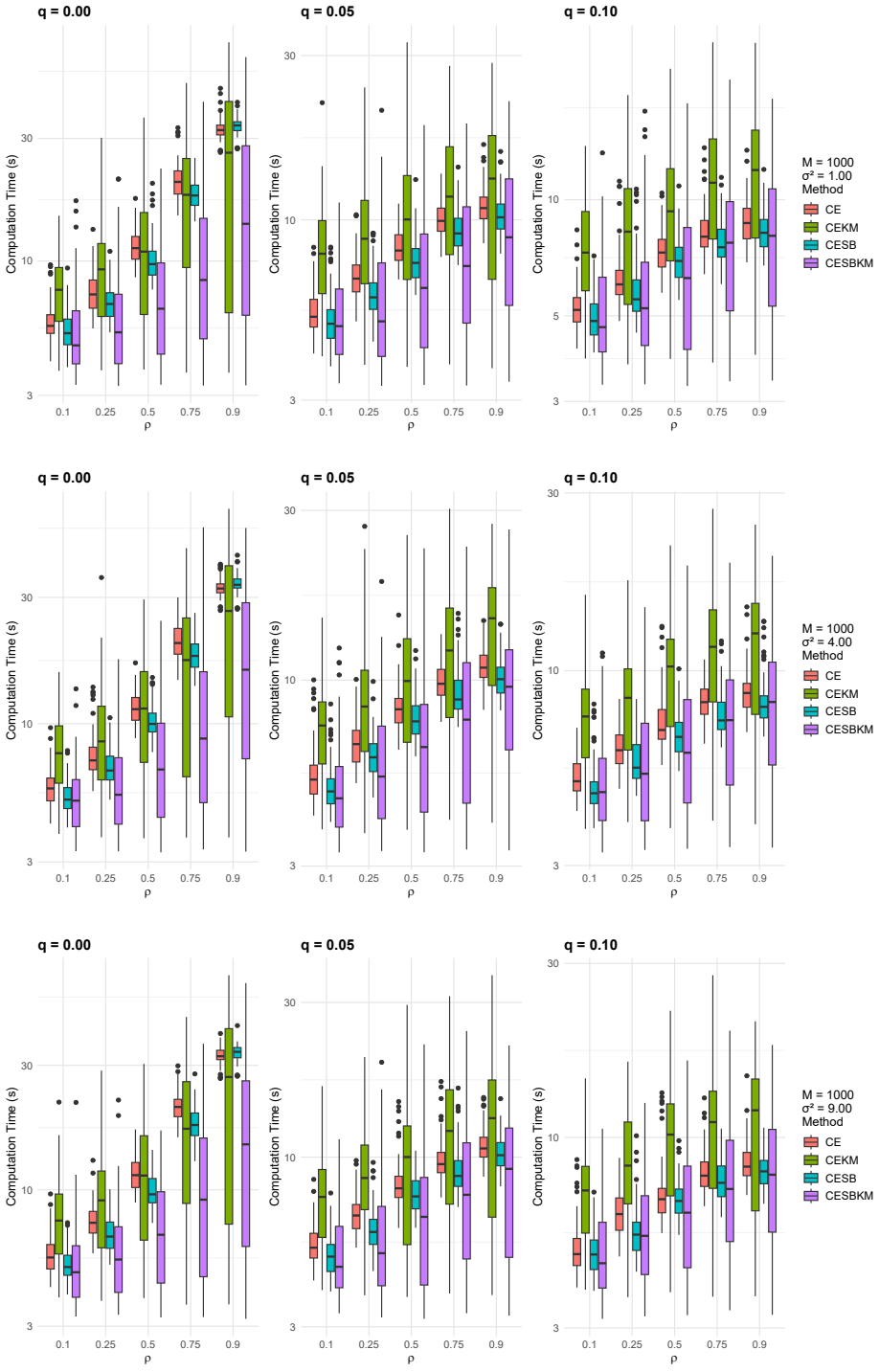


Figure 6.46 – Computation Time (y-axis in log-scale) for Noise model with SBC on centroids: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

Pure Noise Model With SBC On The First Element of The Cluster Vector

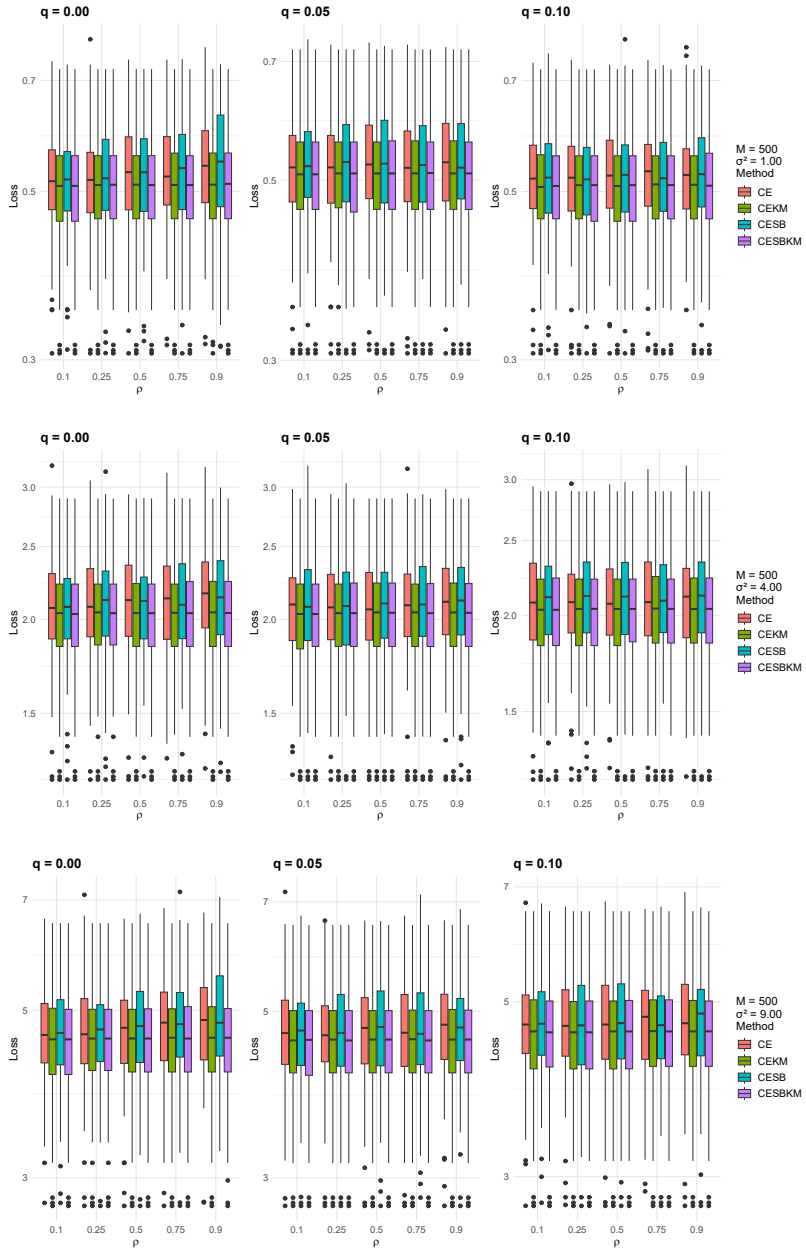


Figure 6.47 – Loss values (y-axis in log-scale) for Noise model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

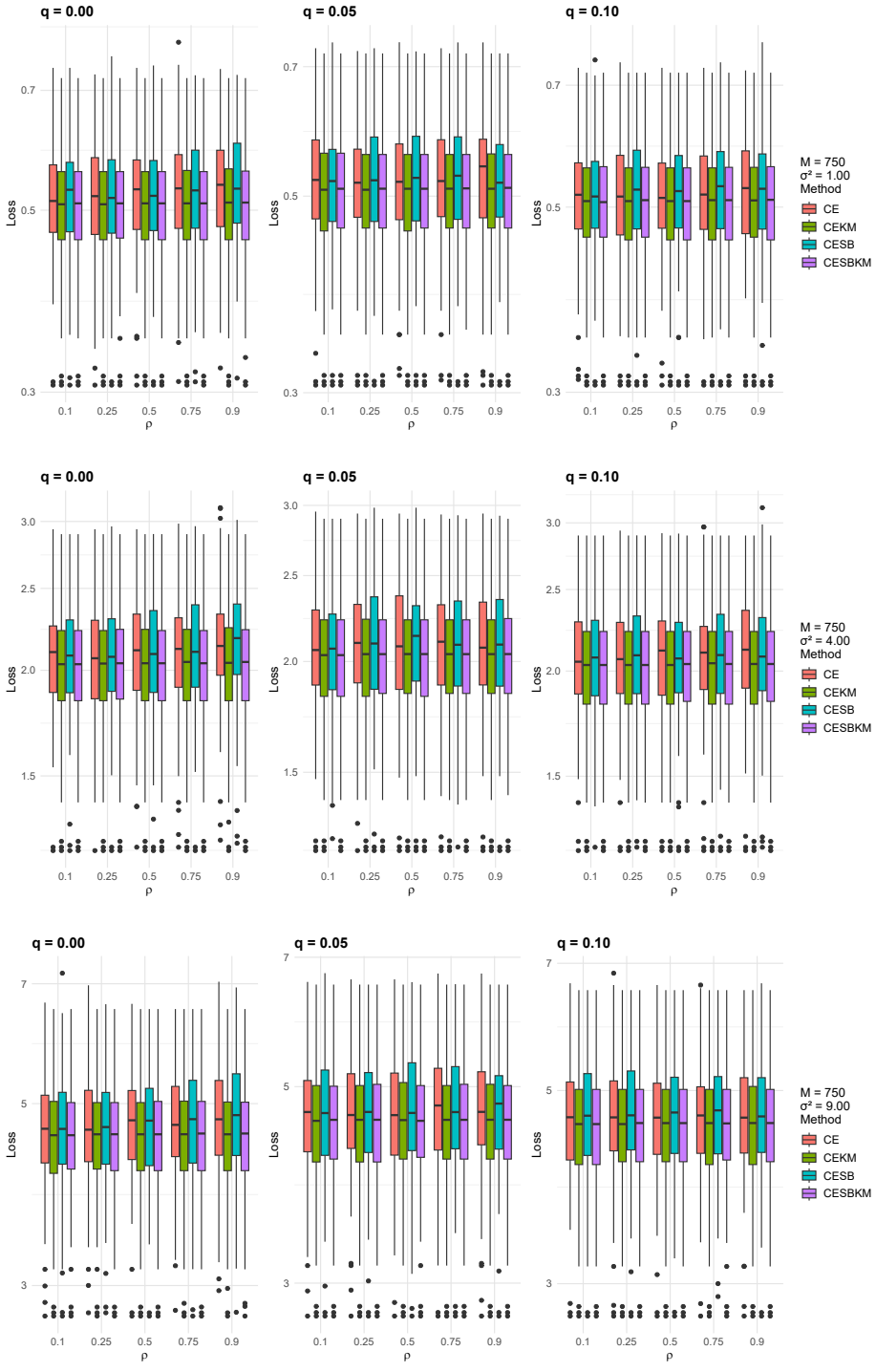


Figure 6.48 – Loss values (y-axis in log-scale) for Noise model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

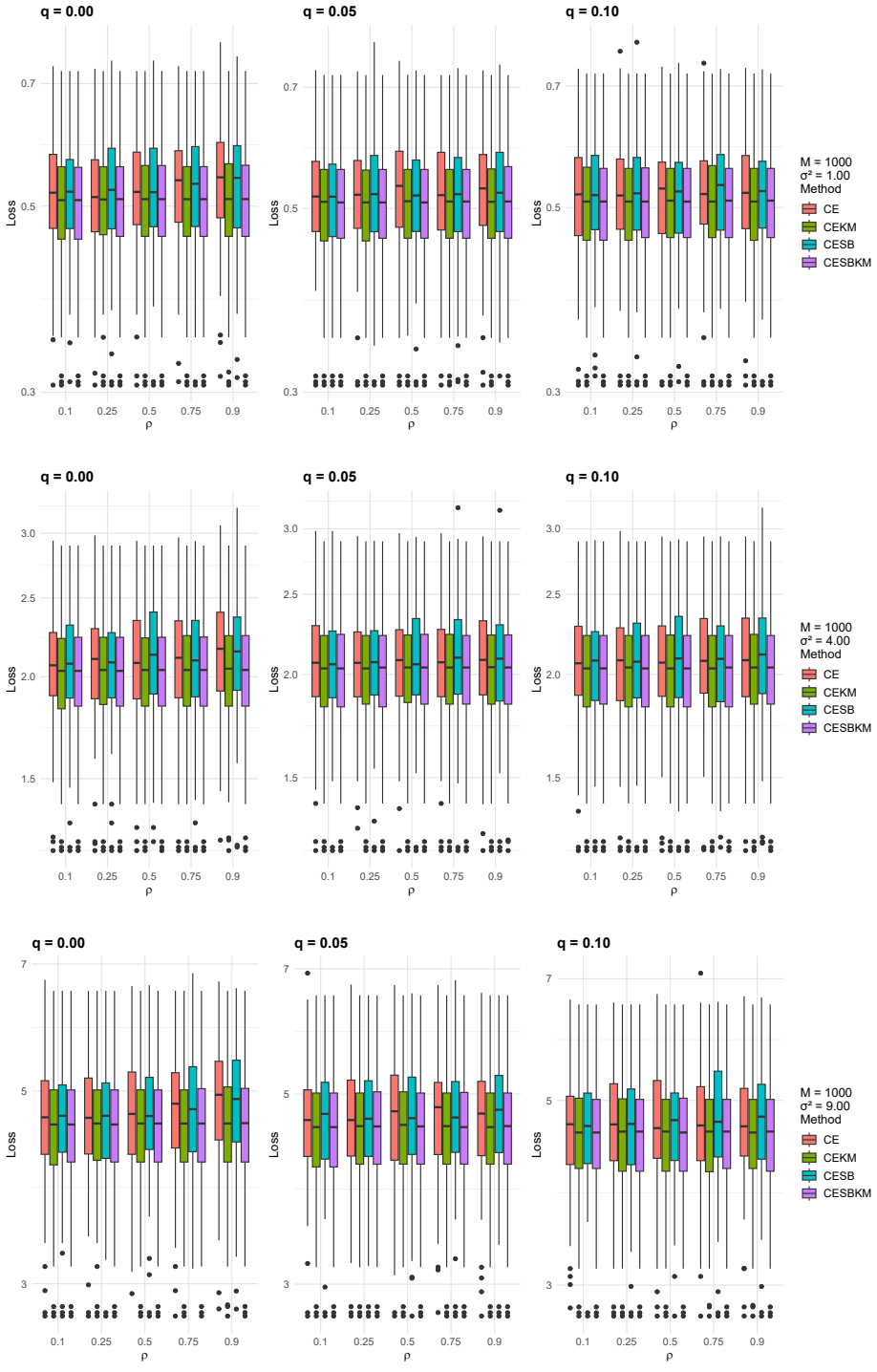


Figure 6.49 – Loss values (y-axis in log-scale) for Noise model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

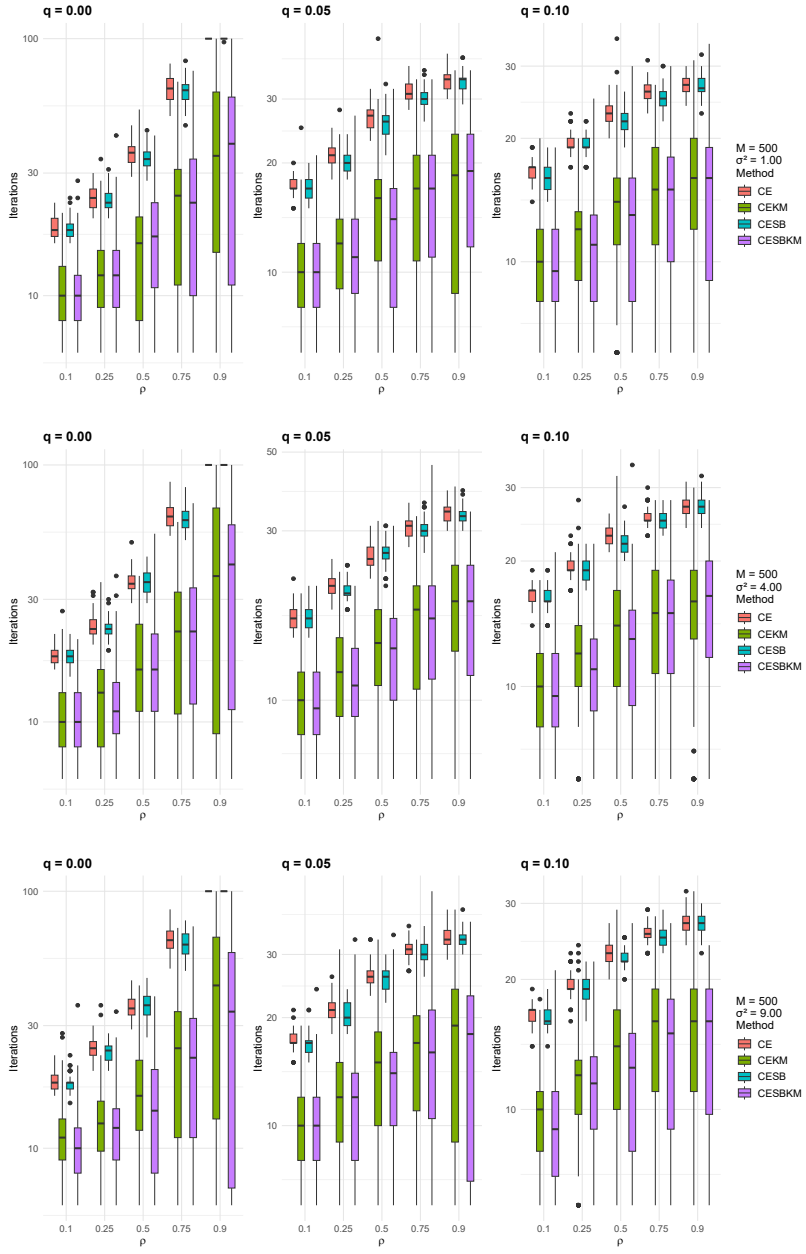


Figure 6.50 – Number of Iterations (y-axis in log-scale) for Noise model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

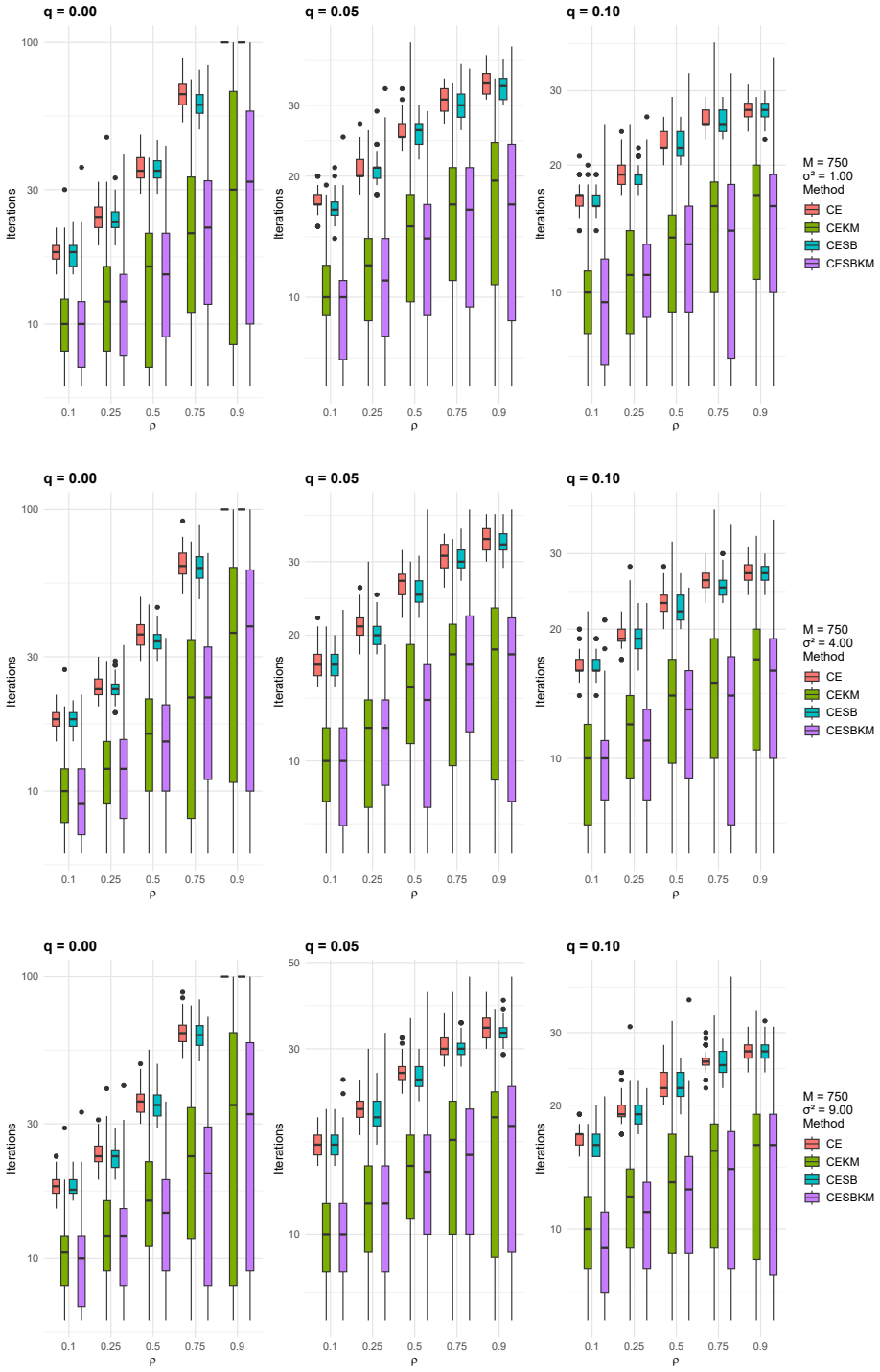


Figure 6.51 – Number of Iterations (y-axis in log-scale) for Noise model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

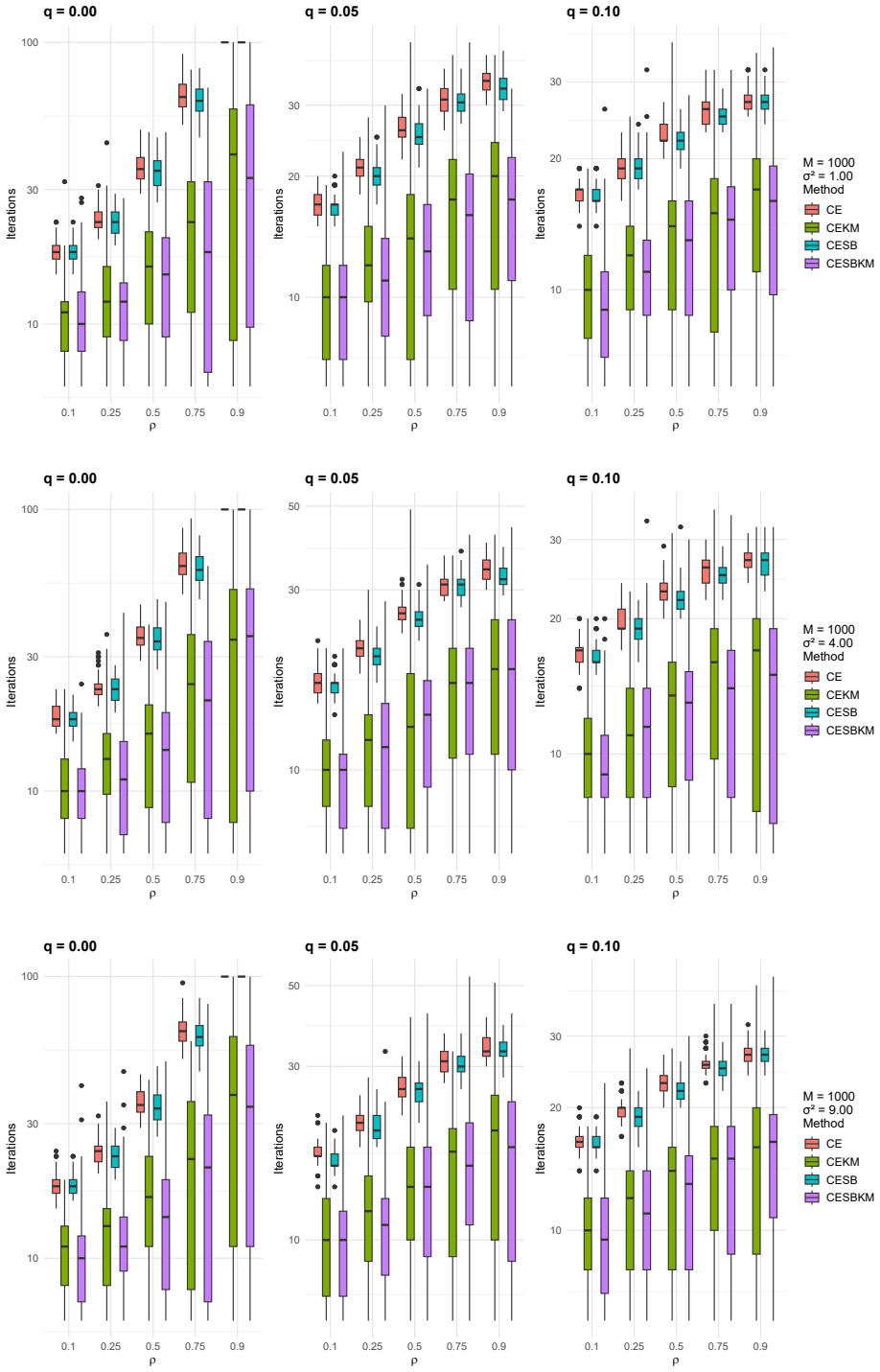


Figure 6.52 – Number of Iterations (y-axis in log-scale) for Noise model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

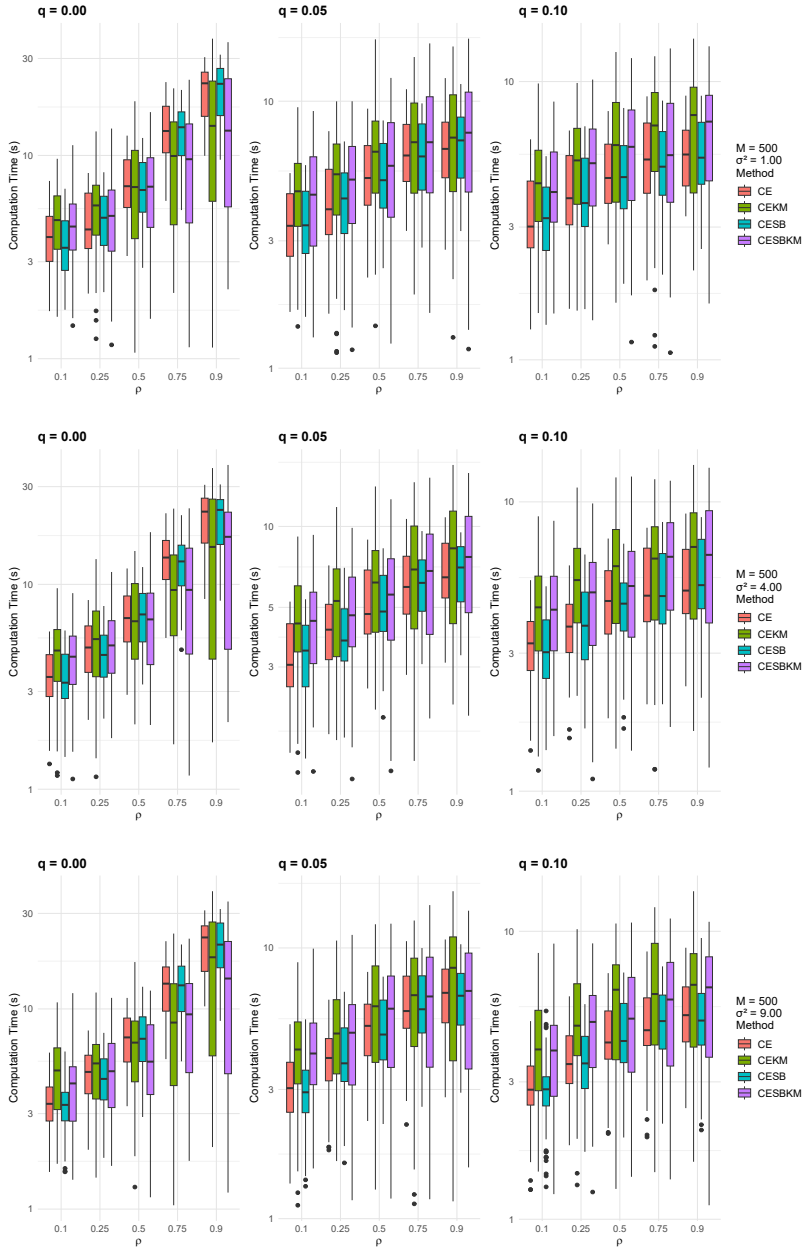


Figure 6.53 – Computation Time (y-axis in log-scale) for Noise model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=500)

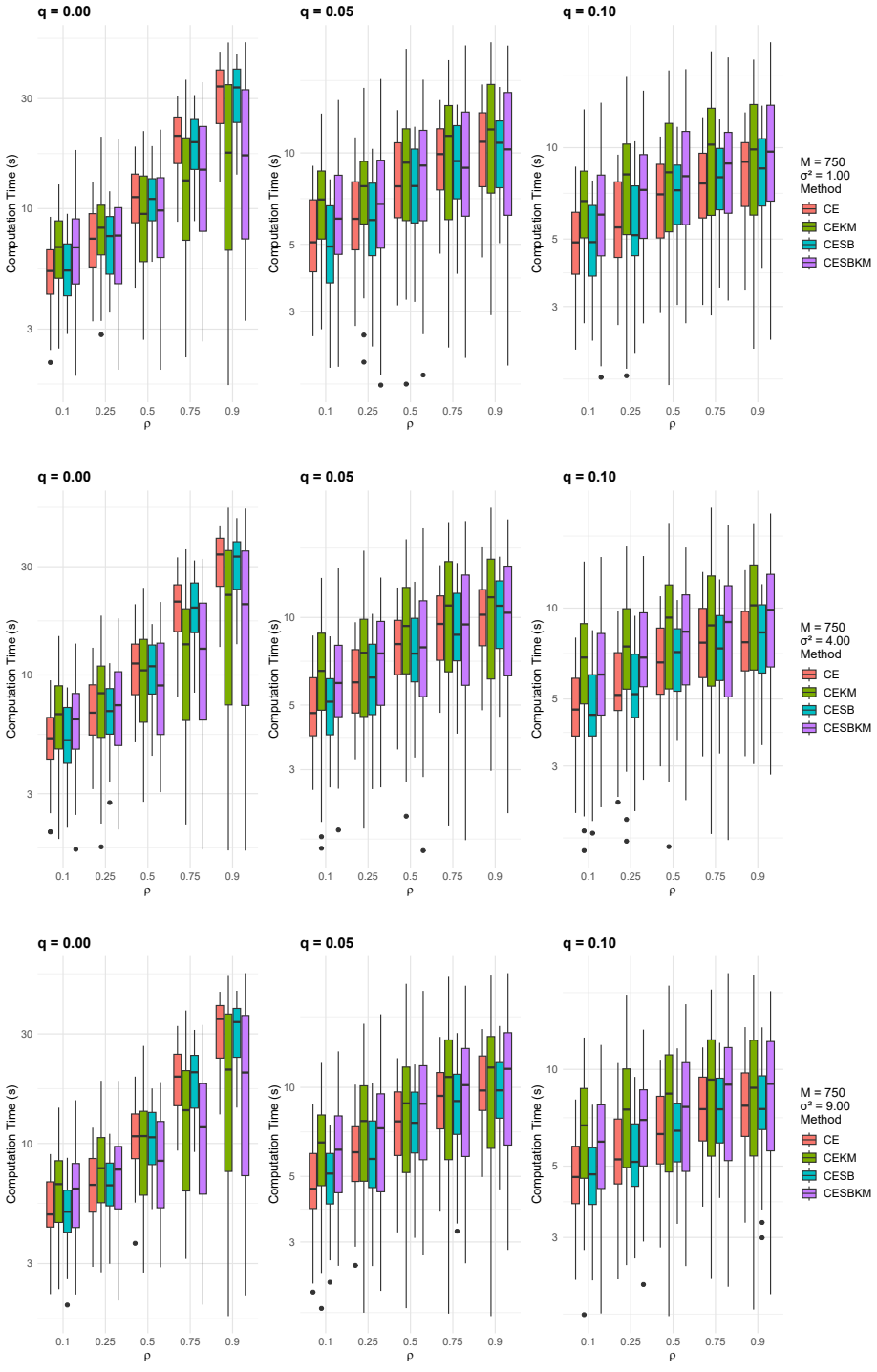


Figure 6.54 – Computation Time (y-axis in log-scale) for Noise model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=750)

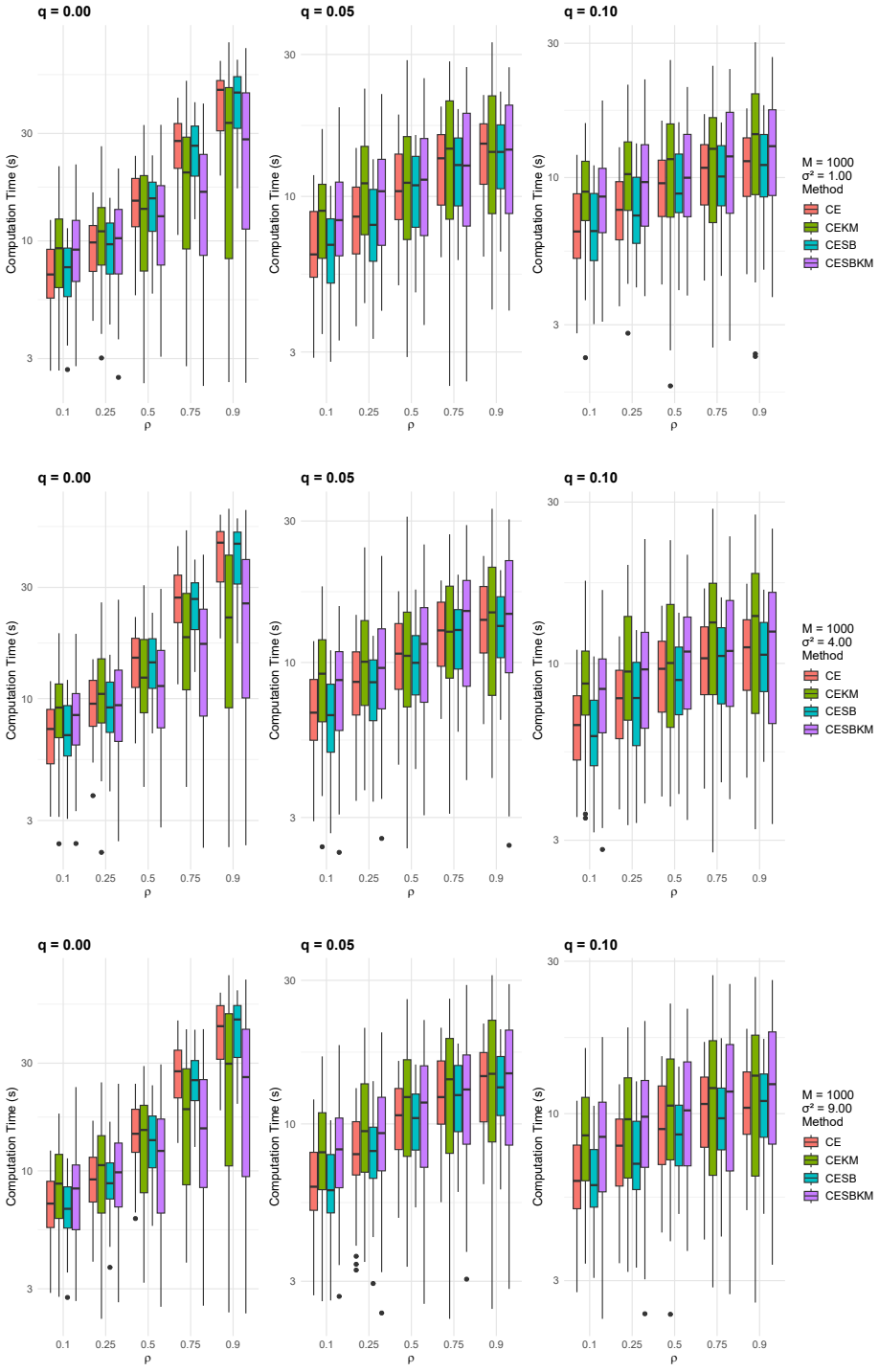


Figure 6.55 – Computation Time (y-axis in log-scale) for Noise model with SBC on the first element of the cluster vector: Comparison of CESBKM variants across noise levels $\sigma^2 = \{1, 4, 9\}$, with the number of Monte Carlo samples (M=1000)

6.3 Appendix of Chapter 4

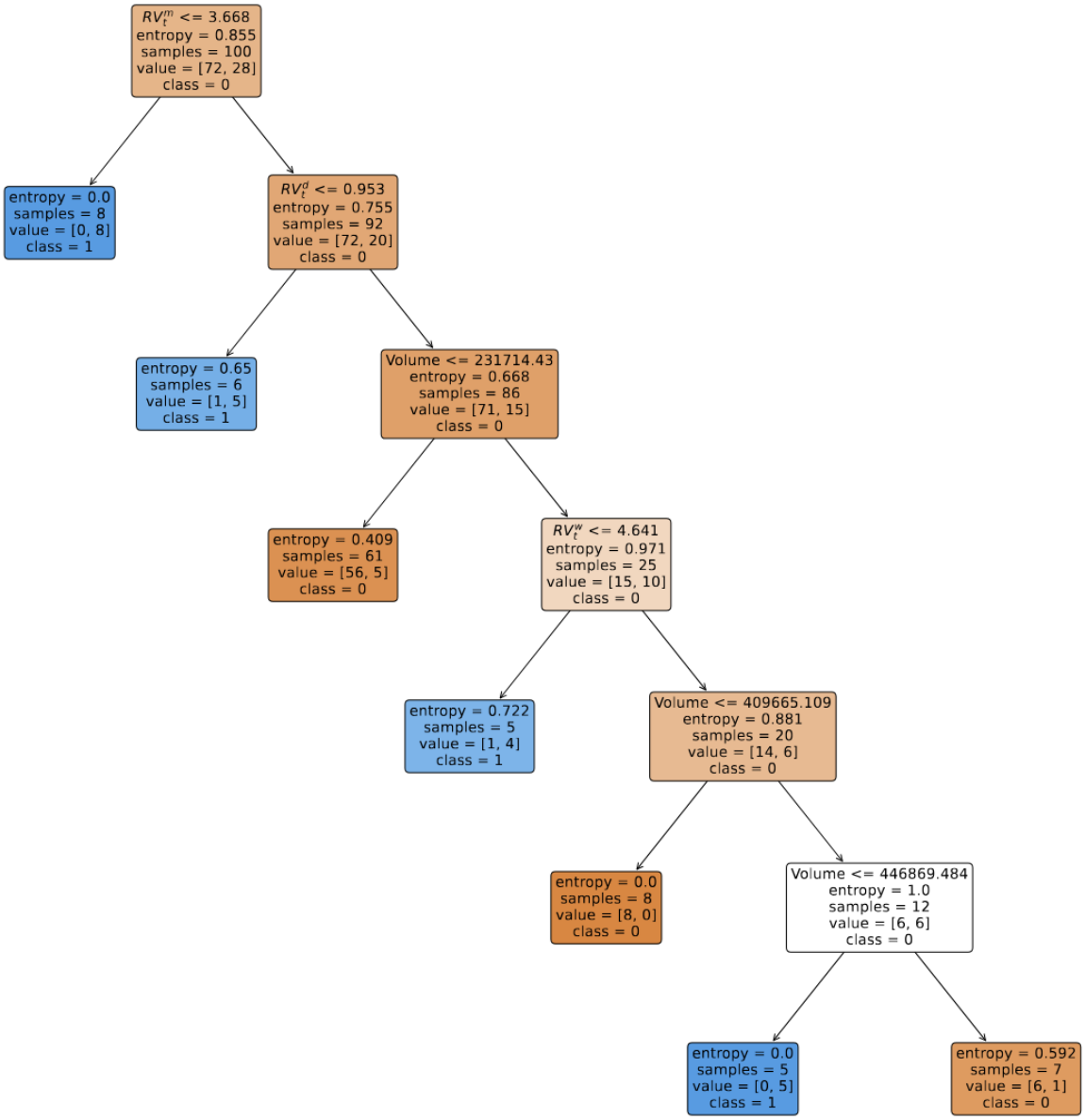


Figure 6.56 – Regime classification for Bitcoin's realized volatility.

Dep. Variable:	$RV_{t+1}^{(d)}$	**R-squared:**	0.058
Model:	OLS	**Adj. R-squared:**	0.028
Method:	Least Squares	**F-statistic:**	1.960
Date:	Mon, 07 Oct 2024	**Prob (F-statistic):**	0.125
Time:	04:10:43	**Log-Likelihood:**	-385.73
No. Observations:	100	**AIC:**	779.5
Df Residuals:	96	**BIC:**	789.9
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	8.0654	3.159	2.553	0.012	1.794	14.337
$RV_t^{(d)}$	0.0622	0.124	0.502	0.617	-0.184	0.308
$RV_t^{(w)}$	0.6170	0.364	1.694	0.094	-0.106	1.340
$RV_t^{(m)}$	-0.8160	0.574	-1.421	0.159	-1.956	0.324

Omnibus:	159.380	**Durbin-Watson:**	1.945
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	8692.031
Skew:	5.920	**Prob(JB):**	0.00
Kurtosis:	47.112	**Cond. No.**	45.2

Table 6.6 – OLS Regression Results for a Simple HAR model

BIBLIOGRAPHY

D. Y. Aharon and M. Qadan. Bitcoin and the day-of-the-week effect. *Finance Research Letters*, 31, Dec. 2019. ISSN 1544-6123. doi: 10.1016/j.frl.2018.12.004. URL https://www.sciencedirect.com/science/article/pii/S1544612317307894. (Cited on p. 39)

M. Ahmed, R. Seraj, and S. M. S. Islam. The k-means Algorithm: A Comprehensive Survey and Performance Evaluation. *Electronics*, 9(8):1295, Aug. 2020. ISSN 2079-9292. doi: 10.3390/electronics9081295. URL https://www.mdpi.com/2079-9292/9/8/1295. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute. (Cited on p. 71)

T. Andersen and T. Bollerslev. Answering the Skeptics: Yes, Standard Volatility Models Do Provide Accurate Forecasts. *International Economic Review*, 39(4):885–905, 1998. URL https://econpapers.repec.org/article/ieriecrev/v_3a39_3ay_3a1998_3ai_3a4_3ap_3a885-905.htm. Publisher: Department of Economics, University of Pennsylvania and Osaka University Institute of Social and Economic Research Association. (Cited on p. 8)

T. G. Andersen, T. Bollerslev, and F. X. Diebold. Roughing It Up: Including Jump Components in the Measurement, Modeling, and Forecasting of Return Volatility. *The Review of Economics and Statistics*, 89(4):701–720, 2007. URL https://ideas.repec.org/a/tpr/restat/v89y2007i4p701-720.html. Publisher: MIT Press. (Cited on p. 19, 95)

A. Arratia and A. X. López-Barrantes. Do Google Trends forecast bitcoins? Stylized facts and statistical evidence. *Journal of Banking and Financial Technology*, Mar. 2021. ISSN 2524-7956, 2524-7964. doi: 10.1007/s42786-021-00027-4. URL http://link.springer.com/10.1007/s42786-021-00027-4. (Cited on p. 39)

H. Asgharian, C. Christiansen, and A. J. Hou. The effect of uncertainty on stock market volatility and correlation. *Journal of Banking & Finance*, 154:106929, Sept. 2023. ISSN 0378-4266. doi: 10.1016/j.jbankfin.2023.106929. URL https://www.sciencedirect.com/science/article/pii/S0378426623001097. (Cited on p. 12)

N. Aslanidis, A. F. Bariviera, and Ó . G. López. The link between cryptocurrencies and Google Trends attention. *Finance Research Letters*, 47:102654, June 2022. ISSN 1544-6123. doi: 10.1016/j.frl.2021.102654. URL https://www.sciencedirect.com/science/article/pii/S1544612321005833. (Cited on p. 39)

F. Audrino and F. Corsi. Modeling tick-by-tick realized correlations. *Computational Statistics & Data Analysis*, 54(11):2372–2382, Nov. 2010. ISSN 01679473. doi: 10.1016/j.csda.2009.09.033. URL https://linkinghub.elsevier.com/retrieve/pii/S0167947309003636. (Cited on p. 19, 95)

D. Bakas, G. Magkonis, and E. Y. Oh. What drives volatility in Bitcoin market? *Finance Research Letters*, 50:103237, Dec. 2022. ISSN 1544-6123. doi: 10.1016/j.frl.2022.103237. URL https://www.sciencedirect.com/science/article/pii/S1544612322004378. (Cited on p. 39)

O. E. Barndorff-Nielsen and N. Shephard. Econometric Analysis of Realized Volatility and Its Use in Estimating Stochastic Volatility Models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 64(2):253–280, 2002. ISSN 1369-7412. URL https://www.jstor.org.ezproxy.is.cuni.cz/stable/3088799. Publisher: [Royal Statistical Society, Wiley]. (Cited on p. 8)

L. Bauwens and G. Sucarrat. General to specific modelling of exchange rate volatility: A forecast evaluation. *International Journal of Forecasting*, 26:885–907, 2010. (Cited on p. 22)

L. Bauwens, C. Hafner, and S. Laurent. Volatility Models. In *Handbook of Volatility Models and Their Applications*, pages 1–45. John Wiley & Sons, Ltd, 2012. ISBN 978-1-118-27203-9. doi: 10.1002/9781118272039.ch1. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118272039.ch1. Section: One _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118272039.ch1. (Cited on p. 7, 8, 9, 16, 121)

L. Ø. Bergsli, A. F. Lind, P. Molnár, and M. Polasik. Forecasting volatility of Bitcoin. *Research in International Business and Finance*, 59:101540, Jan. 2022. ISSN 0275-5319. doi: 10.1016/j.ribaf.2021.101540. URL https://www.sciencedirect.com/science/article/pii/S0275531921001616. (Cited on p. 9, 10, 33, 39, 89, 119)

M. Bernardi and L. Catania. The Model Confidence Set package for R. *CEIS Research Paper*, Nov. 2015. URL https://ideas.repec.org//p/rtv/ceisrp/362.html. Number: 362 Publisher: Tor Vergata University, CEIS. (Cited on p. 37, 42)

T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31 (3):307–327, Apr. 1986. ISSN 0304-4076. doi: 10.1016/0304-4076(86)90063-1. URL https://www.sciencedirect.com/science/article/pii/0304407686900631. (Cited on p. 8, 11, 20)

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug. 1996. ISSN 1573-0565. doi: 10.1007/BF00058655. URL https://doi.org/10.1007/BF00058655. (Cited on p. 18, 35, 94)

L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Oct. 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324. (Cited on p. 36)

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Taylor & Francis, Jan. 1984. ISBN 978-0-412-04841-8. (Cited on p. 15, 19, 95, 105, 106, 108, 110, 111)

P. J. Brockwell and R. A. Davis. Estimation for ARMA Models. In P. J. Brockwell and R. A. Davis, editors, *Time Series: Theory and Methods*, Springer Series in Statistics, pages 238–272. Springer, New York, NY, 1991. ISBN 978-1-4419-0320-4. doi: 10.1007/978-1-4419-0320-4_8. URL https://doi.org/10.1007/978-1-4419-0320-4_8. (Cited on p. 29)

C. Brooks. *Introductory Econometrics for Finance*. Cambridge University Press, Cambridge, 2 edition, 2008. doi: 10.1017/CBO9780511841644. URL https://www.cambridge.org/core/books/introductory-econometrics-for-finance/4F3AB9473A63F11982D6902D813BC521. (Cited on p. 34)

H. Bystrom and D. Krygier. What Drives Bitcoin Volatility?, July 2018. URL https://papers.ssrn.com/abstract=3223368. (Cited on p. 39)

C. W. S. Chen, M. K. P. So, and F.-C. Liu. A review of threshold time series models in finance. *Statistics and Its Interface*, 4(2):167–181, 2011. ISSN 1938-7989. URL https://www.webofscience.com/wos/woscc/summary/b4546f70-70bf-43e0-943c-dac6e336bc12-00fdb4b2/date-descending/1. Place: Somerville Publisher: Int Press Boston, Inc WOS:000293847000013. (Cited on p. 93)

R. Chen. Threshold Variable Selection in Open-Loop Threshold Autoregressive Models. *Journal of Time Series Analysis*, 16(5):461–481, 1995. ISSN 1467-9892. doi: 10.1111/j.1467-9892.1995.tb00247.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9892.1995.tb00247.x. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9892.1995.tb00247.x. (Cited on p. 9, 14, 15, 79, 94, 96, 97, 98, 101, 103, 109, 112)

W. Chen, H. Xu, L. Jia, and Y. Gao. Machine learning model for Bitcoin exchange rate prediction using economic and technology determinants. *International Journal of Forecasting*, 37(1):28–43, Jan. 2021. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2020.02.008. URL https://www.sciencedirect.com/science/article/pii/S0169207020300431. (Cited on p. 34)

A. Clements and D. P. A. Preve. A Practical Guide to harnessing the HAR volatility model. *Journal of Banking & Finance*, 133:106285, Dec. 2021. ISSN 0378-4266. doi: 10.1016/j.jbankfin.2021.106285. URL https://www.sciencedirect.com/science/article/pii/S0378426621002417. (Cited on p. 46)

R. Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, 1(2):223–236, 2001. (Cited on p. 7, 9, 14, 43, 138)

R. Cont. Volatility Clustering in Financial Markets: Empirical Facts and Agent-Based Models. In G. Teyssière and A. P. Kirman, editors, *Long Memory in Economics*, pages 289–309. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-22694-9. doi: 10.1007/978-3-540-34625-8_10. URL http://link.springer.com/10.1007/978-3-540-34625-8_10. (Cited on p. 14, 87)

V. Corradi, W. Distaso, and A. Mele. Macroeconomic determinants of stock volatility and volatility premiums. *Journal of Monetary Economics*, 60(2):203–220, Mar. 2013. ISSN 0304-3932. doi: 10.1016/j.jmoneco.2012.10.019. URL https://www.sciencedirect.com/science/article/pii/S0304393212001341. (Cited on p. 12)

F. Corsi. A Simple Long Memory Model of Realized Volatility. *Journal of Financial Econometrics*, 7: 174–196, Feb. 2009. doi: 10.1093/jjfinec/nbp001. (Cited on p. 8, 13, 14, 19, 46, 87, 95, 118)

F. Corsi, D. Pirino, and R. Renò. Threshold Bipower Variation and the Impact of Jumps on Volatility Forecasting. LEM Papers Series, Laboratory of Economics and Management (LEM), Sant'Anna School of Advanced Studies, Pisa, Italy, July 2010. URL https://econpapers.repec.org/paper/ssalemwps/2010_2f11.htm. (Cited on p. 19, 95)

F. Corsi, F. Audrino, and R. Renò. HAR Modeling for Realized Volatility Forecasting. In *Handbook of Volatility Models and Their Applications*, pages 363–382. John Wiley & Sons, Ltd, 2012. ISBN 978-1-118-27203-9. doi: 10.1002/9781118272039.ch15. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118272039.ch15. Section: Fifteen _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118272039.ch15. (Cited on p. 13, 14, 19, 95)

A. Costa, O. D. Jones, and D. Kroese. Convergence properties of the cross-entropy method for discrete optimization. *Operations Research Letters*, 35(5):573–580, Sept. 2007. ISSN 01676377. doi: 10.1016/j.orl.2006.11.005. URL https://linkinghub.elsevier.com/retrieve/pii/S0167637706001313. (Cited on p. 71)

P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A Tutorial on the Cross-Entropy Method. *Annals of Operations Research*, 134(1):19–67, Feb. 2005. ISSN 0254-5330, 1572-9338. doi: 10.1007/s10479-005-5724-z. URL http://link.springer.com/10.1007/s10479-005-5724-z. (Cited on p. 53, 55, 58, 63, 64, 98, 100, 101, 114)

A. Dean, D. Voss, and D. Draguljić. *Design and Analysis of Experiments*. Springer Texts in Statistics. Springer International Publishing, Cham, 2017. ISBN 978-3-319-52248-7 978-3-319-52250-0. doi: 10.1007/978-3-319-52250-0. URL http://link.springer.com/10.1007/978-3-319-52250-0. (Cited on p. 82)

W. S. DeSarbo and W. L. Cron. A maximum likelihood methodology for clusterwise linear regression. *Journal of Classification*, 5(2):249–282, Sept. 1988. ISSN 1432-1343. doi: 10.1007/BF01897167. URL https://doi.org/10.1007/BF01897167. (Cited on p. 48)

D. Dickey and W. Fuller. Distribution of the Estimators for Autoregressive Time Series With a Unit Root. *JASA. Journal of the American Statistical Association*, 74, June 1979. doi: 10.2307/2286348. (Cited on p. 34)

G. Elliott and A. Timmermann. Economic Forecasting. *Journal of Economic Literature*, 46(1):3–56, Feb. 2008. ISSN 0022-0515. doi: 10.1257/jel.46.1.3. URL https://pubs.aeaweb.org/doi/10.1257/jel.46.1.3. (Cited on p. 35)

R. F. Engle. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50(4):987–1007, 1982. ISSN 0012-9682. doi: 10.2307/1912773. URL https://www.jstor.org/stable/1912773. Publisher: [Wiley, Econometric Society]. (Cited on p. 8)

A. Escribano and G. Sucarrat. Equation-by-equation estimation of multivariate periodic electricity price volatility. *Energy Economics*, 74:287–298, Aug. 2018. ISSN 0140-9883. doi: 10.1016/j.eneco.2018.05.

017. URL https://www.sciencedirect.com/science/article/pii/S0140988318301841. (Cited on p. 36)

A. E. Ezugwu, A. M. Ikotun, O. O. Oyelade, L. Abualigah, J. O. Agushaka, C. I. Eke, and A. A. Akinyelu. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, Apr. 2022. ISSN 0952-1976. doi: 10.1016/j.engappai.2022.104743. URL https://www.sciencedirect.com/science/article/pii/S095219762200046X. (Cited on p. 45)

C. Francq and G. Sucarrat. An Exponential Chi-Squared QMLE for Log-GARCH Models Via the ARMA Representation. *MPRA Paper*, Oct. 2013. URL https://ideas.repec.org//p/pra/mprapa/51783.html. Number: 51783 Publisher: University Library of Munich, Germany. (Cited on p. 28)

C. Francq and G. Sucarrat. An equation-by-equation estimator of a multivariate log-GARCH-X model of financial returns. *Journal of Multivariate Analysis*, 153:16–32, Jan. 2017. ISSN 0047259X. doi: 10.1016/j.jmva.2016.09.010. URL https://linkinghub.elsevier.com/retrieve/pii/S0047259X16300938. (Cited on p. 17, 21, 22)

C. Francq and L. Q. Thieu. QML INFERENCE FOR VOLATILITY MODELS WITH COVARIATES. *Econometric Theory*, 35(1):37–72, Feb. 2019. ISSN 0266-4666, 1469-4360. doi: 10.1017/S0266466617000512. URL https://www.cambridge.org/core/product/identifier/S0266466617000512/type/journal_article. (Cited on p. 11, 12, 20, 21, 24)

C. Francq and J.-M. Zakoian. *GARCH Models: Structure, Statistical Inference and Financial Applications*. John Wiley & Sons, June 2011. ISBN 978-1-119-95739-3. Google-Books-ID: hwR1aWSg9PUC. (Cited on p. 10)

C. Francq and J.-M. Zakoïan. Estimating ARCH Models by Least Squares. In *GARCH Models*, pages 127–140. John Wiley & Sons, Ltd, Chichester, UK, July 2010. ISBN 978-0-470-67005-7 978-0-470-68391-0. doi: 10.1002/9780470670057.ch6. URL https://onlinelibrary.wiley.com/doi/10.1002/9780470670057.ch6. (Cited on p. 26)

C. Francq, O. Wintenberger, and J.-M. Zakoïan. GARCH models without positivity constraints: Exponential or log GARCH? *Journal of Econometrics*, 177(1):34–46, Nov. 2013. ISSN 0304-4076. doi: 10.1016/j.jeconom.2013.05.004. URL https://www.sciencedirect.com/science/article/pii/S0304407613001267. (Cited on p. 12, 21, 28)

P. H. Franses and D. v. Dijk. *Nonlinear time series models in empirical finance*. Cambridge University Press, Cambridge, UK ; New York, 2000. ISBN 978-0-521-77041-5 978-0-521-77965-4. (Cited on p. 10)

J. Friedman, R. Tibshirani, and T. Hastie. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. doi: 10.18637/jss.v033.i01. (Cited on p. 27, 31)

G. Gardner, A. C. Harvey, and G. D. A. Phillips. An Algorithm for Exact Maximum Likelihood Estimation of Autoregressive–Moving Average Models by Means of Kaiman Filtering. *Journal of the Royal Statistical Society Series C*, 29(3):311–322, 1980. URL https://ideas.repec.org//a/bla/jorssc/v29y1980i3p311-322.html. Publisher: Royal Statistical Society. (Cited on p. 27, 29)

J. Geweke. Commet. *Econometric Reviews*, 5(1):57–61, Jan. 1986. ISSN 0747-4938. doi: 10.1080/07474938608800097. URL https://doi.org/10.1080/07474938608800097. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/07474938608800097. (Cited on p. 25)

L. R. Glosten, R. Jagannathan, and D. E. Runkle. On the Relation between the Expected Value and the Volatility of the Nominal Excess Return on Stocks. *The Journal of Finance*, 48(5):1779–1801, Dec. 1993. ISSN 00221082. doi: 10.1111/j.1540-6261.1993.tb05128.x. URL https://onlinelibrary.wiley.com/doi/10.1111/j.1540-6261.1993.tb05128.x. (Cited on p. 12, 25)

A. Goldsztejn, C. Jermann, V. Ruiz de Angulo, and C. Torras. Variable symmetry breaking in numerical constraint problems. *Artificial Intelligence*, 229:105–125, Dec. 2015. ISSN 0004-3702. doi: 10.1016/j.artint.2015.08.006. URL https://www.sciencedirect.com/science/article/pii/S0004370215001216. (Cited on p. 59)

T. Guo, A. Bifet, and N. Antulov-Fantulin. Bitcoin Volatility Forecasting with a Glimpse into Buy and Sell Orders. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 989–994, Nov. 2018. doi: 10.1109/ICDM.2018.00123. URL http://arxiv.org/abs/1802.04065. arXiv:1802.04065 [cs, stat]. (Cited on p. 10)

J. D. Hamilton. A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle. *Econometrica*, 57(2):357–384, 1989. ISSN 0012-9682. doi: 10.2307/1912559. URL https://www.jstor.org/stable/1912559. Publisher: [Wiley, Econometric Society]. (Cited on p. 93)

J. D. Hamilton. *Time series analysis*. Princeton university press, 1994. (Cited on p. 20, 29)

P. R. Hansen, A. Lunde, and J. M. Nason. The Model Confidence Set. *Econometrica*, 79(2):453–497, 2011. ISSN 0012-9682. URL https://www.jstor.org/stable/41057463. Publisher: [Wiley, Econometric Society]. (Cited on p. 37, 42, 43, 137)

J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979. (Cited on p. 51)

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media, 2001. ISBN 978-0-387-95284-0. Google-Books-ID: VRzITwgNV2UC. (Cited on p. 35, 105, 108)

R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, Lexington, Ky., 2nd edition edition, May 2018. ISBN 978-0-9875071-1-2. (Cited on p. 42, 137)

A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622:178–210, Apr. 2023. ISSN 0020-0255. doi: 10.1016/j.ins.2022.11.139. URL https://www.sciencedirect.com/science/article/pii/S0020025522014633. (Cited on p. 49)

X. Jin and J. Han. K-Means Clustering. In C. Sammut and G. I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 695–697. Springer US, Boston, MA, 2017. ISBN 978-1-4899-7687-1. doi: 10.1007/978-1-4899-7687-1_431. URL https://doi.org/10.1007/978-1-4899-7687-1_431. (Cited on p. 52)

D. P. Kroese, R. Y. Rubinstein, and T. Taimre. Application of the cross-entropy method to clustering and vector quantization. *Journal of Global Optimization*, 37(1):137–157, Jan. 2007. ISSN 1573-2916. doi: 10.1007/s10898-006-9041-0. URL https://doi.org/10.1007/s10898-006-9041-0. (Cited on p. 45, 49, 50, 59, 61, 71)

H. R. Kunsch. The Jackknife and the Bootstrap for General Stationary Observations. *The Annals of Statistics*, 17(3):1217–1241, Sept. 1989. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1176347265. URL https://projecteuclid.org/journals/annals-of-statistics/volume-17/issue-3/The-Jackknife-and-the-Bootstrap-for-General-Stationary-Observations/10.1214/aos/1176347265.full. Publisher: Institute of Mathematical Statistics. (Cited on p. 36)

M. B. Kursa and W. R. Rudnicki. Feature Selection with the Boruta Package. *Journal of Statistical Software*, 36:1–13, Sept. 2010. ISSN 1548-7660. doi: 10.18637/jss.v036.i11. URL https://doi.org/10.18637/jss.v036.i11. (Cited on p. 27, 32)

D. Kwiatkowski, P. C. B. Phillips, P. Schmidt, and Y. Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1):159–178, Oct. 1992. ISSN 0304-4076. doi: 10.1016/0304-4076(92)90104-Y. URL https://www.sciencedirect.com/science/article/pii/030440769290104Y. (Cited on p. 34)

D. Lieber. *Rare Events Estimation Via Cross-Entropy and Importance Sampling*. PhD thesis, The Technion - Israel Institute of Technology, Haifa, 1998. URL https://drive.google.com/file/d/1ojHMJOfj6iGHc5wUJY_4-_MCe4zho0qQ/view?usp=drive_link&usp=embed_facebook. (Cited on p. 46, 76)

J. Lintner. The Valuation of Risk Assets and the Selection of Risky Investments in Stock Portfolios and Capital Budgets. *The Review of Economics and Statistics*, 47(1):13–37, 1965. ISSN 0034-6535. doi: 10.2307/1924119. URL http://www.jstor.org/stable/1924119. (Cited on p. 7)

S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, Mar. 1982. ISSN 1557-9654. doi: 10.1109/TIT.1982.1056489. Conference Name: IEEE Transactions on Information Theory. (Cited on p. 45, 47)

Q. Long, A. Bagirov, S. Taheri, N. Sultanova, and X. Wu. Methods and Applications of Clusterwise Linear Regression: A Survey and Comparison. *ACM Transactions on Knowledge Discovery from Data*,

17(3):1–54, June 2023. ISSN 1556-4681, 1556-4a72X. doi: 10.1145/3550074. URL https://dl.acm.org/doi/10.1145/3550074. (Cited on p. 45, 51)

M. Á. López-Cabarcos, A. M. Pérez-Pico, J. Piñeiro-Chousa, and A. Šević. Bitcoin volatility, stock market and investor sentiment. Are they connected? *Finance Research Letters*, 38:101399, Jan. 2021. ISSN 15446123. doi: 10.1016/j.frl.2019.101399. URL https://linkinghub.elsevier.com/retrieve/pii/S1544612319309274. (Cited on p. 39)

J. Ma, J. Feng, J. Chen, and J. Zhang. Volatility Spillover from Carbon Prices to Stock Prices: Evidence from China's Carbon Emission Trading Markets. *JRFM*, 17(3):1–24, 2024. URL https://ideas.repec.org//a/gam/jjrfmx/v17y2024i3p123-d1358823.html. Publisher: MDPI. (Cited on p. 12)

S. McHardy. python-binance: Binance Exchange API python implementation for automated trading, 2023. URL https://github.com/sammchardy/python-binance. Accessed: 2024-05-11. (Cited on p. 33, 119)

A. Milhoj. A Conditional Variance Model for Daily Deviations of an Exchange Rate. *Journal of Business & Economic Statistics*, 5(1):99–103, 1987. URL https://ideas.repec.org//a/bes/jnlbes/v5y1987i1p99-103.html. Publisher: American Statistical Association. (Cited on p. 25)

A. Miller. *Subset Selection in Regression*. Chapman and Hall/CRC, New York, 2 edition, Apr. 2002. ISBN 978-0-429-11918-7. doi: 10.1201/9781420035933. (Cited on p. 30)

D. B. Nelson. Conditional Heteroskedasticity in Asset Returns: A New Approach. *Econometrica*, 59 (2):347–370, 1991. ISSN 0012-9682. doi: 10.2307/2938260. URL http://www.jstor.org/stable/2938260. Publisher: [Wiley, Econometric Society]. (Cited on p. 11, 20)

H. B. Nielsen and A. Rahbek. Penalized quasi-likelihood estimation and model selection with parameters on the boundary of the parameter space. *The Econometrics Journal*, 27(1):107–125, Jan. 2024. ISSN 1368-4221. doi: 10.1093/ectj/utad022. URL https://doi.org/10.1093/ectj/utad022. (Cited on p. 16, 21, 29)

S. G. Pantula. Comment. *Econometric Reviews*, 5(1):71–74, Jan. 1986. ISSN 0747-4938, 1532-4168. doi: 10.1080/07474938608800099. URL http://www.tandfonline.com/doi/abs/10.1080/07474938608800099. (Cited on p. 8, 21, 25)

S. Park and O. Linton. Realized Volatility: Theory and Applications. In *Handbook of Volatility Models and Their Applications*, pages 317–345. John Wiley & Sons, Ltd, 2012. ISBN 978-1-118-27203-9. doi: 10.1002/9781118272039.ch13. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118272039.ch13. Section: Thirteen _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118272039.ch13. (Cited on p. 7, 9, 12)

A. J. Patton. Volatility forecast comparison using imperfect volatility proxies. *Journal of Econometrics*, 160(1):246–256, Jan. 2011. ISSN 0304-4076. doi: 10.1016/j.jeconom.2010.03.034. URL https://www.sciencedirect.com/science/article/pii/S030440761000076X. (Cited on p. 37)

R. S. Pedersen and A. Rahbek. TESTING GARCH-X TYPE MODELS. *Econometric Theory*, 35(05): 1012–1047, Oct. 2019. ISSN 0266-4666, 1469-4360. doi: 10.1017/S026646661800035X. URL https://www.cambridge.org/core/product/identifier/S026646661800035X/type/journal_article. (Cited on p. 11, 24)

S. Potter. Nonlinear Time Series Modelling: An Introduction. *Journal of Economic Surveys*, 13(5):505–528, 1999. ISSN 1467-6419. doi: 10.1111/1467-6419.00096. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-6419.00096. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-6419.00096. (Cited on p. 93)

F. Pretis, J. Reade, and G. Sucarrat. Automated general-to-specific (gets) regression modeling and indicator saturation for outliers and structural breaks. *Journal of Statistical Software*, 86:1–44, 2018. (Cited on p. 16, 22)

Y. Qiu, X. Zhang, T. Xie, and S. Zhao. Versatile HAR model for realized volatility: A least square model averaging perspective. *Journal of Management Science and Engineering*, 4(1):55–73, Mar. 2019. ISSN 2096-2320. doi: 10.1016/j.jmse.2019.03.003. URL https://www.sciencedirect.com/science/article/pii/S2096232019300046. (Cited on p. 46)

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2023. URL https://www.R-project.org/. (Cited on p. 27, 34)

R. Rubinstein. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology And Computing In Applied Probability*, 1(2):127–190, Sept. 1999. ISSN 1573-7713. doi: 10.1023/A:1010091220143. URL https://doi.org/10.1023/A:1010091220143. (Cited on p. 10, 18, 45, 46, 55, 57, 58, 75, 76, 94, 99)

R. Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, May 1997. ISSN 0377-2217. doi: 10.1016/S0377-2217(96)00385-2. URL https://www.sciencedirect.com/science/article/pii/S0377221796003852. (Cited on p. 58, 98)

R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Information Science and Statistics. Springer-Verlag, New York, 2004. ISBN 978-0-387-21240-1. doi: 10.1007/978-1-4757-4321-0. URL https://www.springer.com/gp/book/9780387212401. (Cited on p. 54, 55, 56, 99, 101, 114)

R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo Method*. Wiley Publishing, 3rd edition, 2016. ISBN 978-1-118-63216-1. (Cited on p. 54)

R. Y. Rubinstein, B. Melamed, and A. Shapiro. *Modern Simulation and Modeling*. Wiley, Mar. 1998. ISBN 978-0-471-17077-8. Google-Books-ID: Z11RAAAAMAAJ. (Cited on p. 55)

V. Sapovadia. Chapter 13 - Legal Issues in Cryptocurrency. In D. Lee Kuo Chuen, editor, *Handbook of Digital Currency*, pages 253–266. Academic Press, San Diego, Jan. 2015. ISBN 978-0-12-802117-0. doi:

10.1016/B978-0-12-802117-0.00013-8. URL https://www.sciencedirect.com/science/article/pii/B9780128021170000138. (Cited on p. 10)

W. F. Sharpe. Capital Asset Prices: A Theory of Market Equilibrium Under Conditions of Risk*. *The Journal of Finance*, 19(3):425–442, Sept. 1964. ISSN 1540-6261. doi: 10.1111/j.1540-6261.1964.tb02865.x. URL http://onlinelibrary.wiley.com/doi/10.1111/j.1540-6261.1964.tb02865.x/abstract. (Cited on p. 7)

K. Sheppard and A. Patton. Good Volatility, Bad Volatility: Signed Jumps and The Persistence of Volatility. *Review of Economics and Statistics*, 97, Oct. 2011. doi: 10.1162/REST_a_00503. (Cited on p. 19, 95)

H. Späth. Algorithm 39 clusterwise linear regression. *Computing*, 22(4):367–373, 1979. (Cited on p. 14, 45, 47, 51)

C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics*, 8:1–21, 2007. (Cited on p. 111)

G. Sucarrat. *lgarch: Simulation and Estimation of Log-GARCH Models*, 2015. URL https://CRAN.R-project.org/package=lgarch. R package version 0.6-2. (Cited on p. 28)

G. Sucarrat. The log-GARCH model via ARMA representations. In *Financial Mathematics, Volatility and Covariance Modelling*. Routledge, 2019. ISBN 978-1-315-16273-7. Num Pages: 24. (Cited on p. 9, 12, 21, 24, 25, 26, 34, 36)

G. Sucarrat. garchx: Flexible and Robust GARCH-X Modelling. *MPRA Paper*, May 2020. URL https://ideas.repec.org//p/pra/mprapa/100301.html. Number: 100301 Publisher: University Library of Munich, Germany. (Cited on p. 11)

G. Sucarrat. garchx: Flexible and robust garch-x modelling. *The R Journal*, 13:276–291, 2021. URL https://journal.r-project.org/archive/2021/RJ-2021-057/. (Cited on p. 11, 20, 24)

G. Sucarrat and A. Escribano. Automated model selection in finance: General-to-specific modelling of the mean and volatility specifications. *Oxford Bulletin of Economics and Statistics*, 74:716–735, 2012. (Cited on p. 16, 22, 26)

G. Sucarrat and A. Escribano. Estimation of log-GARCH models in the presence of zero returns. *The European Journal of Finance*, 24(10):809–827, July 2018. ISSN 1351-847X. doi: 10.1080/1351847X.2017.1336452. URL https://doi.org/10.1080/1351847X.2017.1336452. Publisher: Routledge _eprint: https://doi.org/10.1080/1351847X.2017.1336452. (Cited on p. 28)

G. Sucarrat, S. Grønneberg, and A. Escribano. Estimation and inference in univariate and multivariate log-GARCH-X models when the conditional density is unknown. *Computational Statistics & Data Analysis*, 100:582–594, Aug. 2016. ISSN 01679473. doi: 10.1016/j.csda.2015.12.005. URL https://linkinghub.elsevier.com/retrieve/pii/S0167947315003059. (Cited on p. 25, 28, 29)

T. Teräsvirta. An Introduction to Univariate GARCH Models. In T. Mikosch, J.-P. Kreiß, R. A. Davis, and T. G. Andersen, editors, *Handbook of Financial Time Series*, pages 17–42. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-540-71297-8. doi: 10.1007/978-3-540-71297-8_1. URL https://doi.org/10.1007/978-3-540-71297-8_1. (Cited on p. 10)

R. Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. ISSN 0035-9246. URL https://www.jstor.org/stable/2346178. Publisher: [Royal Statistical Society, Wiley]. (Cited on p. 30, 43)

H. Tong. On a threshold model. In C. Chen, editor, *Pattern Recognition and Signal Processing*, pages 575–586. Sijthoff & Noordhoff, Netherlands, 1978. ISBN 978-90-286-0978-5. URL http://eprints.lse.ac.uk/19500/. Issue: 29 Num Pages: 655 Number: 29. (Cited on p. 93, 96, 97)

H. Tong. Some Basic Concepts. In H. Tong, editor, *Threshold Models in Non-linear Time Series Analysis*, Lecture Notes in Statistics, pages 34–58. Springer, New York, NY, 1983. ISBN 978-1-4684-7888-4. doi: 10.1007/978-1-4684-7888-4_2. URL https://doi.org/10.1007/978-1-4684-7888-4_2. (Cited on p. 9, 93)

H. Tong. Nonlinear Time Series Analysis Since 1990: Some Personal Reflections. *Acta Mathematicae Applicatae Sinica*, 18(2):177, June 2002. ISSN 1618-3932. doi: 10.1007/s102550200017. URL https://doi.org/10.1007/s102550200017. (Cited on p. 93)

H. Tong. Threshold models in time series analysis-30 years on. *Statistics and Its Interface*, 4 (2):107–118, 2011. ISSN 1938-7989. URL https://www.webofscience.com/wos/woscc/summary/b4546f70-70bf-43e0-943c-dac6e336bc12-00fdb4b2/date-descending/1. Place: Somerville Publisher: Int Press Boston, Inc WOS:000293847000002. (Cited on p. 93, 94)

G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697. (Cited on p. 34)

F. Violante and S. Laurent. Volatility Forecasts Evaluation and Comparison. In *Handbook of Volatility Models and Their Applications*, pages 465–486. John Wiley & Sons, Ltd, 2012. ISBN 978-1-118-27203-9. doi: 10.1002/9781118272039.ch19. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118272039.ch19. Section: Nineteen _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118272039.ch19. (Cited on p. 37)

Y. Wang, F. Ma, Y. Wei, and C. Wu. Forecasting realized volatility in a changing world: A dynamic model averaging approach. *Journal of Banking & Finance*, 64:136–149, Mar. 2016. ISSN 0378-4266. doi: 10.1016/j.jbankfin.2015.12.010. URL https://www.sciencedirect.com/science/article/pii/S0378426615003647. (Cited on p. 19, 95)

S. Wu and R. Chen. THRESHOLD VARIABLE DETERMINATION AND THRESHOLD VARIABLE DRIVEN SWITCHING AUTOREGRESSIVE MODELS. *Statistica Sinica*, 17(1):241–S38, 2007. ISSN 1017-0405. URL http://www.jstor.org/stable/26432520. Publisher: Institute of Statistical Science, Academia Sinica. (Cited on p. 96)

D. Xu and Y. Tian. A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, 2(2): 165–193, June 2015. ISSN 2198-5812. doi: 10.1007/s40745-015-0040-1. URL https://doi.org/10.1007/s40745-015-0040-1. (Cited on p. 45)

L. R. Y. Moving blocks jackknife and bootstrap capture weak dependence. *Exploring the Limits of Bootstrap*, 1992. URL https://cir.nii.ac.jp/crid/1573668924689635584. Publisher: Wiley. (Cited on p. 36)

G. Zhang. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50: 159–175, Jan. 2003. ISSN 09252312. doi: 10.1016/S0925-2312(01)00702-0. URL https://linkinghub.elsevier.com/retrieve/pii/S0925231201007020. (Cited on p. 16, 22)

H. Zhang and B. Singer. *Recursive Partitioning and Applications*. Springer Series in Statistics. Springer-Verlag, New York, 2 edition, 2010. ISBN 978-1-4419-6823-4. doi: 10.1007/978-1-4419-6824-1. URL https://www.springer.com/gp/book/9781441968234. (Cited on p. 105, 108)

J. Zhu, C. Wen, J. Zhu, H. Zhang, and X. Wang. A polynomial algorithm for best-subset selection problem. *Proceedings of the National Academy of Sciences*, 117(52):33117–33123, Dec. 2020. doi: 10.1073/pnas.2014241117. URL https://www.pnas.org/doi/10.1073/pnas.2014241117. Publisher: Proceedings of the National Academy of Sciences. (Cited on p. 31, 32)

J. Zhu, X. Wang, L. Hu, J. Huang, K. Jiang, Y. Zhang, S. Lin, and J. Zhu. abess: A fast best subset selection library in python and r. *Journal of Machine Learning Research*, 23(202):1–7, 2022. URL https://www.jmlr.org/papers/v23/21-1060.html. (Cited on p. 27)

R. Zieliński. Uniform strong consistency of sample quantiles. *Statistics & Probability Letters*, 37(2): 115–119, Feb. 1998. ISSN 0167-7152. doi: 10.1016/S0167-7152(97)00108-9. URL https://www.sciencedirect.com/science/article/pii/S0167715297001089. (Cited on p. 78)

COLLEGE MATHSTIC
DOCTORAL BRETAGNE
BRETAGNE OCEANE

Université Bretagne Sud

Titre : Méthodes statistiques pour la modélisation de la volatilité conditionnelle avec des applications en finance

Mot clés : Estimation de la volatilité, Méthode de l'entropie croisée, Apprentissage automatique, Modèles à changement de régime, Séries temporelles financières

Résumé : Cette thèse de doctorat se concentre sur le développement de méthodes statistiques innovantes pour la prévision de la volatilité conditionnelle, en intégrant des techniques d'apprentissage automatique et en améliorant les stratégies d'optimisation numérique. Avec un accent particulier sur le marché hautement volatil du Bitcoin, la recherche introduit l'algorithme de sélection de variables Log-TGARCHX (VS-LTGARCHX), qui permet d'identifier les variables exogènes pertinentes pour améliorer les prédictions de volatilité tout en réduisant le surapprentissage. La thèse présente également l'algorithme Entropie Croisée avec Brisure de Symétrie et K-means (CESBKM), qui affine les techniques de clustering pour capturer les comportements à changement de régime dans les séries temporelles financières. De plus, elle introduit l'algorithme Entropie Croisée avec Arbres de Décision (CEDT) pour les modèles Autorégressifs à Seuil en Boucle Ouverte (OLTAR), combinant l'optimisation globale avec l'apprentissage par arbres de décision afin de mieux capturer les dynamiques de régime. Les applications empiriques montrent que les modèles proposés surpassent les références traditionnelles, offrant des outils méthodologiques utiles avec des applications pratiques dans la modélisation financière.

Title: Statistical methods for conditional volatility modeling with applications to finance

Keywords: Volatility Estimation, Cross-entropy method, Machine learning, regime-switching models, Finacial time series

Abstract: This doctoral thesis focuses on developing innovative statistical methods for conditional volatility forecasting, integrating machine learning techniques, and improving numerical optimization strategies. With an emphasis on the highly volatile Bitcoin market, the research introduces the Variable Selection Log-TGARCHX (VS-LTGARCHX) algorithm, which helps identify key exogenous variables to improve volatility predictions and mitigate overfitting. The thesis also presents the Cross Entropy with Symmetry Breaking and K-Means (CESBKM) algorithm, refining clustering techniques for capturing regime-switching behaviors in financial time series. Additionally, it introduces the Cross Entropy with Decision Trees (CEDT) algorithm for Open-Loop Threshold Autoregressive (OLTAR) models, combining global optimization with decision tree learning to better capture regime dynamics. Empirical applications demonstrate that the proposed models outperform traditional benchmarks, offering useful methodological tools with practical applications in financial modeling.