High-dimensional variable selection and time series classification and forecasting with potential change-points

Thesis

Lok Ting Yuen

Department of Statistics

London School of Economics and Political Science

This dissertation is submitted for the degree of

Doctor of Philosophy

February 2021

To my cat

Declaration

I confirm that Chapter 1-3 were jointly co-authored with Professor Piotr Fryzlewicz and I contributed 75% of these works.

Chapter 1 has been submitted to a peer-reviewed statistical journal and we plan to submit Chapter 2 for publication soon.

I declare that my thesis consists of 31545 words.

Lok Ting Yuen

February 2021

Acknowledgements

First I would like to show my gratitude to my supervisor, Professor Piotr Fryzlewicz, for his continuous support and invaluable guidance throughout my PhD study. It was a great privilege to work with such a fine researcher who has a lot of intriguing ideas and immense knowledge in his expertise. I am deeply grateful for his time and effort spent on my research.

I would like to thank all the staff and colleagues in the Department of Statistics at the London School of Economics for their continuous support. I am also very thankful for the financial support of the LSE Statistics PhD Scholarship. Without it, this thesis could not have been undertaken nor completed.

I would like to thank my parents for their support throughout my life, and being understandable on my decision of pursuing a PhD study.

Finally, I would like to thank my cat. While he probably will never understand the content of my thesis nor my research, his constant support and company made the four year PhD study enjoyable.

Abstract

This thesis studies high-dimensional variable selection and time series with potential change-points.

In Chapter 1 we propose Combined Selection and Uncertainty Visualiser (CSUV), which estimates the set of true covariates in high-dimensional linear regression and visualises selection uncertainties by exploiting the (dis)agreement among different base selectors. Our proposed method selects covariates that get selected the most frequently by the different variable selection methods on subsampled data. The method is generic and can be used with different existing variable selection methods. We demonstrate its variable selection performance using real and simulated data. The variable selection method and its uncertainty illustration tool are publicly available as R package `CSUV` (https://github.com/christineyuen/CSUV). The graphical tool is also available online via https://csuv.shinyapps.io/csuv.

In Chapter 2 we explore the potential and shortcomings of the "estimation-simulation-classification" approach for time series model identification. Assume there is only one realisation of a time series available and we would like to find the true model specification for a given time series. With the success of deep learning in classification in recent years, we explore the possibility of using classifiers for model identification. The application of classifiers on model identification is not straightforward as classifiers require a sufficient number of observations to train but we only have one time series at hand. One possible solution is to generate pseudo training data that is similar to the observed time series, and use them to fit the classifiers. We call it the "estimation-simulation-classification" (ESC) approach.

We find that if the model complexity is not taken into account, more flexible models are favoured by this approach. The advantage of using a good classifier can be discounted by the ignorance of the model complexity, and some simple methods (e.g. information criteria) that take into account the model complexity may outperform classifiers with the ESC approach. Based on our observations on the ESC approach, we propose using BIC and consider ResNet via the ESC approach for time series model identification. The newly proposed methods are implemented in `R` and will be available online via https://github.com/christineyuen/ESC.

In Chapter 3, we propose different procedures to extend the use of Narrowest-Over-Threshold (NOT) to time series with dependent noise, with the objective to provide better forecasting performance. The new method takes into account the potential dependent structure of the noise. We also explore using cross-validation to select the set of change-points from the NOT solution path. We demonstrate the prediction performance of the proposed procedures using real and simulated data, and compare the performance with some other methods in different settings. The newly proposed methods are implemented in `R` and will be available online via https://github.com/christineyuen/NOT-ARMA.

Table of contents

List of figures

List of tables

Chapter 1

Exploiting disagreement between high-dimensional variable selectors for uncertainty visualisation

1.1 Introduction

Model and variable selection in high-dimensional regression settings have been widely discussed in the past decades. In the context of the linear model, the best subset selection (dated back to at least Beale et al., 1967) is computationally infeasible when the number of covariates p is large. Regularisation methods with convex penalties, such as the Lasso (Tibshirani, 1996), are capable of performing variable selection in large-p settings and yet they are computationally efficient. Elastic Net (Zou and Hastie, 2005) is believed to be particularly suitable for designs with a high degree of correlation between the covariates. Group Lasso (Yuan and Lin, 2006) is designed for situations in which the covariates are best considered in groups. Regularised regression methods with non-convex penalties such as the smoothly clipped absolute deviation (SCAD, Fan and Li, 2001) and minimax concave penalty (MCP, Zhang et al., 2010) methods are designed to reduce estimation bias. The theoretical evaluation of the properties of these and many other variable selection methods has been the subject of intense research effort. For example, the irrepresentable condition (Zhao and Yu, 2006) is sufficient and almost necessary for the Lasso to be sign consistent. Fan and Lv (2010) provide a detailed review of different variable selection methods in high-dimensional settings.

There has also been a growing focus on post-selection inference. Van de Geer et al. (2014), Zhang and Zhang (2014) and Javanmard et al. (2018) advocate the de-biasing approach, which constructs confidence intervals for covariates by de-sparsifying the Lasso estimators. Lee et al. (2016), Tibshirani et al. (2016) and Tibshirani et al. (2018) propose a conditional approach which provides confidence intervals for the selected covariates using the distribution of a post-selection estimator conditioning on the selection event. Chatterjee and Lahiri (2011) and Liu et al. (2013) suggest using bootstrapping on some existing variable selection methods.

In this chapter we focus on identifying the true set of covariates and illustrating the selection uncertainty in the linear model. We assume that the observed data are the realisation of:

$$Y_i = \beta_0 + \sum_{j=1}^{p} \beta_j X_i^j + \varepsilon_i, \quad i = 1, ..., n, \tag{1.1}$$

where p is the number of covariates, n is the number of observations, and we potentially have $p > n$. X_i^j is the j^{th} covariate of the i^{th} observation of $\boldsymbol{X}$ and $\boldsymbol{X}$ is a fixed $n \times p$ design matrix. $\boldsymbol{X}$ is standardised with each covariate X^j has $\sum_{i=1}^n X_i^j/n = 0$ and $\sum_{i=1}^n (X_i^j)^2/n = 1$. ε is i.i.d. noise with mean zero and variance σ^2. Furthermore, the model is assumed to be sparse with the set of true covariates $S = \{j \in \{1, ..., p\} : \beta_j \neq 0\}$, $s = |S| \ll p$.

Less effort has been devoted in the literature to *selecting the best variable selection method* for the data at hand. Various theoretical performance guarantees are available for a range of methods, but many of them are not testable in practice; for instance, checking the irrepresentable condition usually requires knowing the true set of covariates. Therefore, this type of theory can be of limited use in method selection. How to select a method remains an open and yet very important question to ask, as it affects our selection of the set of relevant variables. To illustrate this impact, let us consider two real-life datasets in Examples 1.1 and 1.2.

Example 1.1 (Riboflavin data)**.** The riboflavin dataset concerns the riboflavin (vitamin B2) production by bacillus subtilis. The response is the logarithm of the riboflavin production rate by bacillus subtilis and the $p = 4088$ covariates are the logarithms of the expression levels of 4088 genes. The number of samples is $n = 71 \ll p$. The dataset is available in the R package `hdi`.

Example 1.2 (Prostate cancer data, Stamey et al., 1989)**.** The prostate cancer dataset comes from a study that examined the relationship between the level of prostate-specific antigen and $p = 8$ clinical measures (logarithm of weight, age, Gleason score, among others) in men who were about to receive a radical prostatectomy. The sample size is $n = 97$. The dataset is available in the R package `lasso2`.

We process the datasets using five different variable selection methods: the Lasso, Elastic Net, relaxed Lasso (Meinshausen, 2007), MCP and SCAD in R with default tuning in the corresponding R packages (see Section 1.5.1.1 for more details). We justify the choice of these particular methods in Section 1.3.3.5. Working with default parameters would be a commonly used starting point for the non-expert applied user. The selection results are shown in Figures 1.1 and 1.2.

Figure 1.1 shows that for the riboflavin dataset the sets of covariates selected vary significantly among the methods, which makes it difficult to justify the validity of the set of covariates selected using any one method. For the prostate cancer dataset, even though there are only eight covariates to choose from, there is still selection disagreement among the methods (Figure 1.2).

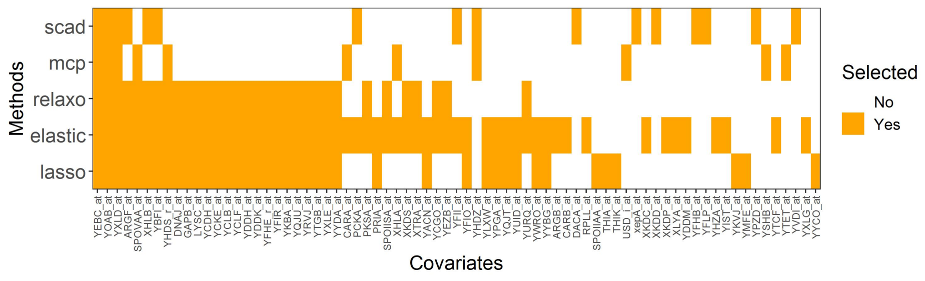

Fig. 1.1 Graphical illustration of selections by different variable selection methods (Lasso, Elastic Net, relaxed Lasso, MCP and SCAD) with default tuning using the riboflavin dataset from Example 1.1. Covariates that are not selected by any methods are not shown in the graph for readability.

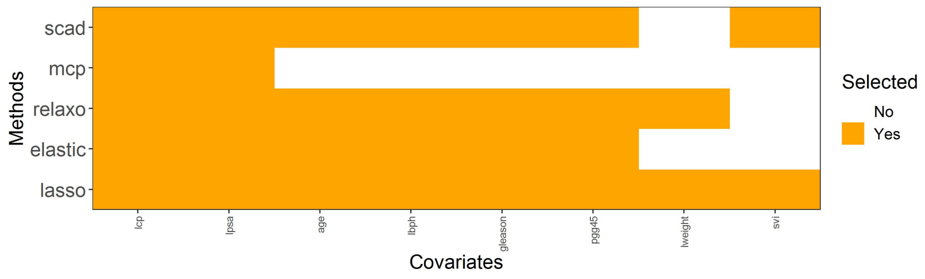

Fig. 1.2 Graphical illustration of selections by different variable selection methods (Lasso, Elastic Net, relaxed Lasso, MCP and SCAD) with default tuning using the prostate dataset from Example 1.2.

Such disagreement among methods as shown in Figures 1.1 and 1.2 is not an exception but a common observation. The distance heat maps in Appendix A.3 show that selection disagreement manifests itself across different simulation settings (see Section 1.5 for more details on the simulation settings). Having observed disagreement, one possible way to proceed would be to rank the different models considered (e.g. using cross-validation or an information criterion) and select the highest-ranked one. In this chapter, we consider eBIC (Chen and Chen, 2008) and delete-$n/2$ cross-validation (Zhang and Yang, 2015) as they are suitable for high-dimensional settings. Further details of these two methods are discussed in Section 1.2.1. Our simulation results show that in general eBIC performs better than the delete-$n/2$ cross-validation in terms of variable selection (see Tables A.1-A.15 in

Appendix A.4). In fact, eBIC in many simulation settings performs very similarly to the best performing individual variable selection method.

Although eBIC seems to be able to select a single good model fit, can more be said regarding the uncertainty of variable selection, based on the disagreement between the methods tested? The similarities and disagreements among the different variable selectors, which is a piece of information not typically used by any one of them, may provide us with some useful insight. For example, in Figure 1.1 all of the methods select the first three covariates whereas the remaining covariates are selected by some of the methods only. Does it mean that the first three covariates are more likely to be the true covariates? This question is central to this chapter, and motivates our main development, described next. In this chapter, we propose a new tool for variable selection with uncertainty visualisation, termed Combined Selection and Uncertainty Visualiser (CSUV). CSUV combines, in a particular way, a number of different base variable selection methods into a new variable selector, and illustrates the output of this new selector together with a graphical representation of its uncertainty. It makes use of sets of covariates selected on different subsamples of the data with different variable selection methods. A full description of the proposed method is in Section 1.3 and 1.4. The variable selection part of the proposed procedure can be summarised as follows: first, split the data into the training and test sets and fit different variable selection methods on the training set over a grid of tuning parameter values. Estimate the performance of the fitted models on the test set, and retain only the k best-performing models. Repeat the process a number of times and select the covariates that appear the most frequently in the collection of the retained fitted models.

The other component of CSUV is a graphical tool designed to visualise the selection uncertainty by using disagreement among the different model fits. See Figure 1.3 as an example of a graphical output of CSUV. The plot shows the frequency with which each covariate is selected and the variability of the non-zero estimated coefficients. As we will see in Section 1.4.2, the graphical tool can be used to assist variable selection.

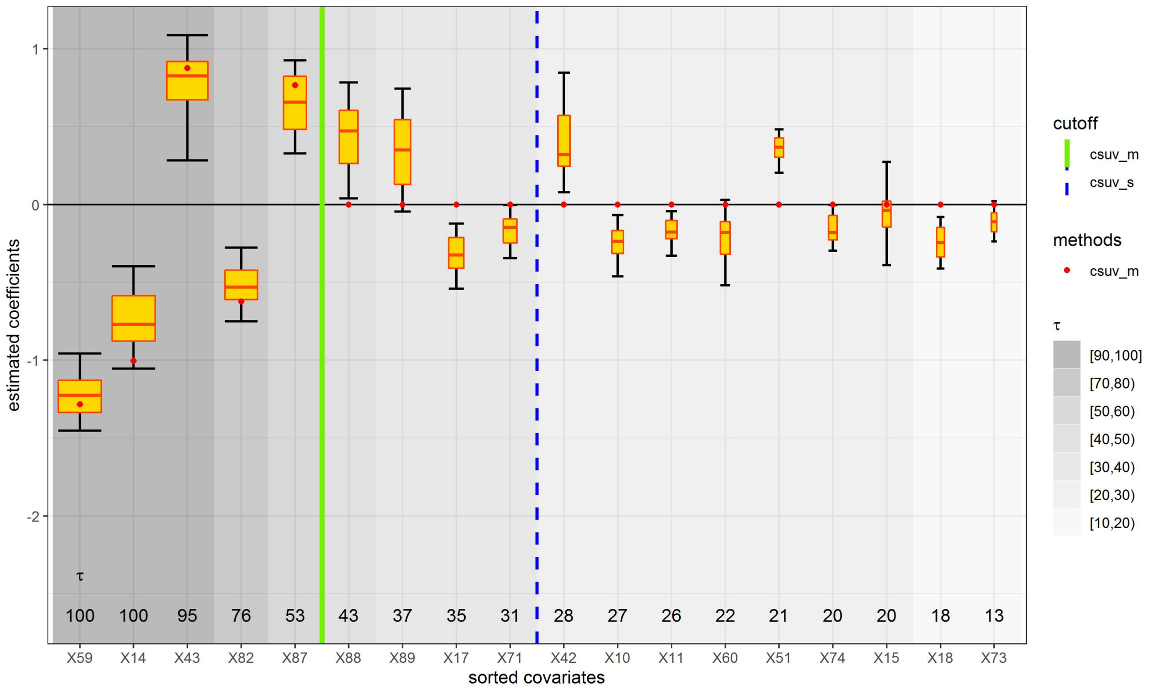

Fig. 1.3 Example of the CSUV graphical tool with simulated data from model 2 parameter setting 5 (see Section 1.5.1.4 for more details on the simulation setting). Box plots illustrate the empirical distributions of the estimated coefficients conditional on them being non-zero, and the whiskers represent their 5% and 95% percentiles. The ordering of the covariates is according to the CSUV solution path (see Definition 1.3 in Section 1.3.4) and the width of each box plot along the x-axis is proportional to the level of the relative same sign frequency τ_j (see Definition 1.1 in Section 1.3.2; heuristically, the higher the value of τ_j, the higher the frequency with which the corresponding variable has been selected with the same positive or negative sign). The numbers at the bottom of the graph show the actual values of τ_j times 100 and the shade in the background corresponds to the level of τ_j with ranges as shown in the legend. Dots (red in the colour version) are the estimated coefficients by CSUV-m (see Definition 1.2 in Section 1.3.3.4). The solid vertical line (green in the colour version) represents the cut-off of CSUV-m, and the dotted vertical line (blue in the colour version) represents the cut-off of CSUV-s (see Definition 1.4 in Section 1.3.4). Covariates with $\tau_j < 0.1$ are not shown for readability.

Our numerical experience (see Figure 1.4 for a summary) suggests that the fitted models selected by CSUV tend to be distributed fairly uniformly over the entire range of the base variable selection methods used. This shows CSUV generally makes use of most of the base variable selection methods to get the final fitted model.

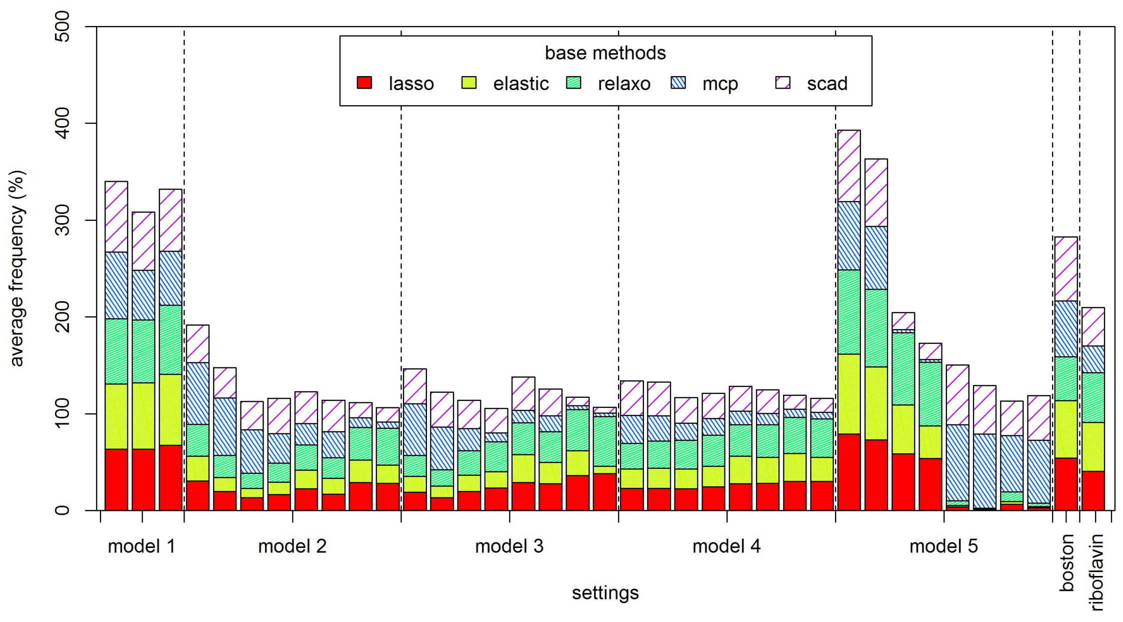

Fig. 1.4 Average relative frequency of the constituent methods selecting the same set of covariates as the fitted models retained by CSUV when the Lasso, Elastic Net, relaxed Lasso, MCP and SCAD are used as the constituent variable selection methods for CSUV in our simulations (see Section 1.5.1.4 for more details on the simulation settings). The sum of the average frequency of methods can be more than 100% as multiple methods can select the same set of covariates.

The chapter is organised as follows. In Section 1.2, we provide a literature review on related works. In Section 1.3, we discuss the main ideas behind CSUV, and we present the variable selection and coefficient estimation part of CSUV. In Section 1.4, we introduce the graphical tool of CSUV to illustrate the disagreement in variable selection and the variability in coefficient estimation, and demonstrate its capability in assisting variable selection. In Section 1.5, we present the simulation results. We conclude the chapter with a discussion in Section 1.6.

1.2 Literature review

1.2.1 Model selection procedures

One possibility open to analysts when faced with competing fitted models is to select one of them. For example, Chen and Chen (2008) propose eBIC, an extension of BIC to high-dimensional data which takes into account both the number of unknown parameters and the complexity of the model space. Zhang and Yang (2015) advocate the use of the delete-$n/2$ cross-validation to select a method among all the candidate methods. For each iteration, delete-$n/2$ cross-validation uses half of the data for fitting and half for evaluation. The authors argue that in order to consistently identify the best variable selection procedure by cross-validation, the evaluation part has to be sufficiently large so that there are (1) more observations in the testing part to provide better evaluation and (2) fewer observations in the training part to magnify the difference in performance between methods.

1.2.2 Model combination with a single method

Model combination with subsampling has been used to improve variable selection performance of a single variable selection method. For example, Bolasso (Bach, 2008) fits the Lasso on each bootstrap sample and takes the intersection of all the selections. Wang et al. (2014) propose the median selection subset aggregation estimation (MESSAGE) algorithm which aims to perform variable selection on large-n datasets. It runs a variable selection method (e.g. the Lasso) in parallel on each subset of the data and selects the set of covariates whose median is non-zero. The ranking-based variable selection (Baranowski et al., 2018) algorithm uses subsampling to identify the set of consistently highly-ranked covariates. Stability selection (Meinshausen and Bühlmann, 2010 and Shah and Samworth, 2013) provides control over the finite sample familywise type I errors via subsampling. The stability selection procedure repeatedly samples observations and fits the sampling data using a variable selection method (e.g. the Lasso). It then keeps the covariates with selection frequency higher than a certain threshold.

Similarly to the methods above, CSUV, our proposal, fits variable selection methods on subsampled data and selects the covariates that appear the most frequently. Unlike these other approaches, however, CSUV makes use of different variable selection methods as we observe that no one method outperforms all other methods in all settings. This brings various advantages, including obtaining access to good model fits from different variable selection methods, and being able to exploit disagreement between the selectors to evaluate selection uncertainty. We elaborate on these points later.

1.2.3 Model combination with multiple methods

Adaptive regression by mixing (ARM, Yang, 2001) and its variation, adaptive regression by mixing with screening (ARMS, Yuan and Yang, 2005), aggregate fits from different methods by estimating weights through subsampling. ARM uses half of the data to fit some candidate models/procedures (e.g. smoothing splines with cross-validation tuning) and estimate σ. The remaining data is used to evaluate the prediction loss. The weight for each candidate model/procedure is calculated using $\hat{\sigma}$ and prediction loss. ARM gets the final weights by averaging the weights from different iterations. Finally it fits the full set of data using all the candidate models/procedures and obtains the final model by averaging the fits using the estimated weights. ARMS is similar to ARM except it uses half of the data to calculate AIC or BIC and retains only the models that have low AIC or BIC. The final fitted model from ARM or ARMS is not necessarily sparse as it is a weighted average of a number of models. Variable selection deviation measures (VSD, Nan and Yang, 2014) aim to provide a sense of how trustworthy a set of selected covariates is. The VSD of a target model m is the weighted cardinality of the symmetric difference between m and each candidate model. Nan and Yang (2014) suggest using the sets of fitted models on the solution paths from the

Lasso, SCAD and MCP as candidate models and the weight of each candidate model is calculated based on information criteria or ARM. The simulation results in Nan and Yang (2014) show that a large VSD compared to the size of the target model means that the target model is not trustworthy, but a small VSD does not necessarily mean that the target model is close to the true model. Yang and Yang (2017) propose to select a set of covariates that minimises the total Hamming distance with all the candidate models in terms of VSD (we refer to this method as VSD-minimising in the remainder of the chapter). The authors also propose using different thresholds, where the threshold of 0.5 is equivalent to minimising the standard Hamming distance. Another related method is Sparsity Oriented Importance Learning (SOIL, Ye et al., 2018), for which it attempts to measure the variable importance for high-dimensional regression. SOIL is similar to VSD in terms of the weighting method and the set of fitted models used, but its weighting is for each variable instead of for each fitted model. For variable selection method combinations that do not involve subsampling, Tsai and Hsiao (2010), Mares et al. (2016) and Pohjalainen et al. (2015) provide empirical results on combining sets of selected covariates from different variable selection methods by intersection, union and/or some other set operations.

Both our method, VSD and SOIL use resampling and different variable selection methods to provide an assessment of how good the final set of covariate selection is. VSD focuses on the whole model fit and SOIL and our method focus on the uncertainty of individual covariates. Our method has a graphical tool for which is designed to illustrate these uncertainties. In terms of methodology detail, our method combines the sets of covariates selected in resampling fits whereas VSD and SOIL combine sets of covariates selected on the solution path when fitting using all the data. Resampling data is only used in VSD and SOIL for calculating the weight of each set of covariates. In our simulation study we compare the variable selection performance of our method to the VSD-minimising method proposed by Yang and Yang

(2017), as it is the method the most similar to CSUV. The simulation results in Section 1.5.2.2 show that in general our method outperforms the VSD-minimising model.

1.3 CSUV variable selection methodology

1.3.1 Simple aggregation

The first goal of this chapter is to use the similarity of fits from different methods to obtain the final set of covariates. One naive way to do so would be as follows.

- Step 1: fit the data using different variable selection methods.
- Step 2: record the percentage of times a covariate X_j is selected among the different methods. Denote it by θ_j.
- Step 3: get the final set of covariates by selecting covariates with high θ_j's. For example, select the set of covariates $\{X_j : \theta_j \geq 0.5\}$.

Different variable selection methods optimise different objective functions. In the case of regularised regression, the difference among methods is usually in terms of the penalty. If a covariate is selected by the majority of methods, it means the covariate is chosen to minimise many different objective functions. We expect that a true covariate j should have a high θ_j, i.e. it should frequently be chosen regardless of the objective function used. This simple procedure, however, suffers from the following drawbacks.

- Some variable selection methods can be similar in terms of selection regardless of the data as their objective functions are similar. Taking an extreme example, if we include two equivalent variable selection methods, such as the constrained and the penalised forms of the Lasso with equivalent regularisation parameters, the sets selected by both methods will be the same. Such set is selected twice not because it maximises two different object functions but merely because two equivalent methods are considered. This issue can cause an uneven “sampling” of methods and the corresponding fitted models.

- The above procedure assigns the same weight to all the base methods. When the performance across methods is very different (for example one method is substantially better than the others), such equal weight assignment is not ideal.
- Several methods can be wrong at the same time. For example, a false covariate can be wrongly selected by most methods if it has a spuriously high sample correlation with the response. When all methods are not performing well, a false covariate may have a high θ_j.

In the next section, we discuss how to overcome these drawbacks.

1.3.2 CSUV variable selection

Motivated by the above discussion, the variable selection in CSUV uses the general simple aggregation principles introduces in the previous section, but is also supplemented with the additional principles below:

- Only include the fitted models that exhibit good performance, in the sense specified in Section 1.3.3.2.
- Repeat the fitting on subsampled data, to incorporate the variability in selection caused by the variability in data.

The variable selection procedure of CSUV can be summarised as follows. First, randomly split the data into training and test sets, and fit different variable selection methods on the training set over a grid of regularisation parameters without tuning (see Section 1.3.3.5 and 1.5.1.1 for more the details on the grid of regularisation parameters considered). Then, use the test set to calculate the performance of the fitted models and retain only the first k fitted models that have the best performance (see Section 1.3.3.2 for more details on performance measure). Repeat the process many times to record a list of retained fitted models. Finally, select the covariates that appear the most frequently with the same positive or negative sign in the retained fitted models. The pseudo-code in Algorithm 1.1 provides a more detailed description of the variable selection part of CSUV. Coefficient estimation on the selected set is discussed in Section 1.3.3.1.

Before we present Algorithm 1.1, we define the relative same sign frequency τ_j, which measures the percentage of times that the jth covariate is selected with the same sign.

Definition 1.1 (Relative same sign frequency τ_j)**.** *Assume we have a set of fitted models $\mathscr{M}$. The relative same sign frequency of covariate X_j is defined as:*

$$\tau_j = \frac{1}{|\mathscr{M}|} \max\left(\sum_{M_k \in \mathscr{M}} \mathbb{1}_{\hat{\beta}_j^{M_k} > 0}, \sum_{M_k \in \mathscr{M}} \mathbb{1}_{\hat{\beta}_j^{M_k} < 0} \right)$$

where $\hat{\beta}_j^{M_k}$ is the estimated coefficient of the j^{th} covariate on the fitted model $M_k \in \mathscr{M}$, and $\mathbb{1}_x$ is the indicator function.

Algorithm 1.1 Select a set of covariates in CSUV

```
Input: variable selection methods $\mathscr{A}_1, ..., \mathscr{A}_R$ with the corresponding generation of the grid of regularisation parameters; $n$ observations with $p$ covariates $\boldsymbol{X}$ and response $Y$; number of repetitions $B$, percentile parameter $q$; frequency threshold $t$; percentage of data used in training set $w\%$; performance measure.
Output: set of selected covariates $\hat{S}$.
1: for $b$ in $\{1, ..., B\}$ do
2:     randomly assign $w\%$ of the observations as training data with labels $I^b_{train}$ and the rest as test data with label $I^b_{test}$. Fit data with label $I^b_{train}$ using $\mathscr{A}_1, ..., \mathscr{A}_R$ over grids of $K'_r$ different values of the corresponding regularisation parameters, $r \in \{1, ..., R\}$. For each method $\mathscr{A}_r$, denote the fitted models as $\tilde{M}^b_{r,1}, ..., \tilde{M}^b_{r,K'_r}$ and the set of covariates selected by each fitted model as $S^{\tilde{M}^b_{r,k}} = \{j : \tilde{\beta}_j^{\tilde{M}^b_{r,k}} \neq 0\}, k \in \{1, ..., K'_r\}$.
3:     remove any duplication *within each method* in terms of variable selection to get $S^{\tilde{M}^b_{r,1}}, ..., S^{\tilde{M}^b_{r,K_r}}$ such that for each $r, S^{\tilde{M}^b_{r,k}} \neq S^{\tilde{M}^b_{r,k'}}\ \forall k \neq k' \in \{1, ..., K_r\}$. Record the sets of covariates selected by each fitted model $S^{\tilde{M}^b_{1,1}}, ..., S^{\tilde{M}^b_{1,K_1}}, ..., S^{\tilde{M}^b_{R,1}}, ..., S^{\tilde{M}^b_{R,K_R}}$ and re-index as $S^{\tilde{M}^b_1}, ..., S^{\tilde{M}^b_{K^b}}$, where $K^b$ is the number of fitted models recorded.
4:     if the number of selected covariates $|S^{\tilde{M}^b_k}| < |I^b_{train}|$, refit the selected set of covariates $S^{\tilde{M}^b_k}$ using ordinary least squares (OLS), to get the fitted models $\hat{M}^b_1, ..., \hat{M}^b_{K^b}$ with the estimated coefficients $\hat{\beta}_j^{\hat{M}^b_k}$. Otherwise, set $\hat{\beta}_j^{\hat{M}^b_k} = \tilde{\beta}_j^{\tilde{M}^b_k}$.
5:     use data with label $I^b_{test}$ to estimate the performance of each fitted model $\hat{M}^b_k$ from Step (4). Order the models $\hat{M}^b_1, ..., \hat{M}^b_{K^b}$ by the performance measure calculated from the best to the worst to obtain $\hat{M}^b_{(1)}, ..., \hat{M}^b_{(K^b)}$.
6:     retain the first $q\%$ of the fitted models $\hat{M}^b_{(1)}, ..., \hat{M}^b_{(K^b_q)}$, where $K^b_q = \text{round}(K^b \times q/100)$.
7: end for
8: denote the set of retained fitted models by $\mathscr{M} = \{\hat{M}^1_{(1)}, ..., \hat{M}^1_{(K^1_q)}, ..., \hat{M}^B_{(1)}, ..., \hat{M}^B_{(K^B_q)}\}$.
9: calculate the relative same sign frequency $\tau_j$ for each variable $j$ according to Definition 1.1 and select the covariates such that:
       $$\hat{S} = \{j : \tau_j \geq t\}$$
10: return $\hat{S}$.
```

In Step (3) of Algorithm 1.1, duplicated sets of covariates selected within each method from Step (2) are removed. This is because while multiple selections of a set of covariates in Step (2) may suggest that the covariates in the set are likely to be the true covariates, it can also just be because many similar regularisation parameter values have been used in Step (2). In order to reduce the dependency of the frequency of the appearance of a set of covariates on the choice of the grid of regularisation parameters, duplicated sets of covariates selected within each method from Step (2) are removed. Note that duplicated sets of covariates selected across methods are not removed.

Algorithm 1.1 involves repeated fits on subsamples of data, and this can be computationally expensive. Fortunately, the algorithm can easily be parallelised by running iterations on different cores/machines. This makes the algorithm feasible for high-dimensional data analysis. For example, on a 3-core machine, Applying CSUV on the riboflavin dataset of Example 1.1 takes less than 2 minutes when following the specifications recommended in Section 1.3.3 ($B = 100$, the constituent methods = {Lasso, MCP, SCAD}, etc.). Applying CSUV with the recommended specification on each simulated dataset in Section 1.5.1.4 with $p = 300$ takes less than 30 seconds.

1.3.3 Specifications for CSUV variable selection

Algorithm 1.1 provides a general framework for the CSUV variable selection approach. Here we discuss how the various parameters should or may be set for practical use.

1.3.3.1 Coefficient estimation

Algorithm 1.1 only selects a set of covariates without estimating the β coefficients. In our implementation we use ordinary least squares (OLS) to estimate the β coefficients on the selected set $\hat{S}$ using the full set of data to form the final fitted model. If the number of covariates selected is larger than the number of observations, we use ridge regression to estimate the coefficients by cross-validation (in this case, we use the default cross-validation setting from the `glmnet` R package).

1.3.3.2 Performance measure

Step (5) of Algorithm 1.1 aims to rank the fitted models based on their variable selection performance. As we do not know the true covariates, we are not able to measure variable selection performance directly. In general, in attempting to select fitted models or methods with good variable selection performance, it is common to use prediction measures such as MSE or information criteria such as BIC or eBIC. Theoretically, BIC is consistent in model identification when p is fixed and eBIC is consistent in high-dimensional settings (Chen and Chen, 2008). Our empirical experiments, however, show that when using BIC or eBIC as performance measures in Algorithm 1.1, the resulting fitted models tend to select too few covariates so the final selection by CSUV omits too many true covariates. By contrast, using MSE as the performance measure in Algorithm 1.1 in our simulation settings provides good variable selection performance. Although MSE measures prediction rather than variable selection performance, MSE is often used for variable selection methods such as in selecting tuning parameter λ for SCAD (Fan and Li (2001)).

1.3.3.3 Percentage of data used in training set $w\%$

Following Yang (2001), Yuan and Yang (2005) and Zhang and Yang (2015), we use 50% of the data for fitting and the remaining 50% for testing; this splitting ratio attempts to ensure a sufficiently large sample size for both. Stability selection (Meinshausen and Bühlmann, 2010 and Shah and Samworth, 2013) also uses the same splitting ratio although their rationale is that subsampling with such a ratio behaves similarly to bootstrapping.

Empirically, our simulations show that using a smaller training set (25% of the data) results in selecting fitted models with too few covariates. When using a large training set

(75% of the data), the selected fitted models are too similar to each other, which causes CSUV to select too many false covariates.

1.3.3.4 Frequency threshold t

The frequency threshold t features in Step (9) of Algorithm 1.1. In this chapter, we set $t = 1/2$, which means that covariates with $\tau_j \geq 1/2$ are selected. We have the following definition, in which the “m" stands for median, because selecting covariates with $\tau_j \geq 1/2$ is equivalent to selecting covariates with a non-zero median in $\mathscr{M}$.

Definition 1.2 (CSUV-m)**.** *The CSUV method described by Algorithm 1.1 and using $t = 1/2$ is denoted by CSUV-m.*

The following results hold.

Proposition 1.1. *If the signs for the non-zero $\hat{\beta}_j^k$'s for all k for which $M_k \in \mathscr{M}$ are the same, i.e.*

$$\tau_j = \frac{\sum_{\{k|M_k\in\mathscr{M}\}} \mathbb{1}_{\beta_j^k\neq 0}}{|\mathscr{M}|},$$

then selecting a covariate j when $\tau_j \geq 1/2$ is equivalent to minimising the average Hamming distance between the final selected sets of covariates $\hat{S}$ and all the fitted models $M_k \in \mathscr{M}$.

Proposition 1.2. *Let $\tau_j^+ = \frac{1}{|\mathscr{M}|}\sum_{\{k|M_k\in\mathscr{M}\}} \mathbb{1}_{\hat{\beta}_j^k>0}$ and $\tau_j^- = \frac{1}{|\mathscr{M}|}\sum_{\{k|M_k\in\mathscr{M}\}} \mathbb{1}_{\hat{\beta}_j^k<0}$ (note $\tau_j = max(\tau_j^+, \tau_j^-)$). Consider the following distance function between a model M and all the fitted models $M_k \in \mathscr{M}$:*

$$dist(M,\mathscr{M}) = \sum_{j=1}^{p} \sum_{\{k|M_k\in\mathscr{M}\}} |s_j^M - sign(\hat{\beta}_j^k)|$$

where s_j^M is the sign of the coefficient of the covariate j in model M which can take the value -1, 0 or 1. Selecting a covariate j when $\tau_j \geq 1/2$ and setting $s_j^M = +1$ when $\tau_j^+ \geq 1/2$ and -1 when $\tau_j^- \geq 1/2$ minimises $dist(M,\mathscr{M})$.

The proofs are in Appendix A.1 and A.2. Selecting covariates via thresholding τ_j in CSUV-m (Definition 1.2) is not the only option. In Section 1.3.4 we introduce CSUV-s, which uses information provided by the sizes of the retained models.

1.3.3.5 Constituent variable selection methods $\mathscr{A}_1, ..., \mathscr{A}_R$

CSUV is designed to be generic so that any variable selection methods can be used as the constituent methods $\mathscr{A}_1, ..., \mathscr{A}_R$ in CSUV. Ideally, all the methods $\mathscr{A}_r$ should have good variable selection performance, and there should be some variability among the methods in terms of false selection. The constituent methods should also be computationally efficient as Algorithm 1.1 fits the constituent methods on subsampled data multiple times. In this chapter, we choose the Lasso, MCP and SCAD to be the default constituent methods as they are optimising different objective functions. Methods like Elastic Net or relaxed Lasso are not selected as the default constituent methods as they are relatively similar to Lasso. The default constituent methods we choose are also computationally feasible in high-dimensional settings with efficient fitting algorithms available, and there is also a default way to compute the grid of regularisation parameters to consider. For example, the R package `ncvreg` for MCP and SCAD by default computes a sequence of parameters λ with equal spacing on the log scale and of length 100, starting from the smallest value 0.001. See Section 1.5.1.1 for more details on the R packages used. We do not consider some two-stage methods such as the adaptive Lasso (Zou, 2006) as they are relatively slow. We also do not consider methods without default parameter tuning in R (e.g. the Dantzig selector, Candes and Tao, 2007) as it makes the comparison with other methods like delete-$n/2$ cross-validation more complicated.

CSUV can also tolerate duplicated or very similar methods, although it is not recommended due to the computational time. Including duplicated or very similar methods, though not preferable as it extends the computation time, in our experience it does not affect the variable selection performance much when the percentage parameter q is small. In our simulation, when methods that usually select similar sets as the Lasso (such as the Elastic Net or relaxed Lasso) are included, the performance of CSUV is close to when these similar methods are not included.

1.3.3.6 Percentile parameter q

In our simulation $q = 0$ and $q = 5$ are used with MSE as the performance measure recommended in Section 1.3.3.2, with $q = 0$ corresponds to selecting one single fitted model with the lowest MSE. The performance is similar with $q = 0$ and $q = 5$, although $q = 0$ provides slightly better results. When the larger percentile $q = 20$ is used, again the performance is still close to that of $q = 0$. With $q = 50$, CSUV performs poorly as it includes too many fitted models.

1.3.3.7 Number of repetitions B

The number of repetitions B should be large enough to stabilise the value of τ_j and at the same time it should not be too large so that Algorithm 1.1 can be run within a reasonable time. $B = 100$ is used in our simulation when $n = 100$ and $p = 100, 300$ and it provides a good compromise between stability and computational time.

1.3.4 Solution path and selection with other thresholds

The CSUV-m uses $t = 1/2$, which is equivalent to selecting the covariates for which $\tau_j \geq 1/2$. Empirically, based on our simulation results, CSUV-m provides good variable selection results by striking a good balance between false inclusion and false omission. Comparing to other variable selection methods, CSUV-m usually includes many fewer false covariates, with the trade off being that it occasionally omits some true covariates. When the analyst's focus is on performance criteria other than variable selection, for example on prediction, they may want to select more covariates. This can be done by considering other thresholds t on the sign frequency τ_j, or a threshold on the model size as described in Algorithm 1.2.

Algorithm 1.2 generates a solution path (see Definition 1.3) by ordering covariates from the highest to the lowest relative same sign frequency τ_j. This solution path can be regarded as a series of nested sets of covariates with increasing model sizes. Given a fixed model size s as the size threshold, Algorithm 1.2 selects the first s covariates on the solution path and returns them as the final selection set.

Definition 1.3 (CSUV solution path)**.** *The CSUV solution path orders covariates so that*

$$R_j < R_{j'} \text{ if } \tau_j > \tau_{j'} \text{ or } (\tau_j = \tau_{j'} \text{ and } |\bar{\hat{\beta}}_j| > |\bar{\hat{\beta}}_{j'}|)$$

where R_j is the position of covariate j on the solution path, τ_j is the relative same sign frequency calculated in Step (9) of Algorithm 1.1 and $\bar{\hat{\beta}}_j$ is the average of the estimated coefficients in $\mathscr{M}$ in Step (8) of Algorithm 1.1.

Algorithm 1.2 CSUV with a given model size

```
Input: relative same sign frequency $\tau_j$ calculated in Step (9) and $\mathscr{M}$ in Step (8) of Algorithm
    1.1; size threshold $s$.
Output: set of selected covariates.
1: obtain the solution path (Definition 1.3) using $\tau_j$ and $\mathscr{M}$.
2: return the first $s$ covariates ordered in Step (1), i.e.
       $\{j | R_j \leq s\}$
```

The standardisation of the design matrix in Equation (1.1) ensures the comparison of the size of the estimated coefficients is meaningful. In the particular implementation of CSUV described in this chapter, we set the size threshold s equals to the median size of the selected sets in $\mathscr{M}$ in Step (8) of Algorithm 1.1 and we define CSUV with this threshold as CSUV-s.

Definition 1.4 (CSUV-s). *The CSUV method in Algorithm 1.2 with size threshold $s = median(|S^{\tilde{M}^1_{(1)}}|, ..., |S^{\tilde{M}^1_{(K)}}|, ..., |S^{\tilde{M}^B_{(1)}}|, ..., |S^{\tilde{M}^B_{(K)}}|)$, i.e. s equal to the median size of the selected sets in $\mathscr{M}$ in Step (8) of Algorithm 1.1 is denoted by CSUV-s, where s stands for size.*

1.4 CSUV visualisation of uncertainty

1.4.1 Graphical component of CSUV

In this section, we introduce the graphical component of CSUV, which is a tool designed to illustrate the variable selection and estimation uncertainty. An example of a plot is shown in Figure 1.3 and the graphical tool is available interactively via a Shiny app at https://csuv.shinyapps.io/csuv and in the R package `CSUV`. It has the following ingredients.

- Box plots that visualise the estimated coefficient uncertainty: each box plot correspond to a covariate X_j and it shows the lower and the upper quartiles of the empirical distributions of the estimated coefficients conditional on them being non-zero, i.e. only take into account of the non-zero coefficients $\{\hat{\beta}_j^{M_k} | \hat{\beta}_j^{M_k} \neq 0, M_k \in \mathscr{M}\}$ from Step (8) of Algorithm 1.1. Its whiskers corresponding to the 5% and 95% percentile of the non-zero estimated coefficients (default, level can be changed in the `CSUV` R package). The width of each box is proportional to the relative same sign frequency τ_j (Definition 1.1). The median value of the non-zero estimated coefficients is shown as a horizontal line in each box (red in the colour version). The box plots are ordered according to the solution path (Definition 1.3). Together, the width and the vertical aspect of each box plot visually describe the variability of the corresponding estimated coefficient over the different data subsamples drawn.

- Shaded background representing τ_j: the background behind each box plot is shaded according to the relative same sign frequency τ_j of the corresponding covariate. The darker the colour, the higher the value of $\lfloor 100\%\tau_j/10 \rfloor$. The actual value of τ_j is displayed in black underneath the box plots.

- Lines showing the cut-off points for variable selection by the various versions of CSUV: CSUV-m (Definition 1.2) selects all covariates to the left of the solid vertical line. CSUV-s (Definition 1.4) selects all those to the left of the dotted vertical line.

Covariates with $\tau_j < 0.1$ are not included in the plot for readability. Users wishing to have a more detailed look into the empirical distribution of the non-zero estimated coefficients can superimpose the corresponding violin plots on the box plots in the `CSUV` package. See Figure 1.5 as an example of such a plot.

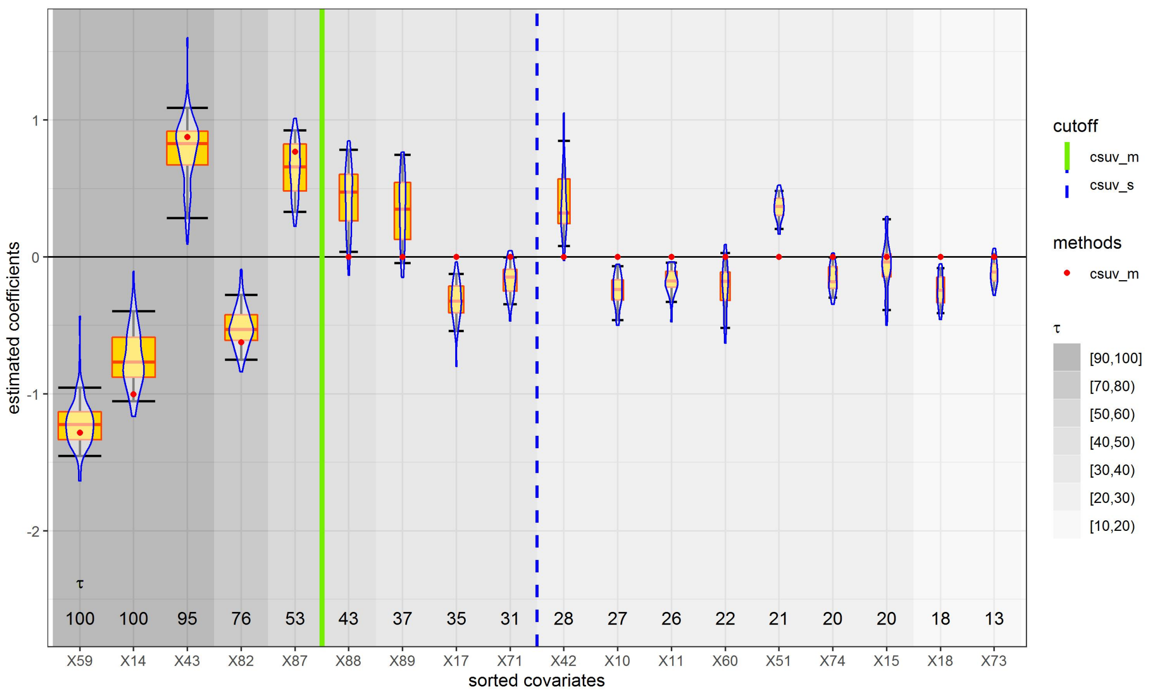

Fig. 1.5 Same as Figure 1.3 but with violin plots superimposed to show the conditional kernel density.

The default plot such as the one shown in Figures 1.3 and 1.5 only considers the empirical distributions of the estimated coefficients conditional on them being non-zero (we refer to them as “conditional box plots”). This is because box plots that use all the estimated coefficients in $\mathscr{M}$ in Step (8) of Algorithm 1.1 that are both the zero and non-zero ones ("unconditional box plots", see Figure 1.6 for an example) hardly provide useful information beyond that already provided in the value of τ_j, the latter also being reflected in the width of the conditional boxes. Nevertheless, the `CSUV` package allows users to create the unconditional box plots as well.

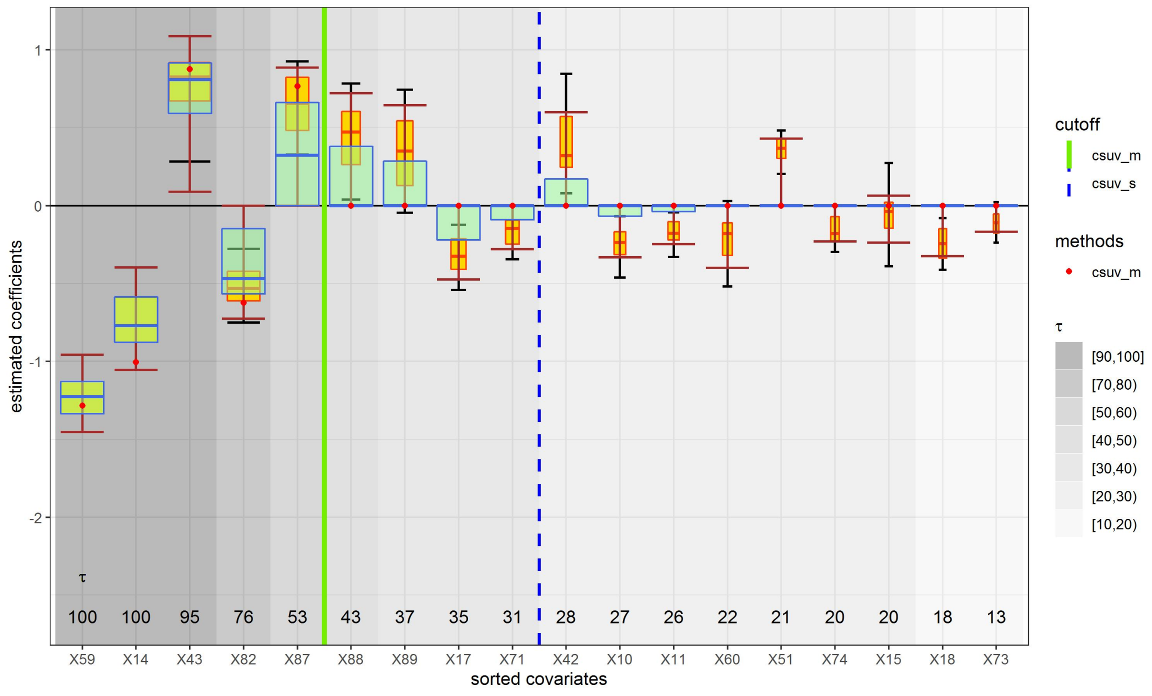

Fig. 1.6 Same as the box plot in Figure 1.3 but with the semi-transparent boxes (green in the colour version, usually they are wider than the conditional boxes underneath them) which represent all the estimated coefficients in $\mathscr{M}$ in Step (8) superimposed on top of it.

The `CSUV` package users wishing to compare the results returned by CSUV with any individual variable selection procedures of their choice (as long as their outputs are in a compatible format stated in the R package documentation) are also able to produce an enhanced CSUV plot, showing all of the above, and with addition of the items below.

- Graphical representation of the selection by a group of user-provided variable selection methods: the number (blue in the colour version) in the bottom part of the graph shows the percentage of user-provided methods that have selected the corresponding covariate when fitting with all the observations.

- Graphical representation of the selection by any single user-provided method: the coefficient estimates by the given method are shown as empty circles (white circles with a blue outline in the colour version).

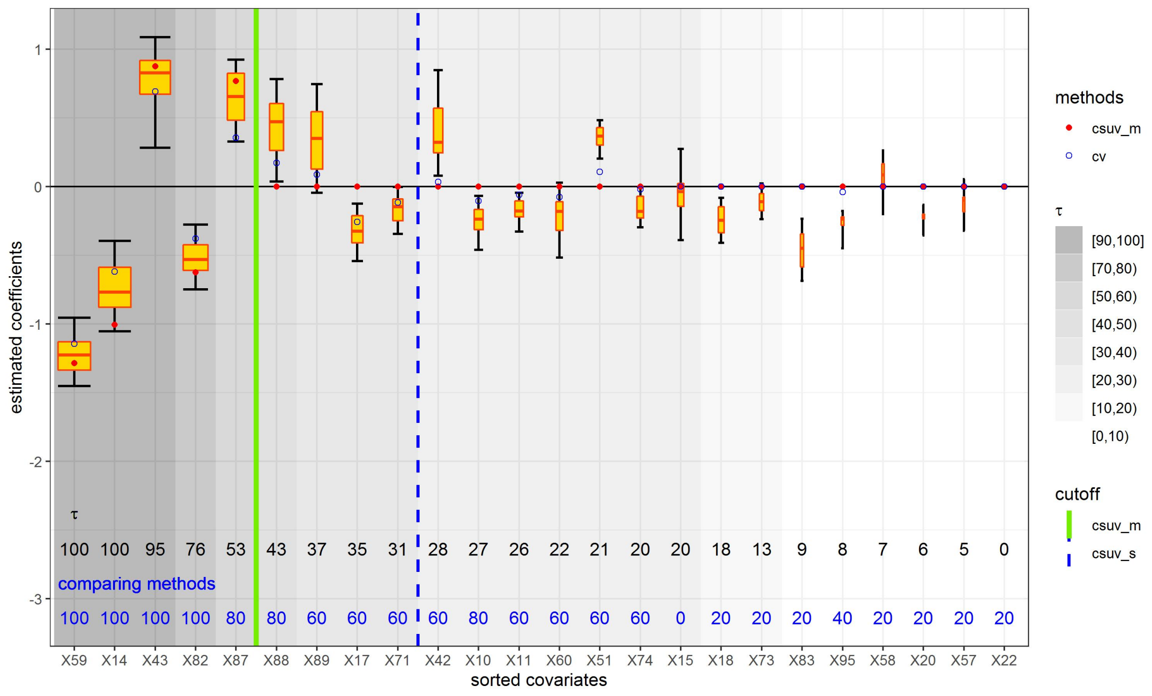

Fig. 1.7 Example of the CSUV graphical tool with additional information of the fitting results from five individual variable selection methods (Lasso, Elastic Net, relaxed Lasso, MCP and SCAD) and delete-$n/2$ cross-validation, using simulated data from model 2 parameter setting 5 (see Section 1.5.1.4 for more details on the simulation setting). The plot is the same as Figure 1.3 with the following extra information: Empty circles (white circles with blue outline in the colour version) represent the coefficients estimated by a single method (here is delete-$n/2$ cross-validation). Numbers at the bottom (blue in the colour version) represent the relative percentage proportion of the group of the individual methods that select the corresponding covariates. Covariates that are not selected by any methods and $\tau_j < 0.1$ are not shown for readability.

See Figure 1.7 for an example for such a plot. The user can decide if a covariate should be selected by considering if the corresponding CSUV box plot, the coefficient estimated by a single method and the percentage of selection by a group of comparing methods agree to some extent.

Note that it is common that CSUV and other model selection procedures agree to some extent. For example, in Figure 1.7, CSUV, cross-validation and all the individual variable selection methods select the first four covariates. The methods, however, have some disagreements over the other covariates. For example, the fifth covariate is selected by both versions of CSUV, cross-validation and 80% of the individual variable selection methods, but one of the individual variable selection methods does not select the covariate. The next ten covariates are selected by the majority of the individual methods and cross-validation, but they are not chosen by CSUV-m. These non-selection decisions taken by CSUV-m are correct, as in this particular simulation setting only the first five covariates have non-zero coefficients.

1.4.2 CSUV assessment of uncertainty

The CSUV plot provides a graphical tool to illustrate both the selection and the estimation uncertainty in the coefficients. The uncertainty illustrated by the CSUV plot should be interpreted to originate from the randomness of ε. This is similar to the classical confidence intervals in fixed-p, fixed-design regression.

In this section, our focus is on the uncertainty illustration by the default conditional boxes and whiskers, and on whether and how the information they carry can be used to assess the uncertainty in selection and estimation. Therefore our mentions of "boxes" or "whiskers" in this section refer to the conditional boxes and whiskers. Roughly speaking, the selection uncertainty is represented by the width of the boxes along the x-axis, and the estimation uncertainty is represented by the range of the boxes and whiskers along the y-axis. The plot provides a graphical aid to help users to decide whether to select a covariate by considering both dimensions of the corresponding box. The following similarities between the CSUV boxes and confidence intervals can be identified.

- Both provide intervals that likely cover the value of the true coefficient.
- Both aid the users in deciding if a covariate should be selected.

However, we also highlight the following differences between the two.

- *Information content.* Unlike the classical confidence interval, which is one-dimensional, the CSUV box is two-dimensional: both its width and its range should be used in deciding whether or not to include the corresponding covariate. This is because the ranges of CSUV boxes only contain information on non-zero estimated coefficients (i.e. any zero estimates for the coefficient are not reflected in the range of the box, but only in its width). For this reason, a covariate that is rarely chosen (and in particular, is not selected by CSUV-m) may have a box plot that does not cross 0. Therefore, the width of the box plot, which is directly proportional to the same-sign frequency with which the corresponding coefficient is selected, should also be considered in deciding whether or not to include the corresponding covariate in the model.
- *Covering percentiles.* The boxes in the CSUV plot represent the upper and the lower quartiles (i.e. 25% and 75% percentile) of the non-zero estimated coefficients. By contrast, classical confidence intervals are often considered in the context of much larger coverage; frequently, 90 or 95%. With this in mind, we set the whiskers in the box plots to describe the [5%, 95%] range (of the non-zero estimated coefficients) by default. This default range for the whiskers can be changed by users in the R package `CSUV`.

Moreover, the box plots are based on the individual empirical estimated coefficients, and do not take into account the effect of the selection uncertainty in other covariates. For example, if covariates X_1 and X_2 are highly correlated, whether X_2 is selected affects the estimated coefficients of X_1. While the conditional approach considered by Loftus and Taylor (2014) and Tibshirani et al. (2016), and the debiased approached considered by Zhang and Zhang (2014) in principle can be used here, the generalisation to CSUV is not straightforward and the conditional approach is computationally intensive.

The intertwining of the selection uncertainty and the estimation uncertainty makes it difficult to propose one simple interval that covers the true covariate with a given confidence level without a complicated adjustment e.g. as in Loftus and Taylor (2014) or Tibshirani et al. (2016). We instead restrict ourselves to investigating if the whiskers are useful in deciding if a covariate selected by CSUV-m should be chosen, without providing a confidence level guarantee.

Our investigation is as follows: using the simulated data from model settings 2-5 in Section 1.5.1.4, for covariates selected by CSUV-m, we want to find out if the covariates for which the whiskers cover zero are more likely to be the false covariates. For each realisation of the simulated data, we separate the CSUV-m selected covariates into two sets: (1) whiskers covering zero, and (2) whiskers not covering zero. We then find out the frequency with which the covariates in the two sets are the true covariates.

The simulation results show that a covariate with whiskers crossing zero is much more likely to be a false covariate than a covariate with whiskers not crossing zero (Figure 1.8). This indicates that observing if the whiskers of a covariate cross zero do provide useful information in deciding if the covariate is a true one.

1.5 Simulation study

In this section, we evaluate the performance of CSUV with numerical examples which consist of five simulated data settings and two real datasets. The main focus of our simulation is to compare the performance of CSUV with some model selection procedures including

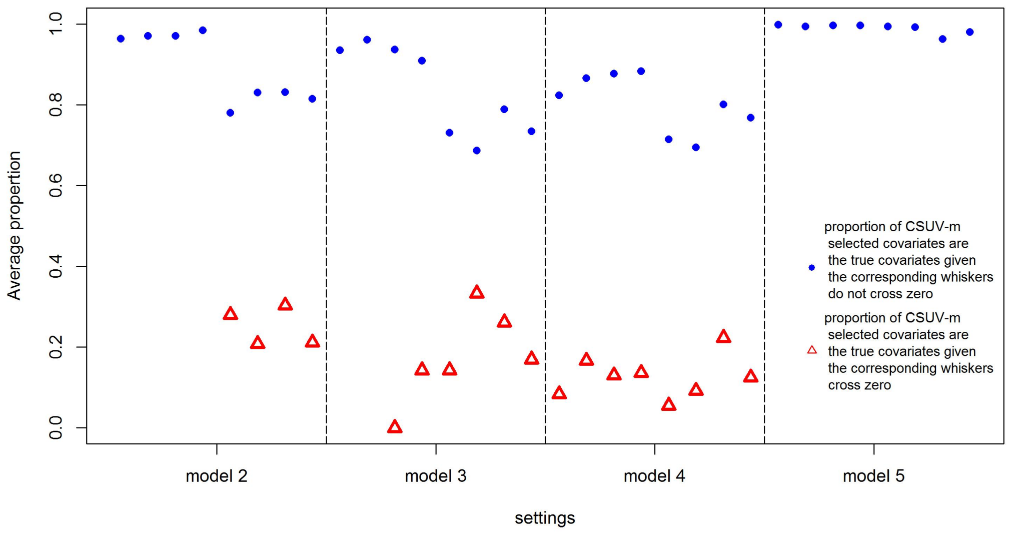

Fig. 1.8 Average proportion of the CSUV-m selected covariates are the true covariates, using simulated data from simulation model 2-5 with eight different parameter settings under each model setting (see Section 1.5.1.4 for more details on the simulation settings). Circles (blue in the colour version) show the average proportions of the CSUV-m selected covariates are the true covariates given the corresponding whiskers do not cross zero whereas the triangles (red in the colour version) show the average proportions of the CSUV-m selected covariates are the true covariates given the corresponding whiskers cross zero. If there is no triangle for a particular setting, it means that none of the CSUV-m selected covariates have whiskers crossing zero.

cross-validation and information criteria as they are popular approaches when there are different variable selection methods available. We also compare the performance of CSUV under different specifications (e.g. percentile parameter $q = 0$ vs $q = 5$, different constituent methods) to verify some claims we made in Section 1.3.3.

1.5.1 Simulation settings

1.5.1.1 R implementations

In the simulation, we consider CSUV with different sets of constituent methods:

1. Lasso, MCP and SCAD (default)

2. Lasso, Elastic Net, relaxed Lasso, MCP and SCAD

3. MCP

The first set is our primary interest. When we mention CSUV without specifying the corresponding constituent methods, we implicitly assume that this set of methods is used. The second combination is used to verify the claim that adding some similar methods does not affect the performance too much. The third set is used to verify the claim that using more constituent methods in general provides better results. We use MCP here because in the majority of the simulation settings it has the best variable selection performance among the individual variable selection methods in terms of the F-measure and the number of false classifications.

We use publicly available R packages for the implementation of the constituent methods (Lasso, Elastic Net, relaxed Lasso, MCP, SCAD) used in CSUV. See Table 1.1 for the list of the corresponding R packages, functions and parameter settings used in the `CSUV` package and also in this simulation. The concavity values of SCAD and MCP are set to the value recommended by the original papers from Fan and Li (2001) and Zhang et al. (2010) respectively, which are also the default values in the `ncvreg` R package. For Elastic Net, we use $\alpha = 0.5$.

Method	R package	R function	Parameters	λ tuning
Lasso (Tibshirani, 1996)	glmnet	cv.glmnet		default 10-fold cross-validation
Elastic Net (Zou and Hastie, 2005)	glmnet	cv.glmnet	α: 0.5	default 10-fold cross-validation
Relaxed Lasso (Meinshausen, 2007)	relaxo	cvrelaxo		default 5-fold cross-validation
SCAD (Fan and Li, 2001)	ncvreg	cv.ncvreg	concavity: 3.7	default 10-fold cross-validation
MCP (Zhang et al., 2010)	ncvreg	cv.ncvreg	concavity: 3	default 10-fold cross-validation

Table 1.1 Variable selection methods and the corresponding R packages and functions used in CSUV

1.5.1.2 Methods to compare

We use eBIC and delete-$n/2$ cross-validation as the major comparing methods to CSUV. We use eBIC instead of BIC as eBIC is designed for high-dimensional data. The details of the two methods are described in Section 1.2.1. We also include the simulation results of each constituent method (Lasso, Elastic Net, relaxed Lasso, MCP and SCAD), VSD-minimising method (Yang and Yang, 2017) and BIC for readers' reference.

Both eBIC and delete-$n/2$ cross-validation uses the Lasso, MCP and SCAD (i.e. the methods used in the default case of CSUV) as the base methods. eBIC selects the fitted model that minimises the corresponding information criterion value while delete-$n/2$ cross-validation selects the method that has the lowest estimated prediction error. The R packages and the parameter values used for the base methods are the same as what we use in CSUV for a fair comparison. All the variable selection methods require tuning the regularisation parameter λ. Default tuning in the R packages are used to simplify the analysis and the details of the tuning are shown in Table 1.1. eBIC and cross-validation have their own parameters and we set them as follow: For eBIC, we set $\gamma = 0.5$, which is one of the values considered in the simulations of the original paper (Chen and Chen, 2008) and the value used in Lim and Yu (2016). For the delete-$n/2$ cross-validation, we set the number of resampling $B = 100$, which is the same as the number of iterations we use in CSUV.

For the VSD-minimising method, we use the `glmvsd` R package to calculate the weight on each candidate model and then select the covariates that have an aggregate weight greater than or equal to 0.5. Coefficients of the selected set from VSD is estimated using OLS. We use the default parameters in `glmvsd` (e.g. use the Lasso, MCP and SCAD to get the candidate models) except the weight which we use ARM instead. This is because using the default BIC to calculate the weight provides very poor results in some simulation settings.

1.5.1.3 Performance measures

For the datasets for which we know the true sets of covariates (i.e. simulated data and the modified real dataset), we compare the variable selection performance among different methods by the F-measure, the number of false positives (FP), number of false negatives (FN) and the total number of variable selection error (FP+FN). The F-measure is the harmonic mean of precision and recall:

$$F = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2}{\frac{TP+FP}{TP} + \frac{TP+FN}{TP}} = \frac{2TP}{2TP + FN + FP}$$

Note that comparing the above numbers individually can be misleading. For example, using only FN favors models that select a large number of covariates and using only FP favors models that select fewer number of covariates. Although the F-measure takes both precision and recall into account, assigning same weight to precision and recall is arbitrary. Nevertheless, we use the F-measure as our major measure when we compare the variable selection performance between different methods. Powers (2011) provide a detailed comparison of different evaluation methods.

Although our main focus is variable selection performance, we also compute the prediction mean square errors (MSE) on test set data and the coefficient estimation error (l_1 and l_2) for CSUV and the comparing methods.

1.5.1.4 Synthetic data

Set $\boldsymbol{Y} = \tilde{\boldsymbol{X}}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, $\varepsilon_i \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$. We generate observations with 100 realisations of $\boldsymbol{X}$ using the model specifications below. We then normalise $\boldsymbol{X}$ to get $\tilde{\boldsymbol{X}}$ so that all covariates have mean 0 and variance 1. Except from Model 1, the number of observation is $n = 100$, the number of predictors $p = \{100, 300\}$, the number of true covariates $s = \{5, 10\}$ and $\sigma^2 = 1$.

- **(Model 1) modified example 1 from the original Lasso (Tibshirani, 1996): $\boldsymbol{\beta}$** $=\{3,1.5,0,0,2,0,0,0\}$, $p=8$ and $n=50$. Predictors $\boldsymbol{X}$ follow $\mathcal{N}(0,\Sigma)$, where $\Sigma_{k,m}=0.5^{|k-m|}$ and $\sigma=\{1,3,6\}$. In the Lasso paper $n=20$ but here we use $n=50$ so that there are enough observations for subsampled fit. We include a more challenging SNR with $\sigma=6$ ($\sigma=3$ in the Lasso paper).

- **(Model 2) Toeplitz structure:** predictors $\boldsymbol{X}$ follow $\mathcal{N}(0,\Sigma)$, where Σ is in Toeplitz structure with $\Sigma_{k,m}=\rho^{|k-m|}$ with $\rho=\{0,0.9\}$.

- **(Model 3) block structure:** predictors $\boldsymbol{X}$ follow $\mathcal{N}(0,\Sigma)$, where Σ is in block structure with $\Sigma_{k,m}=1$ for $k=m$. For $k\neq m$, $\Sigma_{k,m}=0$ except $mod_{10}(m)=mod_{10}(k)$ which $\Sigma_{k,m}=\{0.5,0.9\}$.

- **(Model 4) factor model:** latent covariates $\phi_j, j=1,...,J$ are i.i.d. and follow $\mathcal{N}(0,1)$. Each covariate is generated by $X_k=\sum_{j=1}^{J} f_{k,j}\phi_j+\eta_k$, where $f_{k,j}$, η_k are i.i.d. and follow $\mathcal{N}(0,1)$. The number of factor $J=\{2,10\}$.

- **(Model 5) modified example from Zhang and Yang (2015):** $\beta_j=6/j$ for the true covariates $j=1,...,s$ and $\beta_j=0$ otherwise. Predictors $\boldsymbol{X}$ follow $\mathcal{N}(0,\Sigma)$, where $\Sigma_{k,m}=\rho^{|k-m|}, \rho=\{0.5,-0.5\}$. The difference between Zhang and Yang (2015) and the model 5 here is that we use the same n and p as model 2-4.

For models 2-4, $\lfloor\frac{s}{2}\rfloor$ of the coefficient of the true s are chosen randomly from $U(0.5,1.5)$ and $\lceil\frac{s}{2}\rceil$ of them are chosen uniformly from $U(-1.5,-0.5)$. The true βs are chosen randomly among the predictors, and once the βs are set, the same set of βs are used for all realisations.

1.5.1.5 Real datasets

Example 1.3 (Boston housing data, Harrison Jr and Rubinfeld, 1978)**.** The dataset consists of the median value of owner-occupied homes as response and $p=13$ covariates (crime rate, proportion of residential land, etc). Number of observations is $n=506$. The dataset is publicly available in R with the `MASS` package. For each simulation, half of the observations are used as the training data and the other half are used as the test set.

Example 1.4 (Modified riboflavin data)**.** Here we re-examine the riboflavin dataset introduced in Example 1.1. In order to assess the variable selection performance, we randomly permute all but 10 of the 4088 covariates in the riboflavin dataset across all the observations. The same permutation is used for all permuted covariates to keep the original dependence structure among them. The set of 10 unpermuted covariates is chosen randomly among the 200 covariates with the highest marginal correlation with the response.

The modification for the riboflavin dataset ensures that the permuted covariates cannot be the true covariates in this modified dataset. In the simulation results, we refer the 10 unpermuted covariates as the "true" covariates, although in reality they may not be the true covariates.

For the Boston data, we repeat the process for $m = 100$ times with random cuttings of the training and the test data. For the riboflavin data, we repeat the process for $m = 100$ with a random selection of the 10 unpermuted covariates to stabilise the results.

1.5.2 Simulation results

The simulation results are summarised in Table A.1-A.15 in the Appendix A.4. Below we discuss the simulation results in detail.

1.5.2.1 Verification of claims made in Section 1.3

In Section 1.3, we claim that:

- CSUV-m is designed for variable selection whereas CSUV-s is designed for better prediction.
- Performance of CSUV should be similar as long as q is small (e.g. $q = 0$ or $q = 5$).

- Including more (diverse) methods should improve the performance of CSUV.
- Including some similar methods should not worsen the performance of CSUV by much.

The simulation results support the claims above:

- CSUV-m vs CSUV-s: In general CSUV-m has better variable selection performance in terms of the F-measure. CSUV-s usually has a better prediction performance, and it also has a more stable (not too far off from the best method when CSUV is not performing particularly well) prediction performance in terms of MSE than CSUV-m. This may because CSUV-s selects a larger set of covariates than CSUV-m.
- $q = 0$ vs $q = 5$: the performance of CSUV-m when $q = 0$ and $q = 5$ is quite similar in terms of the number of covariates selected, and the prediction and variable selection performance, although $q = 0$ performs slightly better than $q = 5$.
- MCP only vs three different methods: here we only consider $q = 0$ as by using $q = 0$ we do not need to worry about the difference in terms of the number of fitted models selected (with $q = 5$ for example, the number of fitted models from three variable selection methods are around three times of the number of fitted models from a single method). In our simulation, CSUV using MCP only in general has worse performance than CSUV using three different constituent methods. In some other cases like the model 3 with parameter setting 7 and 8, both the prediction and variable selection performance of CSUV using MCP only is much worse than CSUV using three different constituent methods.
- Including some similar methods: here again we only consider $q = 0$. The results of CSUV using three different constituent methods (Lasso, MCP and SCAD) and five different methods (Lasso, Elastic Net, relaxed Lasso, MCP and SCAD, for which the Lasso, Elastic Net and relaxed Lasso are relatively similar) are very similar.

1.5.2.2 Comparing the performance between CSUV and some existing final model selection procedures

In the majority of settings, CSUV-m has a better variable selection performance than the eBIC, delete-$n/2$ cross-validation and VSD-minimising method in terms of the total number of variable selection error and the F-measure, and a better coefficient estimation performance in terms of the l_1 loss. For example, out of the 36 simulation settings that we know the true set of covariates (i.e. the simulated data and the modified riboflavin dataset), CSUV-m has a higher F-measure on 33 of the settings when comparing with the delete-$n/2$ cross-validation and 32 of the settings when comparing with eBIC. CSUV-m also has higher F-measure than VSD-minimising method in 23 settings. CSUV-m usually selects the smallest set of covariates when comparing with eBIC or delete-$n/2$ cross-validation and the individual variable selection methods. In some cases like model 4 parameter setting 6, it selects a much smaller set of covariates than the truth. While this worsens the prediction performance of CSUV-m and we may view it as a limitation of CSUV-m, it may well due to the limitation of variable selection as a whole: Other methods which select much larger sets of covariates usually include a few more true covariates but inevitably they also include many more false covariates. They may perform better than CSUV-m in terms of prediction, but CSUV-m in general outperforms them in terms of variable selection.

The performance of CSUV-s, on the other hands, is much more difficult to draw conclusions on. CSUV-s is better than delete-$n/2$ cross-validation in terms of variable selection. When comparing with eBIC, while it performs better than eBIC in one measure in some settings, it performs worse than eBIC in some other settings with the same measure.

One encouraging result about CSUV is that in many simulation settings like model 2, CSUV-m outperforms not only the final model selections procedures but it also outperforms *all* individual constituent methods in terms of the F-measure and the total number of variable selection error. In some simulation settings, CSUV performs better than the best individual variable selection method in terms of both prediction and variable selection measured by F-measure. For example in model 2, there are quite a few parameter settings (e.g. parameter setting 2) that the MSE of CSUV is lower and the F-measure is higher than all individual variable selection methods.

For the variable selection performance on the real data, both versions of CSUV perform very well on the riboflavin data example. CSUV-s has the best performance in terms of F-measure and the total number of variable selection error.

1.5.3 Analysis of the selection by CSUV

1.5.3.1 Reasons for the selected set to be small for CSUV-m

The number of covariates selected by CSUV-m is often small when compared with other methods and the true size. An investigation into the collection of fitted models $\mathscr{M}$ shows that for many simulation settings, the fitted models in $\mathscr{M}$ can be very different in terms of variable selection. Sometimes all fitted models in $\mathscr{M}$ select different sets of covariates. When the selection decision is so different among $\mathscr{M}$, it is very likely that only a few covariates will have $\tau_j \geq 1/2$. This causes the number of covariates chosen by CSUV-m to be small. Whether a small selected set is desirable depends on the purpose of variable selection. Selecting small(er) number of covariates by this selection rule may cause omission of some true covariates and possibly exclusion of some false covariates that are helpful for prediction. This may result in poor prediction in some situations. On the other hand, the set of covariates selected by CSUV-m often includes fewer false positives than other variable selection methods, as only covariates that are selected by the majority of the subsampled fits are included in CSUV-m.

1.6 Conclusion

Many variable selection methods are available. However, there is no clear guideline on how to select which method to use with the data at hand, or how we can trust the set of covariates selected by a method. In practice, cross-validation and information criteria may be used to select the final models: Zhang and Yang (2015) advocate to use the delete-$n/2$ cross-validation and Chen and Chen (2008) extend the use of BIC to high-dimensional data (eBIC).

In this chapter we suggest a competitive alternative to these two procedures. We also provide a graphical illustration of the selection uncertainties. CSUV does not attempt to select the best method or to find the optimal regularisation parameter. Instead we aggregate the fitted results from different variable selection methods via subsampling, and use a graphical tool to illustrate the uncertainties in selection and estimation. CSUV is very general and can be used with different variable selection methods. The simulation results show that CSUV in general outperforms the delete-$n/2$ cross-validation and eBIC in terms of variable selection. We also show that the graphical tool of CSUV has the capability to aid analysts in variable selection.

Chapter 2

Time series model identification: Exploring the estimation-simulation-classification approach

2.1 Introduction

Assume we only have one realisation of a time series and the time series is either from a short memory change-point or a long memory process. Our objective is to find the correct model specification for the given time series, i.e. to be able to tell if the time series is from a short memory change-point or a long memory process.

A time series is said to have long memory if its autocorrelation function decays slowly and is not absolutely summable. On the contrary, a short memory time series has the autocorrelation function decaying at a faster rate (e.g. exponentially) and is absolutely summable. When there are change-points in the mean level, however, even if each segment between two consecutive change-points is short memory, the sample autocorrelations of such time series can decay slowly and do not converge to zero at the exponential rate (Yau and Davis, 2012). In literature, there are a number of works showing theoretically and via simulation that it is difficult to distinguish between the two models, for example, Diebold and Inoue (2001), Granger and Hyung (2004), etc.

Figure 2.1 show a simulated short memory change-point time series (top left) and a long memory time series (top right) with their corresponding sample autocorrelations (bottom). The left one has mean shift 1 and ARMA parameters change from $\phi_1 = 0.1$ and $\theta_1 = 0.3$ to $\phi_2 = 0.4$ and $\theta_2 = 0.2$. The right one is from the long memory autoregressive fractional integrated moving average (ARFIMA) with $\phi = 0.1$, $\theta = -0.8$ and $d = 0.3$. The two time series are the realisations from our simulation settings (Section 2.5 setting 1 and setting 13). It is difficult to tell which one is long memory and which one has change-points(s) just by looking at the graphs, but the methods we suggested later in this chapter are able to identify the models correctly.

In economics and financial time series, the slow decay in the sample autocorrelation or persistence in the long-run effect of a shock is often observed. For example, Granger and Ding (1995) show that the sample autocorrelations decay slowly for the absolute return of S&P 500, Greene and Fielitz (1977) show that the stock returns are characterised by the long-range dependence and Henry (2002) use semiparametric approaches to argue that there is some evidence of long-range dependence in the stock returns in different stock markets. Pivetta and Reis (2007) find that there is high persistence in inflation in the US from 1965. Figure 2.2 shows the percentage change in the Consumer Price Index (CPI) in the United States and Figure 2.3 shows the autocorrelations of it. The sample autocorrelations decay slowly, which give the impression that the time series may have long memory.

In literature, both long memory and change-point models have been used to model time series with observed long memory and persistence. For inflation data, Hassler and

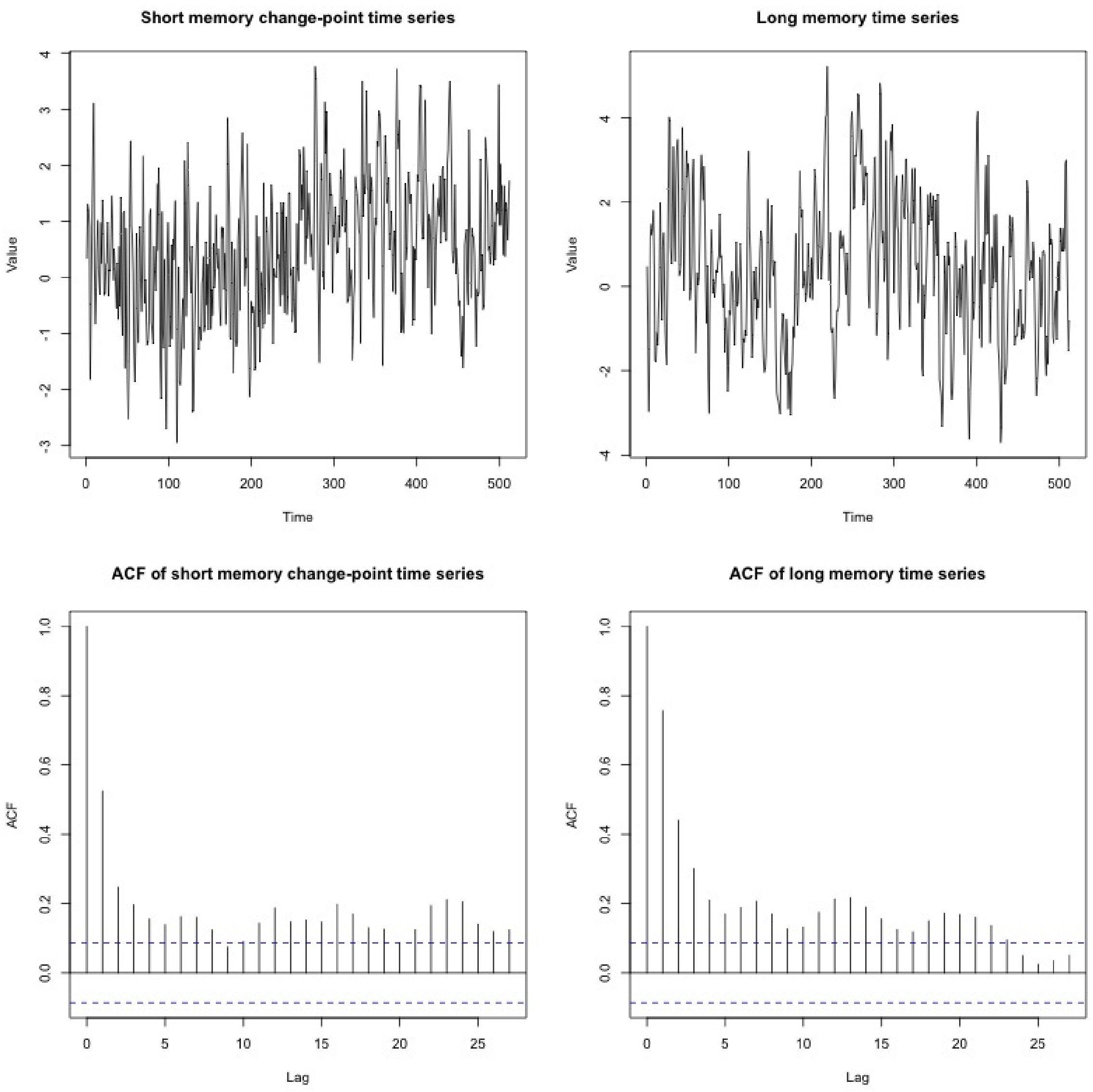

Fig. 2.1 A short memory change-point time series (top left) and a long memory time series (top right) with their corresponding sample autocorrelations(bottom left and bottom right).

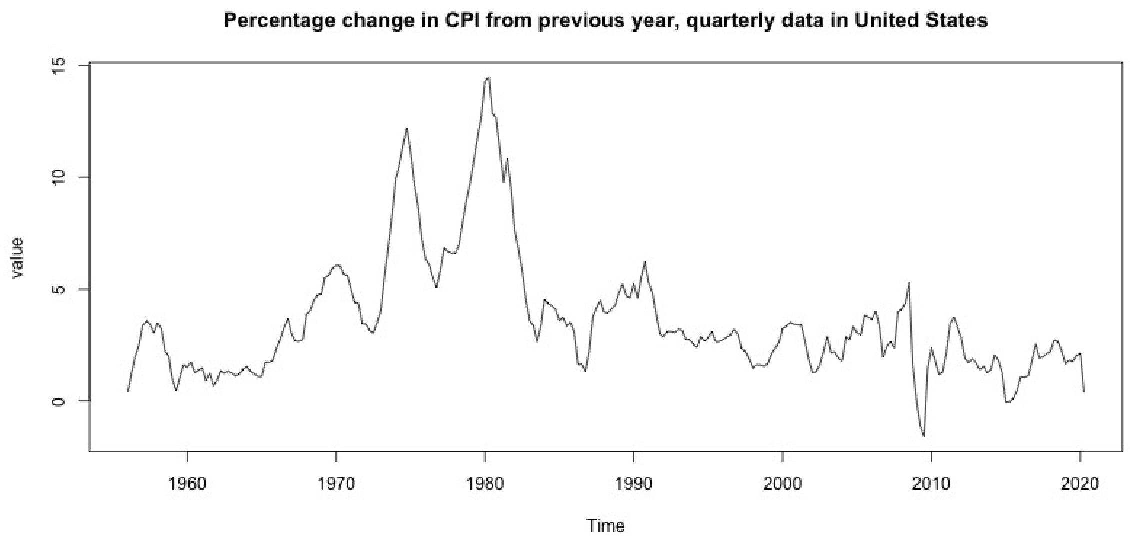

Fig. 2.2 Percentage change in the Consumer Price Index (CPI) from previous year (United States quarterly data). Data retrieved from International Monetary Fund (IMF) via DBnomics.

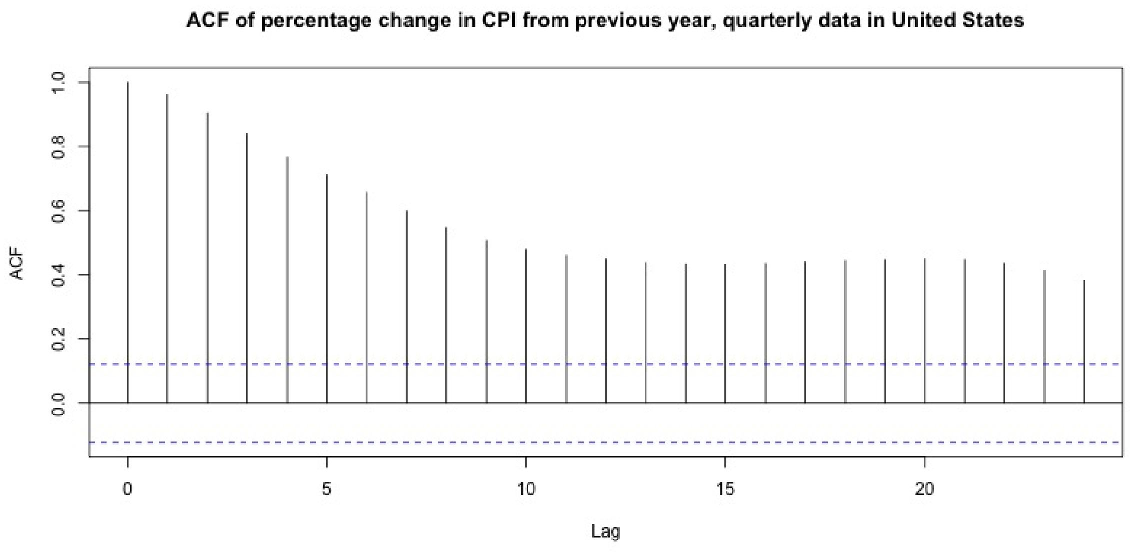

Fig. 2.3 Autocorrelations on percentage change in the Consumer Price Index (CPI) from previous year (United States quarterly data).

Wolters (1995) use a long memory model autoregressive fractional integrated moving average (ARFIMA) whereas Levin and Piger (2002) use a model with structural break in mean (intercept) to model the inflation time series.

The use of long memory or short memory change-point model can provide a very different interpretation for the observed data. The short memory change-point model indicates there are changes in the structure due to a few shocks in the time series, but the long memory model indicates an equal persistence to all shocks. For inflation data, Levin and Piger (2002) argue that the observed persistence in inflation can be explained by the change-point model, with the occasional shifts in the monetary policy regime. This explanation is different from the one provided from the long memory model, where the observed long memory is treated as an inherent characteristic of industrial economies.

The existing statistics literature on distinguishing the two models mainly focuses on using hypothesis testing (for example Yau and Davis, 2012 and Berkes et al., 2006). In their settings, the short memory change-point model is treated as the null hypothesis and the long memory process is set as the alternative. Norwood and Killick (2018) argue in some situations it is difficult to justify the use of the short memory change-point model as the null model. They instead propose a classification approach that chooses the model specification that is more similar to the observed time series in terms of the wavelet spectrum. In this chapter we consider the classification approach, as we do not have a justified null.

In recent years, deep learning methods (LeCun et al., 2015) have been very successful in performing classification tasks such as time series (Wang et al., 2017, Fawaz et al., 2019) and image classification. Intriguingly, Fawaz et al. (2019) show that some deep neural networks such as the residual nets (ResNet, He et al., 2016) perform very well in classifying time series even when the number of samples is small. This makes us wonder if these state-of-the-art classifiers can be used to select the best model for the observed data. The extension of the application of the deep learning classifiers to model identification is not straightforward -

classifiers, especially the deep learning classifiers, require a lot of observations to train but we only have one observed time series at hand.

One idea to extend the use of classifier to model identification is that for two potential model specifications, we generate a large number of time series that are "similar enough" to the observed time series we have at hand, and use these simulated time series to train the classifier. Once we have the trained classifier, we can use it to classify the original time series and inform us which model specification is better for the observed time series. We refer it as the "estimation-simulation-classification" approach or the "ESC" approach in this thesis for easy referencing.

Such an approach is used in Norwood and Killick (2018) in distinguishing the short memory change-point from the long memory models. In order to explore the potential of the ESC approach, we study the methodology and the simulation from Norwood and Killick (2018). We follow the simulation settings from Norwood and Killick (2018) and focus on classifying time series between the long memory model in the form of ARFIMA and the short memory model with one change-point (with ARMA in each segment). We observe that if the model complexity is not fully taken into account, the ESC approach may favour models that are more flexible, which in their case is the short memory change-point model.

With this observation, we suggest using information criteria for selecting the model given the choice of the short memory change-point and the long memory models. Information criteria are often used in model selection. In some empirical works on economics data like Song and Shin (2015), BIC is also used to select between the short memory change-point and the long memory models. From our best knowledge, however, there is no formal discussion of using them in distinguishing the long memory and the short memory change-point models in the statistics literature. We show that under the simulation settings considered by Norwood and Killick (2018), model specification selected by the Bayesian information criterion (BIC or Schwarz information criterion SIC, Schwarz et al., 1978) considered by Yao (1988) and some other versions of BIC outperform the method proposed by Norwood and Killick (2018) in nearly all settings. While we do not intend to suggest that BIC is the best for model identification in general, they should at least be considered as the base case.

Finally, we propose a procedure using ResNet for model identification via the ESC approach. While the poor model identification results from Norwood and Killick (2018) is discouraging, it does not mean the ESC approach is always unfeasible. The impressive performance of ResNet on time series classification may compensate the issue with the ESC approach and the proposed method may provide a reasonable model identification performance.

Before we continue, we would like to highlight some works in literature which use simulated training data to improve the classification performance. For example, Sobie et al. (2018) and Murphey et al. (2006) use simulated data to enhance the performance for fault detection. The state-of-the-art image recognition neural networks like the ResNets use data augmentation like flipping the images to increase the number of training samples. Generative Adversarial Networks (GANs, Mirza and Osindero, 2014) also can be used to simulate training data. These methods, however, either have a given model to generate the simulated data and thus does not need to estimate the model from the given data (e.g. flipping and chopping images for training a deep neural network), or there is a relatively large training set available to train the model generating the simulated samples (e.g. GANs). Neither the model for simulation nor a relatively large amount of training data is available in the situation we consider.

Another topic related to this chapter is approximate models (Davies, 2014). The main idea of approximation from Davies (2014) and the related works from the same author is that there is no so-called “true” model for the observed data we have at hand. Instead, we accept that at best we can have some “approximate” models, which approximate the given observed data well enough, if the typical data sets generated under the model is similar to the observed data. While approximate models have similar ideas about having “typical data” “look like” the observed data, there is a fundamental difference. In Davies (2014), they assume there is no true model, whereas for model identification / classification we assume there is a true model.

The rest of the chapter is organised as follows. In Section 2.2 we review the related literature. In Section 2.3 we study the method from Norwood and Killick (2018), consider the possible problems with the approach and outline our approach to study the problems further. In Section 2.4 we suggest using BIC and propose a procedure to use ResNet for model identification via the ESC approach. In Section 2.5 we verify the claims we made in Section 2.3 and compare the model identification performance of the proposed method from Norwood and Killick (2018) and our proposed methods using the simulated and real data. We conclude the chapter with a discussion in Section 2.6.

2.2 Literature review

2.2.1 Time series analysis

We consider univariate discrete time series in this thesis. Time series is a set of ordered observations recorded at a specific time:

$$\{x_t | t \in T_0\} \tag{2.1}$$

where T_0 is a set of time points. In this chapter our time series in the form of:

$$\{x_1, ..., x_T\} \tag{2.2}$$

where T is the number of observations.

Time series naturally arise in different areas like economics (e.g. GDP, unemployment rate, balance of payment, money supply), finance (e.g. stock prices, P/E ratio, bound yield, commodity prices), social science (e.g. population, crime rate), environment (e.g. temperature, pollution level), medical data (e.g. EEG, MRI, heartbeat), etc.

To account for the uncertainty in the future observations, it is natural to suppose that x_t is a realisation value of a certain random variable X_t and the time series $\{x_t|t \in T_0\}$ is a realisation of the family of random variables $\{X_t|t \in T_0\}$ or a stochastic process $\{X_t|t \in \mathbb{Z}\}$. In this thesis, the term time series is used to refer to both the data and the process of which it is a realisation of.

One of the important concepts in time series is stationarity. Intuitively speaking, the stationarity of a time series guarantees some important statistical properties of the time series to be the same over time, and this enables us to estimate parameters consistently. A time series $\{X_t|t \in \mathbb{Z}\}$ is strictly stationary if the joint distribution is the same, i.e. :

$$\{X_{t_1}, ...X_{t_n}\} \overset{D}{=} \{X_{t_{t_1+h}}, ..., X_{t_n+h}\}$$

for all $t_1, ..., t_n$ and $h \in \mathbb{Z}$.

Strictly stationary is often too restrictive, and weakly stationary is used instead. A time series is weakly stationary if:

- $E|X_t|^2 < \infty$
- $EX_t = m$ for all $t \in \mathbb{Z}$
- $\gamma_X(s,t) = \gamma_X(s+h, t+h)$ for all $s,t,h \in \mathbb{Z}$

where $\gamma_X(s,t) = Cov(X_s, X_t)$ is the autocovariance function of $\{X_T\}$. For weakly stationary process, the autocovariance function can be rewritten as a function of just one variable:

$$\gamma_X(h) \equiv \gamma_X(h, 0)$$

In this thesis, the word “stationary” refers to weakly stationary. Below we introduce the autoregressive moving average (ARMA), which is one of the most commonly used stationary time series models.

Autoregressive moving average process $ARMA(p,q)$

The process $\{X_t\}$ is said to be an $ARMA(p,q)$ process if $\{X_T\}$ is stationary and for every t,

$$X_t - \phi_1 X_{t-1} - ... - \phi_p X_{t-p} = Z_t + \phi_1 Z_{t-1} + ... + \phi_q Z_{t-q} \tag{2.3}$$

where $\{Z_t\} \sim WN(0,\sigma^2)$.

The family of ARMA processes are important, as for any $\gamma(\cdot)$ such that $\lim_{h\to\infty}\gamma(h) = 0$, it is possible to find an ARMA process $\{X_t\}$ such that $\gamma_X(h) = \gamma(h)$, $h = 1,...,k$ for any positive integer k. ARMA thus has the capacity of describing and closely approximating many different stationary time series.

A stationary and invertible ARMA time series has its autocorrelations to be geometrically bounded

$$|\rho_X(h)| \leq cm^{-k}$$

for large k and with $0 < m < 1$ and c is some positive constant, where $\rho_X(h) = \frac{\gamma_X(h)}{\gamma_X(0)}$. Its sum of autocorrelations:

$$\lim_{n\to\infty} \sum_{h=-n}^{n} |\rho_X(h)|$$

is finite (Baillie, 1996). ARMA thus has short memory, as opposed to the long memory we review in the next section.

2.2.2 Long memory models

Long memory processes are characterised by the slow decays in the autocorrelation. Let $\{X_t\}$ be a stationary time series with autocorrelation function $\rho_X(h)$. Following from McLeod and

Hipel (1978), a time series has long memory if:

$$\sum_{h=-n}^{n} |\rho_X(h)|$$

diverges as $n \to \infty$. Equivalently, the spectral density of a time series with long memory is unbounded at low frequency.

Long memory also defines as follows in literature. Let $H \in (1/2, 1)$ and c be a positive constant. If the autocorrelation function of $\{X_t\}$ is

$$\rho_X(h) \sim ch^{2H-2} \tag{2.4}$$

Then $\{X_t\}$ is a long memory process. H is called Hurst parameter. Often in literature the parameter

$$d := H - 1/2, d \in (0, 1/2) \tag{2.5}$$

is used instead of H. The autocorrelation function of a long memory process is decaying at a polynomial rate. The dependence between the present and all past data is not negligible.

Long memory model is used to model persistent shock in economics and financial data. For example, Greene and Fielitz (1977) show that stock returns are characterised by long-range dependence. Henry and Zaffaroni (2003) provide an overview of long memory observed and the long memory model applications in macroeconomic and financial time series. Bai (1997) provide a detailed review of the use of long memory models in econometrics.

Below we introduce the autoregressive fractional integrated moving average (ARFIMA, Hosking, 1981 and Granger and Joyeux, 1980), which is one of the most commonly used stationary time series models.

Autoregressive fractional integrated moving average (ARFIMA)

$ARFIMA(p,d,q)$ is defined as follows:

$$\Phi(B)(1-B)^d X_t = \Theta(B)\varepsilon_t \tag{2.6}$$

where B is the backward-shift operator, $\Phi(L) = 1 - \phi B - ... - \phi^p B^p$, $\Theta(L) = 1 + \theta B + ... + \theta^q B^q$, and $d \in \mathbb{R}$. When $d = 0$, it reduces to an ARMA model. When $d \in (-0.5, 0.5)$ ARFIMA is stationary, and when $d > 0$ ARFIMA has long memory. In this chapter, we follow Norwood and Killick (2018) and focus on ARFIMA with $d \in (0, 0.5)$ so that the corresponding time series are stationary and have long memory.

ARFIMA is widely used to model time series data exhibiting an apparent long memory. For example, ARFIMA is used to model inflation data in Hassler and Wolters (1995), stock index return in Barkoulas et al. (2000) and air quality in Pan and Chen (2008).

2.2.3 Change-point models

Change-point (or structural break) is a change in the parameters or statistical properties of the underlying model of a time series $\{X_t\}$. The properties and parameters of the model of a time series in a segment between two consecutive change-points are the same but between the consecutive segments the parameters or the properties are different in some ways. In this thesis, the location of the change-point is denoted as τ if there is only one change-points, or $\tau_1, ..., \tau_m$ if there are $m > 1$ change-points. Change-point models are of interests in different areas like finance (e.g. Andreou and Ghysels, 2009, Andreou and Ghysels, 2002, Schröder and Fryzlewicz, 2013), climate data (e.g. Reeves et al., 2007, Wang et al., 2007), bioinformation (e.g. Muggeo and Adelfio, 2011, Braun et al., 2000) and neuroscience (e.g. see Koepcke et al., 2016).

In this chapter, we follow the simulation settings from Norwood and Killick (2018) and we focus on time series with one change-point in an ARMA process:

$$X_t \sim \begin{cases} \mu_1 + ARMA(\phi_1, \theta_1), \text{ if } t = 1, 2, ..., \tau \\ \mu_2 + ARMA(\phi_2, \theta_2), \text{ if } t = \tau + 1, \tau + 2, ..., T \end{cases} \tag{2.7}$$

where ARMA is an autoregressive moving average model with ϕ is an autoregressive (AR) parameter, θ is a moving average (MA) parameter, τ is the location of the change-point and T is the number of observations. Some parameters in the first segments $(\mu_1, \phi_1, \theta_1)$ must be different from the parameters in the second segments $(\mu_2, \phi_2, \theta_2)$. The ARMA in each segment is stationary. Similar model is used in literature for economics data. For example, Levin and Piger (2002) use AR model with change in parameter values to model inflation data and Hyung and Franses (2001) consider using model with ARMA noise and change in mean to model inflation data for forecasting purpose.

2.2.4 Change-point detection algorithms

Change-point detection algorithms often estimate both the number of change-points m and the location of the change-points $\tau_1, ..., \tau_m$. They can be used to facilitate the time series modelling for interpretation and to have a better understanding of the time series process. They can also help to provide better prediction by discarding the data points before the change-point which may no longer be relevant for future observations (e.g. Pesaran and Timmermann, 2004).

Change-point detection algorithms can either be offline or online. Offline change-point detection algorithms are retrospective and analyse all the past observations at once. They usually aim to identify all the past change-points to model the time series. Online detection algorithms process each data point as soon as it is observed. They aim to detect change-points in real-time and detect the new change-point as soon as it has occurred (Aminikhanghahi and Cook, 2017). In this thesis we focus on the offline change-point detection algorithms.

While the time series we are focusing on Equation (2.7) has only one change-point, and the number of change-point m is given (i.e. $m = 1$), the multiple change-point selection method Narrowest-Over-Threshold (NOT, Baranowski et al., 2019) is used to study the relation between the model complexity and the performance of the "ESC" approach in the simulation in Section 2.5. Also, in Chapter 3 we consider time series that may have multiple change-points and propose a new procedure based on the multiple change-point detection method NOT as well. Therefore, in this section we will review the change-point detection algorithms for both single change-point and multiple change-points.

Single change-point detection algorithms

For time series with at most one change-point, the detection (and finding the location) of the change-point is often posed as a hypothesis test (Killick and Eckley, 2014), with the null hypothesis corresponds to no change-point whereas the alternative hypothesis corresponds to one single change-point. For example, Hinkley (1970) use a likelihood ratio approach to detect a change-point in mean. They derive the asymptotic distribution of the likelihood ratio statistic for a change in normal mean. Hawkins (1977) also use a likelihood ratio test to test for change in mean.

In the context of likelihood ratio test, under the alternative hypothesis, we select the parameters to maximise the log-likelihood. The location of the potential change-point is also selected to maximise the log-likelihood:

$$\hat{\tau} = \arg\max_{\tau} \log\ L(\tau) \tag{2.8}$$

which

$$L(\tau) = \log p(x_1, ..., x_\tau; \hat{\theta}_1) + \log p(x_{\tau+1}, ..., x_T; \hat{\theta}_2) \tag{2.9}$$

where $\hat{\theta}_1 \neq \hat{\theta}_2$ are the estimated parameters from the maximum likelihood estimation, $p(x_s,...,x_t|\theta)$ is the probability density function corresponding to $x_s,...,x_t$ given the parameter θ. If a null hypothesis corresponding to no change-point is rejected, then the single change-point is estimated as $\hat{\tau}$ from Equation (2.8).

For the change in mean with i.i.d normal noise, finding the location of the change-point through maximising the likelihood is equivalent to maximising the absolute CUSUM (cumulative sum) statistics (Baranowski et al., 2019). The CUSUM statistics $\mathscr{C}^{(b)}$ is defined as the inner product between the time series and a particular 'contrast' weight which is constant on each segment divided by the point b:

$$\mathscr{C}^{(b)} = \sqrt{\frac{T-b}{Tb}} \sum_{t=1}^{b} x_t - \sqrt{\frac{b}{T(T-b)}} \sum_{t=b+1}^{T} x_t \tag{2.10}$$

The change-point τ is estimated as:

$$\hat{\tau} = \arg\max_{b} |\mathscr{C}^{(b)}| \tag{2.11}$$

Multiple change-point detection algorithms

Multiple change-point detection algorithms estimate the location of change-points, and the number of change-points if it is not known. When the number of change-points are unknown, penalty based methods may be used to prevent overfitting and they find the change-points to minimise the function in the form of:

$$\sum_{i=1}^{m+1} L(X_{\tau_{i-1}+1},...,X_{\tau_i}) + pen(m,\tau_1,...,\tau_m) \tag{2.12}$$

where m is the number of change-points, $\tau_0 = 0$, $\tau_{m+1} = T$, T is the length of the time series, L is a loss or cost function for a segment and $pen(m,\tau_1,...,\tau_m)$ is a penalty function on the number and the location of the change-points.

In literature, Yao (1988) and Yao and Au (1989) consider the least square estimator for the location of change-points in mean and both non-penalised (for known m) and penalised least squares approaches (for unknown m) are considered. Yao and Au (1989) show that BIC is consistent for i.i.d. noise with the number of change-points m is either known or unknown. In literature, different forms of penalty have been proposed for the penalty-based approach to estimate the number of change-points (e.g. Liu et al., 1997, Davis et al., 2016) and sometimes the location of the change-points as well (e.g. Chen et al., 2006). The penalty is often related to information criteria or some model complexity measures. The discussion on different forms of penalty and information criteria used for change-point selection is in Section 2.2.5.

The penalty-based change-point detection algorithms that aim to solve a global optimisation problem are slow with the time complexity $O(T^2)$. This makes them not feasible for long time series. Killick et al. (2012) propose Pruned Exact Linear Time (PELT) which uses optimum partition (Jackson et al., 2005) with a pruning step within the dynamic program. PELT has the time complexity $O(T)$ under certain conditions, but in the worst-case scenario the time complexity is $O(T^2)$.

Several greedy change-point detection algorithms are proposed to provide computationally efficient ways to detect change-points. One of the most well-established greedy change-point detection algorithms is binary segmentation. It first applies a single change-point detection method to the entire interval. If a change-point is detected, the interval is then split into two sub-intervals by the detected change-point. The same single change-point detection method is then performed on each sub-interval. The procedure is repeated until no more change-point can be found. For piecewise constant structure with noise, CUSUM-like test can be used as the single change-point detection method. When applying with a simple single change-point detection method, the time complexity of binary segmentation is $O(T\log(T))$.

While binary segmentation is computationally efficient, its solution is not necessarily the global minimum. Binary segmentation may fail if there are multiple change-points in the time series. Fryzlewicz et al. (2014) and Fryzlewicz (2020) propose the wild binary segmentation (WBS) for change-points in mean. In contrast to binary segmentation which applies a single change-point detection algorithm to the whole current interval, WBS draws multiple sub-intervals from the current interval and finds the change-point that maximises the CUSUM among all sub-intervals. Kovács et al. (2020) propose the Seeded Binary Segmentation which draws random segments like WBS (and NOT, which will be reviewed in the next paragraph), but it only focuses on shorter intervals to reduce computational time.

Narrowest-Over-Threshold (NOT, Baranowski et al., 2019) is a generalised change-point detection method as it allows different data structures:

- (S1) constant variance, piecewise-constant mean
- (S2) constant variance, continuous and piecewise-linear mean
- (S3) constant variance, piecewise-linear (not necessarily continuous) mean
- (S4) piecewise-constant variance, piecewise-constant mean

The Algorithm 1 from Baranowski et al. (2019) randomly draws multiple sub-intervals from the whole interval of data set. It then selects the point within the smallest sub-interval which the value of the corresponding contrast function exceeds a given threshold , i.e. select τ as a change-point such that:

$$\tau^* = \underset{\tau | b_m < \tau \leq e_m, m \in \mathscr{M}}{\arg\min} \{|e_m - b_m| : \max_{\tau} C(X_{b_m}, ..., Xe_m, \tau) > \zeta_T\}$$

where $\mathscr{M}$ is a set of randomly drawn intervals, with each interval $I_m = (b_m, e_m], m \in \mathscr{M}$. ζ_T is a predefined threshold and C is a contrast function.

The same procedure is then recursively applied to two sub-intervals divided by the newly detected change-point until no more change-points can be found. Similar to WBS, NOT uses the binary partitioning and random selection of the sub-intervals. Unlike WBS, it uses the contrast function derived from the generalised likelihood ratio (GLR) (instead of just the CUSUM in WBS) and it selects the smallest sub-interval that has a value exceed a given threshold (instead of selecting the sub-interval that has the highest value over the threshold in WBS), as the name "Narrowest-Over-Threshold" implies. This allows NOT to detect change-points in more general settings than the piecewise constant mean setting, which is the only setting considered by WBS.

The Algorithm 2 from Baranowski et al. (2019) generates a threshold-indexed solution path with sets of change-points selected along the thresholds. Such threshold-indexed solution path can be considered as a function mapping the thresholds to the sets of change-points $\zeta_T \to \mathscr{T}(\zeta_T)$. The value of the function only changes at some discrete points, i.e. there exists $0 = \zeta_T^0 < \zeta_T^1 < ... < \zeta_T^N$ such that $\mathscr{T}(\zeta_T^i) \neq \mathscr{T}(\zeta_T^{i+1})$ for all $i = 0, ..., N-1$ and $\mathscr{T}(\zeta_T) = \mathscr{T}(\zeta_T^i)$ for $\zeta_T \in [\zeta_T^i, \zeta_T^{i+1})$ with $\mathscr{T}(\zeta_T^N) = \emptyset$.

In order to select the final set of change-points, Baranowski et al. (2019) propose choosing the threshold on the solution path such that the strengthened Schwarz Information Criterion (sSIC, Liu et al., 1997 or LWZ as referred by Hall et al., 2013b and Bai and Perron, 2006) is minimised. sSIC is reviewed in Section 2.2.5. In their simulation setting, $\alpha = 1$ is used in sSIC, which is equivalent to using BIC. The authors justify the use of sSIC with the theoretical properties in terms of the estimation of the number and the location of change-points. Baranowski et al. (2019) also demonstrate the change-point detection performance in terms of the accuracy of the estimation of the number of change-points on simulated data.

2.2.5 Information criteria and entropy-based methods for model selection

Here we provide a literature review of the use of information criteria and entropy-based methods, in particular Bayesian information criterion (BIC) and its modified versions, in model selection and selecting the number (and the location) of the change-points on time series. Information criteria measure the model quality by considering both the model fit and the model complexity, with the model complexity is usually set as the number of (free) parameters. The model with the lowest information criterion value is preferred. The review of using BIC (and its modified versions) on selecting the number of change-points is largely followed from Hall et al. (2013b).

Bayesian information criterion (BIC, Schwarz et al., 1978) is defined as follows:

$$BIC = -2\log(L(\hat{\theta}; x_1, ..., x_T)) + p\log(T) \tag{2.13}$$

where L is the likelihood function and $\hat{\theta}$ are the parameter values estimated by the maximum likelihood estimation (MLE), p is the number of free parameters and T is the number of observations.

For i.i.d. normal noise, BIC can be simplified to:

$$\begin{aligned}
BIC &= -2\log(L(\hat{\theta};x_1,...,x_T)) + p\log(T) \\
&= -2\sum_{t=1}^{T}\log(\frac{1}{\sqrt{2\pi\widehat{\sigma^2}}}\exp[-\frac{(x_t-\hat{\mu})^2}{2\widehat{\sigma^2}}] + p\log(T) \\
&= T\log(2\pi) + T\log(\widehat{\sigma^2}) + \frac{\sum_{t=1}^{T}(x_t-\hat{\mu})^2}{\widehat{\sigma^2}} + p\log(T) \\
&= T\log(2\pi) + T\log(\widehat{\sigma^2}) + \frac{T\widehat{\sigma^2}}{\widehat{\sigma^2}} + p\log(T) \\
&= T\log(2\pi) + T\log(\widehat{\sigma^2}) + T + p\log(T) \\
&= T\log(\widehat{\sigma^2}) + p\log(T) + C
\end{aligned}$$

where $\widehat{\sigma^2} = \frac{\sum_{t=1}^{T}(x_t-\hat{\mu})^2}{T}$ is the MLE estimated variance and C is some additive constant that does not depends on the estimated parameters or the number of parameters.

In the statistics literature, BIC is commonly used in time series analysis as a criterion for selecting from models with a different number of parameters within the same model class. For example, Pötscher (1990) use BIC to select the order of the ARMA model, and Beran et al. (1998) use BIC to select the order for both short and long memory AR model.

While information criteria are widely used in selecting models for time series, interestingly the discussion is mainly on selecting models from the same model class in the statistics literature. For distinguishing the long memory from the short memory change-point models, which is the main focus of this chapter, there are more discussions in economics on empirical data. For example, Song and Shin (2015) use BIC to select a model for short or long memory time series with or without change-points for realised volatility on the exchange rate. Fukuda (2009) use BIC to choose between the no change-point i.i.d normal / student T model and the one change-point i.i.d normal / student T model for stock returns. They treat the unknown location of the change-point as another parameter, which is similar to Yao (1988). They also provide simulation results which show that BIC performs well in distinguishing the no change-point model and the one change-point (change in variance) model with normal noise in their settings. Granger and Hyung (2004) use BIC to show that the break model and the $I(d)$ model virtually have the same explanatory power on the S&P 500 return data. For selecting models from different classes in some more general settings to information criteria, Lloyd et al. (2014) use BIC to evaluate different models (models with change-points, addition structures, with trend, etc.) in the Automatic Bayesian Covariance Discovery (ABCD) system.

For change-point detection, using BIC (and other information criteria) to select among different change-point models is not as simple as treating the locations of change-points as another parameters. This is because the positions of change-points are discrete variables, but BIC assumes the parameter space is continuous, and hence the positions of change-points are not regular parameters. In literature, several versions of BIC are proposed to extend the use of BIC to change-point models, and we review them in details below.

BIC from Yao (1988) for change-point process

While the location of change-points is not a regular parameter, Yao (1988) establish the consistency of BIC for the estimation of the number of breaks, when treating the location of change-points the same way as other parameters, and the only parameter of interest is the mean of an i.i.d Gaussian process. The BIC is then:

$$BIC = T\log(\widehat{\sigma^2}) + (p+m)\log(T) \tag{2.14}$$

where m is the number of change-points, p is the number of other unknown variables, $\widehat{\sigma^2} = \frac{RSS}{T}$ with RSS is the residual sum of squares. For the piecewise constant mean with common variance, $p = m+2$ is set by Yao (1988) as there are $m+1$ mean and 1 variance to estimate.

Bai et al. (2000) consider using information criteria for the estimation of the number of breaks in a vector autoregression model (VAR). Structural changes are in both the variance-covariance matrices and the regression coefficients. They show that BIC is consistent in selecting the right number of change-points.

The consistency results on BIC do not necessarily imply good empirical performance. Bai and Perron (2006) consider multiple linear regression with breaks. They show that when there is serial correlation in residuals, the number of change-points estimated by BIC is higher than the actual number of change-points.

Strengthened Schwarz information criterion (sSIC, Liu et al., 1997)

Liu et al. (1997) consider segmented multivariate regression models setting and propose a modified version of BIC. Here we refer the criterion as "strengthened Schwarz information criterion" (sSIC) following Fryzlewicz et al. (2014) and Baranowski et al. (2019). The same information criterion is referred as LWZ by Hall et al. (2013b) and Bai and Perron (2006) by taking the first letter from the surname of the authors (Liu, Wu and Zidek). Following Fryzlewicz et al. (2014) and Baranowski et al. (2019), sSIC can be defined as follows:

$$sSIC = -2\sum_{j=1}^{m+1}\log(L(X_{\hat{\tau}_{j-1}+1},\dots,X_{\hat{\tau}_j};\hat{\Theta}_j)) + cp'\log^{\alpha}(T) \tag{2.15}$$

where m is the number of change-points, T is the total number of observations, L is the likelihood function, τ_j is the position of the j^{th} change-point, $\hat{\Theta}_j$ are the estimated parameters, c is some positive constant, $\alpha \geq 1$ is some constant and p' is the total number of estimated parameters, including the location of change-points and other free parameters. When $\alpha = 1$, and $c = 1$, the sSIC is the same as BIC. In Fryzlewicz et al. (2014) where only change in mean is considered, the sSIC can be simplified as:

$$sSIC = T\log(\widehat{\sigma^2}) + (p+m)\log^{\alpha}(T) \tag{2.16}$$

with $\widehat{\sigma^2} = RSS/T$, RSS is the residual sum of squares and m is the number of change-points and $p = m+2$ is the total number of other unknown parameters (mean and variance).

In the original paper from Liu et al. (1997), the criterion is defined as follows:

$$LWZ = T\log(\frac{RSS}{T-p'}) + p'c_0\log^{2+\delta_0}(T) \tag{2.17}$$

where T is the total number of observations, RSS is the residual sum of squares, c_0 and δ_0 are some positive constant and p' is the total number of estimated parameters, including the location of change-points and other free parameters. Note that both the penalty term and the term for the negative log-likelihood in Equation (2.17) are different from the one in Equation (2.14). The term for the negative log-likelihood in Equation (2.17) is different from the one in Equation (2.16) as well, but we still refer both as sSIC given that they both have power on the penalty term.

Liu et al. (1997) argue that a penalty severer than the one used in BIC from Yao (1988) is needed for the correct specification of a non-Gaussian regression model with structural breaks. Liu et al. (1997) show that for sSIC, the estimate of the number of change-points is weakly consistent for errors with any distribution with zero mean and a moment generating function.

Modified information criterion

Chen et al. (2006) introduce the modified information criterion (MIC) which is similar to BIC but its value depends on the position of the change-points as well. MIC is based on the observation that when there is only one change-point, and if the change-point is at the beginning or the end of the time series, one of the two sets of the parameters becomes completely redundant. The MIC with one change-point is defined as follow:

$$MIC(k) = -2[\sum_{i=1}^{\tau}\log L(X_i;\hat{\theta}_1) + \sum_{i=\tau+1}^{T}\log L(X_i;\hat{\theta}_2)] + 2p\log(T) + (\frac{2\tau}{T}-1)^2\log(T)$$

where τ is the position of the change-point, p is the number of parameters in each section, L is the likelihood function and $\hat{\theta}_1$ and $\hat{\theta}_2$ are the estimated parameters. If the change-point is closer to the boundary, the MIC is higher (assuming everything else is the same).

For multiple change-points, Pan and Chen (2006) propose the following:

$$MIC = -2\sum_{j=1}^{m+1}\sum_{i=\tau_{j-1}+1}^{\tau_j} \log L(X_i;\hat{\theta}_j) + p(m+1)\log(T) + C\sum_{j=1}^{m+1}(\frac{\tau_j-\tau_{j-1}}{T} - \frac{1}{m+1})^2 \log(T)$$

where τ_j is the position of the j^{th} change-point with $\tau_0 = 0$ and $\tau_{m+1} = T$, p is the number of parameters in each section and m is the number of change-points. If the change-points are close to each others or close to the boundary, the MIC is higher (assuming everything else is the same). Pan and Chen (2006) show that given the right number of change-points, MIC estimators for the location of the change-points attain the best rate.

Number of change-points has higher weight than other parameters

Ninomiya (2005) suggest the penalty for Akaike Information criterion (AIC, Akaike, 1974) on the number of change-points should be three times the penalty of other parameters on independent Gaussian (or weakly dependent stationary sequences) with change in mean and variance. AIC selects the fitted model minimising the Kullback–Leibler divergence from the true model, or equivalently maximising the expected log-likelihood of $E_X[\log(f(X|\hat{\theta}_Y)]$, where X and Y are from the same distribution but they are independent. However, if we use $\log(f(X|\hat{\theta}_X))$ to estimate $E_X[\log(f(X|\hat{\theta}_Y)]$, we will overestimate the value. Ninomiya (2005) show that for the change-point models, the bias-corrected maximum log-likelihood depends on the expected value of the maximum of a random walk with negative drift. By approximating it with a Brownian motion, Ninomiya (2005) evaluate the bias as $3m+p$ where m is the number of change-points and p is the number of other parameters.

Kurozumi and Tuvaandorj (2011) suggest the penalty for BIC on the number of change-points should be two times the penalty of other parameters for linear multivariate regressions.

For simple referencing we call their proposed BIC as “BIC2”. The regressors can be serially correlated or the lagged dependent variables, and the structural changes in the variance matrices are allowed. Their result follows from the assumption that the prior distribution probability of model $M(m, p, T)$ is $\alpha_{m,T}/T^m$ where m is the number of change-points, T is the length of the time series and $\alpha_{m,T}$ is bounded. This choice of the prior distribution is motivated by some examples like considering the location of each the change-point is independently uniformly distributed. Kurozumi and Tuvaandorj (2011) argue that the model selected by BIC2 is optimal in the sense that it maximises the posterior probability. As the BIC2 proposed by Kurozumi and Tuvaandorj (2011) has a higher penalty on the number of change-points than the BIC from Yao (1988), it tends to choose less number of change-points than the BIC from Yao (1988). Kurozumi and Tuvaandorj (2011) show that under their simulation settings, the BIC from Yao (1988) and BIC2 perform relatively well when the number of change-points $m \leq 1$.

Hall et al. (2013a) and Hall et al. (2013b) extends the result from Ninomiya (2005) to the regression models, and show that the number of change-points in BIC should have three times the penalty of other parameters. We call their proposed BIC as “BIC3”. Hall et al. (2013a) show that under linear regression model the asymptotic expected value of the difference in residual sum of squared (RSS) based on the estimated change-points and RSS based on the true change-points with estimated parameters is $-3m\sigma^2$. On the other hand, the asymptotic expected value of the difference in RSS based on the estimated parameters and the RSS based on the true parameters given the true change-points is $-p(m+1)\sigma^2$, where p is the number of parameters in each segment. This shows that the number of change-points should have three times the penalty of other parameters. They assume the breaks are ‘shrinking’, in the sense that the size of the breaks is assumed to converge to zero when the number of observations increases. Hall et al. (2013b) show that BIC3 provides good empirical performance, even in the presence of serial correlation. For the settings without the serial correlation in the noise, however, the BIC from Yao (1988) performs better.

Minimum description length (MDL)

Minimum description length (MDL) is an entropy-based method which selects the model that can describe the data in the most parsimonious way. MDL, roughly speaking, can be thought as the minimum number of digits in a binary string needed to code the data without any loss of the data. A model with smaller MDL is likely to be simpler and thus preferred. Unlike BIC which may not be directly applicable to some models (e.g. change-point models), MDL can be used on different types of models.

The description length is usually separated into two parts. The first part is the length of the code used to describe the model. The second part is the expected code length to describe the residuals given the model is specified, which is equal to the negative log-likelihood of the residuals. The length of the code used to describe the model can be used as a measure of the model complexity. Davis et al. (2016) show that the location and the number of change-points selected by MDL is consistent when each segment between two consecutive change-points is modelled by a pre-specified family of parametric stationary time series.

Example 2.1. *Assume we have a short memory change-point model with one change-point and each part is ARMA(1,1) (i.e. $\mathscr{M}_1$) and assume that the unknown parameters are μ, θ and ϕ in each segment and the location of the change-point. The description length of the model (i.e. not include the description length of the residuals) following from Davis et al. (2008) and Hansen and Yu (2001) is:*

$$\log(T) + \frac{3\log(\tau)}{2} + \frac{3\log(T-\tau)}{2} \tag{2.18}$$

where $\log(T)$ *is the code length to describe the change-point,* $\frac{3\log(\tau)}{2}$ *is the code length to describe the* 3 *free parameters of the first segment* (θ_1, ϕ_1, μ_1)*, with each parameter has* *the code length of $\frac{\log(\tau_1)}{2}$, assuming the accuracy of parameter estimation is in the order of $O(\sqrt{n})$, $\frac{3\log(T-\tau)}{2}$ is the code length to describe the 3 free parameters of the second segment $(\theta_2, \phi_2, \mu_2)$.*

Similar to MIC, MDL considers the location of the change-points in Example 2.1. Different from MIC, the "penalty" of MDL is smaller when the change-point is near to the boundary, assuming everything else is the same. This makes MDL favours change-point closer to the boundary whereas MIC favours change-point that is in the middle and not close to the boundary.

2.2.6 Time series model identification

In this chapter we consider the situation where we only have one realisation of a time series and the time series is either from a short memory change-point with a single change-point or a long memory process. Our objective is to find the correct model specification for the given time series, i.e. to be able to tell if the time series is a short memory change-point or a long memory process.

2.2.6.1 Hypothesis testing

Berkes et al. (2006) propose a test procedure based on the CUSUM statistics to distinguish time series with change-points in mean from time series with long-range dependence. The null hypothesis is that the time series is weakly dependent with one change-point in mean, and the alternative hypothesis is the long memory model. Baek and Pipiras (2012) note that the test suggested by Berkes et al. (2006) has little power, and they suggest another test based on the local Whittle estimation. Baek et al. (2014) suggest another way to distinguish multiple change-points in mean with the long memory. For a sequence of change-points detected, the long memory parameter d is estimated with the changes in mean removed. The process stops when the hypothesis of the short memory (i.e. $d = 0$) cannot be rejected following the test from Baek and Pipiras (2012). Baek et al. (2014) call this the “LW stopping rule”. The numerical experience from Baek et al. (2014) suggests that if the observed time series is from a long memory process, then the number of change-points estimated by the LW stopping rule is much larger than another stopping rule (e.g. the CUSUM-based test from Berkes et al., 2006). Baek et al. (2014) argue that this is because the power of the test from Baek and Pipiras (2012) is much higher than the test from Berkes et al. (2006). Therefore, Baek et al. (2014) suggest that if the number of change-points estimated by the two procedures is similar, then there is evidence that the time series is from a short memory change-point process. If the number is very different, then there is evidence that the time series is from a long memory process. The authors propose a test for the short memory change-point and the long memory based on a test on whether the two procedures estimate a different number of change-points.

Yau and Davis (2012) propose a likelihood ratio (LR) test to distinguish the long memory in the form of ARFIMA and the short memory change-point time series with each segment an ARMA. They set the short memory change-point model as the null hypothesis and argue that short memory change-point model should be set as null because it is easier to explain a time series with a short memory change-point model than a long memory with fractional integration. The LR statistic is defined as the normalised log-ratio of the Whittle Likelihood between the short memory change-point model and the long memory model, which is asymptotically normally distributed under the null.

2.2.6.2 Classification

Classification is a problem of finding the true category of new observations. While here the literature review of time series classification is put under the time series model identification section, classification techniques cannot be directly applied to time series for model identification. This is because for model identification, only one time series is observed and there is no training data for us to train the classifiers. After reviewing the time series classifiers, we review works on simulating data for classification, which may extend the use of time series classifiers to model identification.

Classifiers

In this thesis, we focus on model identification on time series that are from the short memory change-point model or the long memory model. Our literature review therefore also focuses on some related time series classification problems, for example stationary vs non-stationary time series. Before we review the classifiers that are specialised in these areas, we will briefly review some more general time series algorithms.

In literature, there are many algorithms proposed for time series classification. For time series data like records of motion, records from sensors, ECG, etc, nearest neighbour with Dynamic Time Warping (NN-DTW), shapelet (Ye and Keogh, 2009) with decision trees are some commonly used time series classification algorithms. Bagnall et al. (2016) provide a review of different time series classification algorithms.

While most attention of using deep neural networks in classification has drawn to image classification in recent years, deep neural networks can also be used for time series classification. For example, Wang et al. (2017) propose using multilayer perceptron, fully convolutional networks (FCN) and residual nets (ResNet, He et al., 2016) for time series classification. Serrà et al. (2018) suggest using Encoder, which is a deep neural network like FCN, but the Global Average Pooling (GAP) layer is replaced by an attention layer. Wang et al. (2017) and Fawaz et al. (2019) show that some deep neural networks like ResNet perform well and significantly outperform the NN-DTW when classifying time series from the UCR Time Series Classification Archive (Dau et al., 2018). This is the case even when the number of samples is small.

For classifying (or clustering) stationary and non-stationary time series, Caiado et al. (2006) propose the use of log-normalised periodogram with Euclidean distance and distance based on Kullback–Leibler distance to measure the similarity between time series. They provide an empirical study on the performance of classification based on clustering algorithms with their proposed distance metrics and other distance metrics like Euclidean distance on the original data, Piccolo's distance (Euclidean distance on the estimated coefficients of $AR(\infty)$) and Euclidean distance on ACF and PACF coefficients. Their results show that the methods based on periodogram and ACF provide the best classification results in their simulation settings.

Huang et al. (2004) propose using SLEX (smooth localised complex exponential) for classifying non-stationary time series. Their method selects the best Fourier-type dyadic basis and classifies the new observations based on a discriminant criterion related to the Kullback-Leibler distance of the chosen basis between the SLEX spectra of different groups.

Fryzlewicz and Ombao (2009) propose a classification procedure to classify non-stationary time series using the empirical evolutionary wavelet spectra (EWS) from the locally stationary wavelet (LSW) model. Wavelet is localised in both in time and in frequency, and LSW provides a frequency time-scale decomposition to identify the local time-scale features. Their method assigns the new observation to the group which has the shortest squared quadratic distance between the EWS of the group and the EWS of the new observation. In their simulation they consider piecewise AR and AR with time-varying parameters.

Krzemieniewska et al. (2014) propose a classification method based on Fryzlewicz and Ombao (2009). They observe that wavelet spectrum from some groups may be more variable than the others and their method accounts for the difference in variability by using a variance-corrected squared distance on EWS.

Residual net (ResNet, He et al., 2016) on time series classification

The literature Fawaz et al. (2019) and Wang et al. (2017) on deep learning on time series reviewed above shows that the ResNet provides very good classification performance. Therefore we focus our discussion about deep learning on ResNet and below we provide a literature review of ResNet.

Researchers have been increasing the depth (or the number of layers) of the neural networks (NNs) to improve the image recognition performance. For example, GoogLeNet (Szegedy et al., 2015) has 22 layers comparing to some previous NNs like VGG (Simonyan and Zisserman, 2014, 19 layers) and AlexNet (Krizhevsky et al., 2012, 8 layers). He et al. (2016) explore the relation between the increase in the number of layers and the classification performance. They observe that deeper NNs do not necessarily mean better performance. In their setting, an NN with 56 layers performs worse than an NN with 20 layers. It may seem to be an example of overfitting, as the NN with more layers contains more parameters. Overfitting happens when the training error is decreasing but the test error is increasing with the increase in the number of parameters. However, He et al. (2016) observe that the training error is also higher for the NN with 56 layers than the one with 20 layers. He et al. (2016) argue that deeper NNs are more difficult to train and optimise, so adding more layers to NN can worsen its performance in reality. Instead of adding more layers naively and hoping that they can fit a desired underlying mapping, He et al. (2016) propose adding some shortcuts between some consecutive convolutional layers so that the layers explicitly fit a residual mapping. Such structure makes the NNs easier to be fitted and the ResNet in He et al. (2016) provide good classification performance with 152 layers.

In Fawaz et al. (2019), the ResNet they use has three residual blocks. It has 11 layers in total with the first 9 layers are convolutional (which every three layers are in one residual block) and then a Global Average Pooling (GAP) layer that averages the time series across the time dimension and a final softmax classifier. While the ResNet considered by Fawaz et al. (2019) is much shallower than the one from He et al. (2016), it is the deepest NN considered by Fawaz et al. (2019).

Simulation-driven machine learning: ARMA

In literature, there are some works about using the simulation-driven approach to find the order of the ARMA models. Here I highlight those that are using deep learning methods for fitting. Tang and Röllin (2018) use ResNet to find the order of ARMA time-series with simulated data. The coefficients of ARMA ϕ and θ are generated using the algorithm from Beadle and Djuric (1997), for which it allows the coefficients of the stationary and invertible ARMA time series to be generated uniformly. “Training” data is then simulated from the generated coefficients, and used to train the ResNet without pre-processing. The authors observe that their method outperforms information criteria methods when identifying individual AR and MA orders, but BIC has a better performance in selecting both orders correct simultaneously. On the other hand, Moon et al. (2021) uses 2 dimensional ResNet and Inception for the fitting part. From their paper, it is not very clear how the 362,160 coefficients for the ARMA “training” data are selected. Different from Tang and Röllin (2018), their method requires pre-processing on the “training” time series. The processed “training” data is then used as 2 dimensional input to train the deep learning methods. Their results show that even if the sets of coefficients used in training and testing are different, the deep learning methods with the pre-processed data still outperforms AIC and BIC under their settings. Interestingly they show that in their simulation settings, deep learning methods using the original time series (i.e. without pre-processing) often perform worse than the information criteria methods, when the sets of coefficients used in training and testing are different.

While we can simulate the ARFIMA with a similar approach as Tang and Röllin (2018), it is not clear how we can simulate the size of the jumps without looking at the data. Therefore, we will not consider this approach in this thesis.

Simulation-driven machine learning: others

In literature, simulated data are also used to train classifiers or algorithms for fault detection.

Sobie et al. (2018) use the simulated training data to train classifiers for bearing fault classification in rotating machines. They generate training data from high-resolution simulations of roller bearing dynamics. Classifiers like convolutional neural network (CNN), support vector machine (SVM), k-nearest neighbours (KNN) are then trained using the simulated training data. The performance of the classifiers trained using the simulated data ("simulation-driven classifiers") is compared with the same classifiers trained using the experimental data produced in a laboratory setting ("data-driven classifiers"). The results from Sobie et al. (2018) show that, the simulation-driven classifiers outperform the corresponding data-driven classifiers, even when the size of training data is the same. Sobie et al. (2018) argue that the variation in the data makes the difference - the experimental data produced from a laboratory setting is often limited to specific circumstances and cannot capture the variation seen in an industrial setting. Elkordy et al. (1993) use NNs to identifying changes in the vibrational signatures of a structure. As getting the sample cases through a physical model is time consuming and may not even be feasible, Elkordy et al. (1993) use training data generated from a mathematical model to train the NNs. Elkordy et al. (1993) argue that NNs are able to classify correctly even when they are trained with partially inaccurate sample cases.

Murphey et al. (2006) use simulated training data for multi-class fault detection in an electric drive system. Their simulation model is developed based on the theoretical foundations of electric drives. Simulated data is used to train the fault diagnostic neural network (FDNN) and the performance of the FDNN is evaluated. If the performance of the trained FDNN is not satisfying, the parameter space for the simulation model is updated in order to generate more representative data. This simulation-training-evaluation process repeats until the performance of FDNN is satisfying. Murphey et al. (2006) show that their proposed approach is effective in detecting multiple classes of faults in an electric drive inverter.

For more simulation-driven machine learning for fault detection in literature, Gecgel et al. (2018) propose a simulation-driven machine learning framework for gear fault multi-class classification with the training data simulated from some models like the 6-degrees-of-freedom model, and they evaluate the performance of the approach using different classifiers like decision tree, KNN, SVM, etc. Chen and Randall (2016) use NN to classify the big-end bearing knock fault levels with the training data simulated from a model. Er-raoudi et al. (2016) proposed a classification method based on NNs, discrete wavelet transform and principal component analysis, with training data from the 6-degrees-of-freedom model are used to train the classifier.

For the model identification problem, we only have one observed time series and we do not know which model the time series belongs to. Neither a simulation model based on some previous works or some theoretical foundations nor data to evaluate the quality of the simulated data is available. The simulation model, therefore, has to based on the one observed time series, as it is done in Norwood and Killick (2018).

Data augmentation

When fitting a complex classifier with a relatively small amount of training data, the trained classifier is often over-fitted and does not perform well on the unseen test data (Perez and Wang, 2017). Data augmentation is a method to increase the number of training data by using the information only from the available training data. For image classification, there are many different ways to do the data augmentation. The "classical" data augmentation strategies include affine transformation like cropping, scaling, rotation, translating and reflecting the images, changing the colour of the images (e.g. turn coloured images to grey-scale), and performing shape deformations (elastic and shearing deformation) on the images.

Generative Adversarial Nets (GANs, Mirza and Osindero, 2014) generate new data from the given training data. GANs consist of two NNs, which one NN (generator) generates "fake data" (i.e. the new simulated data) with the same statistics as the training set in order to fool the other net (discriminator) which is trained to be able to distinguish the "fake data" from the "true data" (i.e. the actual training data). The generated data from GANs can be used to improve the performance of classifiers. For example, Frid-Adar et al. (2018) propose using both classical data augmentation and GAN to enlarge the training data set. Their results show that data augmentation improves the performance of the classifier on medical images.

Data augmentation requires a relatively large amount of data to train the method to generate the simulated data (e.g. GANs), or at least some training data with labels on (e.g. affine transformation on images like cropping, scaling, rotation, translating and reflecting to create more training samples with the same labels). These methods cannot directly apply to the model identification problem.

Estimation-simulation-classification (ESC) approach

Norwood and Killick (2018) attempt to identify the most appropriate model for a time series that is either from a short memory change-point or long memory model through classification. They argue their proposed method is not intended to propose a final model for the observed time series, but instead to provide some additional information like confirming the models proposed from other works.

Norwood and Killick (2018) argue that in contrary to the hypothesis testing approach, in some situations there is no clear null model when distinguishing the long memory and short memory change-point process, and it is not easy to set the short memory change-point model as the null. Norwood and Killick (2018) instead attempt to classify if a time series is from a long memory and short memory change-point model using a classifier via simulation. They define the change-point model ($\mathscr{M}_1$) as the same as Equation (2.7), with $p, q \leq 1$.

For the stationary long memory model ($\mathscr{M}_2$), they assume it is in the form of autoregressive fractional integrated moving average (ARFIMA):

$$X_t \sim \mu + ARFIMA(\phi, d, \theta), \text{ if } t = 1, 2, ..., T \tag{2.19}$$

with $0 < d < 0.5$ and $\sum|\phi| < 1$, with $p, q \leq 1$.

They observe that although the long memory and short memory change-point models can behave quite similarly (e.g. within their spectrum), the time series from the two models look different in terms of the time-varying wavelet spectrum. In order to use the wavelet spectrum to distinguish the short memory change-point model from the long memory model similar to Fryzlewicz and Ombao (2009) and Krzemieniewska et al. (2014), they build a classifier with simulated data. Their proposed procedure is summarised in Algorithm 2.1. For simple referencing, we call the method from Norwood and Killick (2018) "WCA", which is abbreviated from "Wavelet Classifier Algorithm".

The distance metric used in Step (6) is a variance-correlated squared distance proposed by Krzemieniewska et al. (2014). Norwood and Killick (2018) illustrate the performance of their proposed method via simulation and compare with the hypothesis testing procedure proposed by Yau and Davis (2012).

2.2.7 Approximate model

The main idea of approximation from Davies (2014) and the related works from the same author is that there is no so-called "true" model for the observed data we have at hand. Instead the best we can do is to have some "approximate" models, which approximate the observed data "well enough". If we need to select one model from the approximate models, we should choose the one that is the "simplest" so that the result is more regularised and stable. In this chapter we assume there is a true model, and therefore we have a different setting from Davies (2014). Nonetheless, the idea of the approximation and having "typical data" "look like" the observed data are related to what we aim to achieve in this chapter - to generate some "training data" that are "similar" to the observed data to fit a classifier. Therefore, we review the works here.

Algorithm 2.1 Wavelet Classifier Algorithm from Norwood and Killick (2018)

```
Input: Observed time series $\{x_1, ..., x_T\}$, model specifications $\mathscr{M}_1$ and $\mathscr{M}_2$, number of training data to simulate $M$.
Output: predicted class for the observed time series $\{x_1, ..., x_T\}$.
1: Get the best fitted model for $\{x_1, ..., x_T\}$ from $\mathscr{M}_1$.
2: Get the best fitted model from for $\{x_1, ..., x_T\}$ from $\mathscr{M}_2$.
3: Get $M$ “training data” by simulating data from the fitted models from Step (1) and (2).
4: Get the evolutionary wavelet spectra for both the training and the observed data. Denote the wavelet spectrum of the observed data as
       $$S^o = \{S^o_k\}_{k=1,2,...,T*J}$$
   and the spectra for each simulated time series from each model $g = \{\mathscr{M}_1, \mathscr{M}_2\}$ as
       $$S^g_{:,m} = \{S^g_{k,m}\}_{k=1,2,...,T*J}$$
   where $k$ indicates the position and the scale of corresponding wavelet spectrum.
5: For a given k, calculate the average Evolutionary Wavelet Spectra for each group:
       $$\bar{S}^g_k = \frac{1}{M}\sum_{m=1}^{M}\{S^g_{k,m}\}$$
6: Calculate the distance of the observed data from each group of the simulated training data. The distance metric is defined as follow:
       $$D^g = \frac{M}{M+1}\sum_{k=1}^{T*J}\frac{(S^o_k - \bar{S}^g_k)^2}{\sum_{m=1}^{M}(S^g_{k,m} - \bar{S}^g_k)^2}$$
7: The observed time series is classified as model group $g$ with
       $$g = \underset{g\in\{\mathscr{M}_1,\mathscr{M}_2\}}{\arg\min}\ D^g$$
8: return g
```

Approximating a data set by a model

Given a data set x_n and a probability model P, a model is said to be an "adequate approximation" for the data if the "typical data" sets generated under P "look like" the observed data x_n. Two data "look like" if they share some specific properties. "Typical data" corresponds to a large proportion of data $X_n(P)$ generated under the model P. When the real data set exhibits the specific properties shared by a large proportion of data generated under the model, the model is said to be an adequate approximation of the real data.

A formal definition of approximation is as follow. Given the data x_n and the model P, the property or properties of interest are identified with a subset $E_n(\alpha,P) \subset \mathbb{R}^n$ such that

$$P(X_n(P) \in E_n(\alpha,P)) \geq \alpha$$

Note the model P here is not assumed to be true. Therefore, the model P does not determine $E_n(\alpha,P)$ by, say, optimisation. Instead, researchers can choose the set $E_n(\alpha,P)$ based on the problem at hand.

Approximation regions

Approximation region can be interpreted as a set of models that behave similar to the observed data. More formally, given α and region $E_n(\alpha,P)$ for each $P \in \mathscr{P}$, the approximation region $\mathscr{A}(x_n,\alpha,\mathscr{P})$ is defined as:

$$\mathscr{A}(x_n,\alpha,P) = \{P : P \in \mathscr{P}, x_n \in E_n(\alpha,P)\}$$

If $\mathscr{P}$ is a parametric family of models,

$$\mathscr{P} = \mathscr{P}_\theta = \{P_\theta : \theta \in \Theta\}$$

then

$$\mathscr{A}(x_n, \alpha, \Theta) = \{\theta : \theta \in \Theta, x_n \in E_n(\alpha, P_\theta)\}$$

is an honest α-approximation region for the values of the parameter θ.

Regularisation and optimality

The approximation regions only return a set of models. If researchers want to get one single model to approximate the observed data, regularisation should be used to select a simpler model according to Davies (2014). Davies (2014) argue the form of regularisation depends on the problem at hand, and the regularisation needs not to be done with respect to the model.

2.3 Studying the method proposed by Norwood and Killick (2018) and our approach

In this section we study further the method proposed by Norwood and Killick (2018), which is referred as "WCA" (abbreviated from Wavelet Classification Algorithm) in this thesis, after the literature review in Section 2.2.6.2. WCA uses the ESC ("estimation-simulation-classification") approach, which uses training data simulated from an estimated model to train a classifier. By studying the simulation results of WCA from Norwood and Killick (2018) and understanding the behaviour of WCA, we want to find out if the ESC approach has the potential to provide good classification performance. Throughout the chapter, we follow the settings from Norwood and Killick (2018), which is to distinguish the long memory model in terms of $ARFIMA(p,0,q)$ $\mathscr{M}_1$ with $p,q \leq 1$ in Equation (2.19) and the short memory change-point model $\mathscr{M}_2$ with one change-point and ARMA in each segment in Equation

(2.7). For simple referencing, we restate the models here. The $\mathscr{M}_1$ model is:

$$X_t \sim \begin{cases} \mu_1 + ARMA(\phi_1, \theta_1), \text{ if } t = 1, 2, ..., \tau \\ \mu_2 + ARMA(\phi_2, \theta_2), \text{ if } t = \tau + 1, \tau + 2, ..., T \end{cases}$$

where ARMA is an autoregressive moving average model with ϕ is an autoregressive (AR) parameter, θ is a moving average (MA) parameter, τ is the location of the change-point and T is the number of observations. Some parameters in the first segments $(\mu_1, \phi_1, \theta_1)$ must be different from the parameters in the second segments $(\mu_2, \phi_2, \theta_2)$. The ARMA in each segment is stationary.

The $\mathscr{M}_2$ model is:

$$X_t \sim \mu + ARFIMA(\phi, d, \theta), \text{ if } t = 1, 2, ..., T$$

with $0 < d < 0.5$ and $\sum |\phi| < 1$, with $p, q \leq 1$.

2.3.1 Simulation results from Norwood and Killick (2018)

The simulation results from Norwood and Killick (2018) show that WCA selects the right model specification when the observed time series is from the short memory change-point model $\mathscr{M}_1$ with 100% accuracy. WCA, however, is not able to recommend the right model specification when the observed data is from the long memory model $\mathscr{M}_2$. For example in the long memory simulation settings 19-22 from Norwood and Killick (2018) (see Table 2.1 for the simulation settings), the classification rate reported in their paper is $0.33 - 0.44$ when number of observations is $T = 512$, which is worse than a random guess. They use the simulation results from Yau and Davis (2012), which is a hypothesis testing, as a comparison.

Norwood and Killick (2018) interpret the poor results under the long memory settings as "the variation within the wavelet spectrum of long memory series that could be interpreted as different levels and hence a change-point model would be more appropriate". Their comment seems to attribute the poor performance of WCA to the wavelet spectrum, which is the classification part of the ESC approach. We, however, suspect that the poor results are also attributed to the "estimation" and the "simulation" parts of the ESC approach. In particular, we suspect the difference in model complexity in $\mathscr{M}_1$ and $\mathscr{M}_2$ is ignored in the estimation and the simulation part of WCA from Norwood and Killick (2018), so that the simulated data from a more complex model (which is $\mathscr{M}_1$ in Norwood and Killick, 2018) are more similar to the observed data than desired.

2.3.2 Simulation, similarity and model complexity

There is a clear problem with fitting a classifier with simulated pseudo training data if the model complexity is not handled properly. Assume model 1 is relatively simple but model 2 is extremely flexible - flexible to the extent that the fitted model according to model 2 is just a perfect replica of the original data with no error. All the "simulated" pseudo training data from model 2 will just look exactly like the observed time series we need to classify. In this case, even if model 1 is the true model, the classifier is very likely to recommend model 2 as the true time series looks exactly the same as the "training" data from model 2.

In a less extreme setting, the fitted model from a false model specification is likely to have a better fit to the actual data than the fitted model from the true model specification in terms of MSE, if the false model specification is more flexible. The pseudo training data from a false but more flexible model specification then can be more similar to the actual data than the pseudo training data from the true but less flexible model specification in a similar measure. This causes the ESC procedure bias towards models that are more "flexible", as the pseudo training data generated from the more flexible model fitted using the observed data is likely to be more similar to the time series at hand. For the simulation-driven machine learning methods or the data augmentation we reviewed in Section 2.2.6.2, the model complexity is not an issue because either the model used to simulate the training data is given (i.e. not estimated from the model), or there are a relatively large amount of the data to train the method to learn how to simulate good data.

A topic related to how model complexity affects the ESC approach is how model complexity affects goodness of fit and model identification. The importance of taking into account the model complexity in model selection has been highlighted in literature. Many different methods like information criteria and entropy-based approaches take into account the model complexity for model selection, with the methods related to the selection of the location and the number of change-points are reviewed in Section 2.2.5. For the information criteria, the model complexity is often measured in terms of the number of parameters.

The importance of model complexity in model selection is also emphasised in different subjects like psychology (Myung, 2000), computer modelling (Brooks and Tobias 1996), ecology (Warren and Seifert, 2011), and many more. Myung (2000), for example, argue that the goodness of fit is necessary but not sufficient for model selection, and the model complexity must also be taken into consideration. Their illustrative example shows that more complex models can provide a better fit than the true model, as the more complex models have the flexibility to capture the random noise. In their simulation, they show that the using methods that take complexity into consideration (e.g. BIC) can select the right model with high probability.

2.3.3 Model complexity in Norwood and Killick (2018)

In the estimation part of Norwood and Killick (2018), the number of free parameters in the short memory change-point model $\mathscr{M}_1$ and the long memory model $\mathscr{M}_2$ considered by WCA is different: The short memory change-point model can have at most 7 parameters (1 change-point and 3 parameters for each ARMA segment with $p,q = 1$) and the long term model can have at most 4 parameters (mean, AR parameter ϕ, MA parameter θ and the fractional difference parameter d). While the number of parameters does not necessarily fully account for the model complexity, and the actual fitted short memory change-point model does not necessarily have more parameters than the fitted long memory model, the difference in the maximum number of parameters in the two models makes us suspect that the actual complexity of the short memory change-point model is likely to be greater than the complexity of the long memory model.

Moreover, the difference in the number of parameters does not seem to be directly taken into account by Norwood and Killick (2018). As the model complexity is not taken into account, the classifier trained using the simulated data should partial towards the more flexible model, and in their case it is the short memory change-point model $\mathscr{M}_1$. This is coincident with the simulation results from Norwood and Killick (2018) that WCA is likely to suggest the short memory change-point model even when the truth is the long memory model - the more flexible short memory change-point model just fits the data (or noise) better.

One sensible question to ask is whether we can incorporate model complexity into WCA or the ESC approach in general. Possible ways to incorporate the model complexity is to penalise the more complex model like the information criteria or the entropy-based methods, or to force both models to have the same model complexity or the same number of parameters. A penalty could be added to the simulation part or the classification part to the ESC approach. For example, we may add a penalty to the distance calculated in Step (6) of Algorithm 2.1, but how can the penalty be calculated? Even if we can calculate it for WCA, it will not be easy to generalise to find the penalty for other methods, not to mention generalising it to the deep learning methods. Forcing both models to have the same parameters seems relatively easier to implement, but what is the right number to use? For time series classification with two models that are deemed to have different in model complexity or the number of parameters, say choosing from a one change-point model from a multiple change-point model, it does not seem to make sense to force them to have the same number of parameters (or complexity).

With the reason above, we do not proceed further to incorporate the complexity to WCA or the ESC approach in this thesis. Instead, we would like to see if some simple methods that take into account both the models fit and the model complexity like BIC, can perform better than a more complicated method using the ESC approach like WCA, which will be discussed further in Section 2.4.1.

2.3.4 Our approach to further study the ESC approach

The relation between the poor classification performance and ignorance in the model complexity discussed in Section 2.3.3 is just a hypothesis. Here we outline our approaches to study the issue further and verify the claims we made via simulation:

- Examine the model complexity of $\mathscr{M}_1$ and $\mathscr{M}_2$: Fit WCA under different simulation settings and examine the model fit (in terms of mean squared error, MSE) and the number of parameters in the short memory change-point $\mathscr{M}_1$ and the long memory models $\mathscr{M}_2$ in Norwood and Killick (2018). We want to see if the fitted models under the short memory change-point model specification $\mathscr{M}_1$ in WCA are indeed more flexible and have more parameters than the long memory model $\mathscr{M}_2$ in WCA.
- Study the performance of WCA in the presence of a more flexible but misspecified model: Consider a more flexible, but misspecified model $\mathscr{M}_3$ under the same simulation settings as above. The details of $\mathscr{M}_3$ shown in Section 2.5.2. $\mathscr{M}_3$ is a short memory change-point model that is more flexible than $\mathscr{M}_1$, with the number of change-points $m \geq 2$. This means $\mathscr{M}_3$ is likely to have more parameters than $\mathscr{M}_1$ and this exaggerates further the difference in model complexity in the model considered. As we suspect that WCA partials to models that are more flexible and with more parameters, we would like to see if including the flexible $\mathscr{M}_3$ will worsen the performance of WCA under the long memory settings, and how the presence of flexible model specifications without

taking into account the model complexity will affect the classification performance of the ESC approach.

The simulation settings used to investigate and the corresponding results are in Section 2.5.

2.4 Our proposed methods

2.4.1 Proposal 1: information-criterion-based model identification

Not being able to take into account the model complexity is one of the possible issues of the ESC approach. We therefore propose identifying time series model using the "information criterion method": the information criterion values are calculated for the fitted models from both the short memory change-point and the long memory model specifications, and the model specification with the lowest information criterion value is selected. We suspect that WCA, a relatively complex method using the estimation-simulation-classifier but does not explicitly take into account the model complexity, can be outperformed by such simple methods which take into account both quality of fit and the model complexity.

As in our scenario the noise is i.i.d normal, BIC (or sSIC with $\alpha = 1$ and $c = 1$) is suitable to be used here. To evaluate the performance of BIC, in Section 2.5 we apply it to time series under the same simulation settings from Norwood and Killick (2018) and see if it outperforms WCA. In the simulation we also consider other variations of BIC which the weight on the number of change-points is higher than other free parameters like BIC2 suggested by Kurozumi and Tuvaandorj (2011) and BIC3 proposed by Hall et al. (2013a) to see if a higher penalty on the model complexity will improve or worsen the model identification performance. We do not consider methods that take into account the location of the change-points in this chapter. The BIC methods are implemented in R and will be available online via https://github.com/christineyuen/ESC.

Note that we are not suggesting that BIC or some versions of it can perform well in distinguishing the short memory change-point models and the long memory models in general settings. Rather, BIC or some other appropriate information criteria should at least be considered when determining the suitable model for a given data or treated as a baseline when comparing with other methods in the numerical experiments.

2.4.2 Proposal 2: simulation-based model identification using ResNet

The main motivation of this chapter is to find out whether deep learning methods can be used for model selection via the ESC approach. In Section 2.3 we have studied the potential problems with the ESC approach through WCA. While the poor simulation performance from Norwood and Killick (2018) highlights the potential drawback with the ESC approach, it does not necessarily mean that the ESC approach can never be useful. The problems we have discussed so far is related to the bias on the simulated “training data”, which is about the “estimation” and the “simulation” parts of the approach. If the classifier is very good, it may still be able to classify well despite the simulated “training data” are not very good. Given the good performance of deep learning methods (especially ResNet) on classifying time series in Wang et al., 2017 and Fawaz et al. (2019), using the deep learning methods for model identification via the ESC approach may still provide good performance on model identification.

With the impressive classification performance of ResNet in literature, in this section we propose using ResNet with the ESC approach for model identification. The details of our proposed procedure are in Algorithm 2.2. We call the proposed method “ES-ResNet”, with E and S stand for estimation and simulation. ES-ResNet is implemented and will be available online via https://github.com/christineyuen/ESC.

Algorithm 2.2 ES-ResNet: model identification by ResNet via ESC

```
Input: Observed time series $\{x_1,...,x_T\}$, model specifications $\mathscr{M}_1$ and $\mathscr{M}_2$, number of
    training data to simulate $M$.
Output: predicted class for the observed time series $\{x_1,...,x_T\}$.
1: Get the best fitted model for $\{x_1,...,x_T\}$ from $\mathscr{M}_1$.
2: Get the best fitted model from for $\{x_1,...,x_T\}$ from $\mathscr{M}_2$.
3: Get $M$ “training data” by simulating data from each of the fitted models from Step (1)
    and (2).
4: Fit the ResNet from Fawaz et al. (2019) using the simulated training data.
5: return Classification label predicted by ResNet on the observed time series.
```

The first 3 steps in the Algorithm 2.2 is the same as Algorithm 2.1. We use the same estimation and simulation methods from Norwood and Killick (2018) but replace the classifier by ResNet. Here the ResNet architecture from Fawaz et al. (2019) is used because Fawaz et al. (2019) have shown that this architecture provides good classification performance on the UCR Time Series Classification Archive, and also its implementation (in Python) is available publicly in the GitHub repository. The only adjustment we made is to reduce the number of epochs to 30 to reduce the computation time. In our simulation settings, the training error reduced to a very low level after the first few epochs and the test classification error from all 22 settings we consider is at most 1.1%. The ResNet does not show signs of underfitting after the reduction in the number of epochs.

To evaluate the performance of the ES-ResNet, in Section 2.5 we apply ES-ResNet to time series under the same simulation settings from Norwood and Killick (2018) and see if it outperforms WCA by using ResNet, which is a much more complicated classifier. We also compare its results with the information criteria methods.

2.5 Numerical study

In this section, we use simulated and real data to evaluate the performance of our proposed methods, compare their performance to WCA and verify the claims we made in Section 2.3. The numerical study first focuses on the BIC methods and verifying the claims. The simulation study on ES-ResNet is then presented in Section 2.5.5 and 2.5.6.

2.5.1 Simulation settings

For fair comparison with WCA from Norwood and Killick (2018), we use all the simulation settings from Norwood and Killick (2018) that are under the model specification of $\mathscr{M}_1$ and $\mathscr{M}_2$. The model parameters of each setting are shown in Table 1. We run each simulation 100 times and consider $T = 128$, $T = 512$ and $T = 1024$ for the original WCA and BIC methods. $T = 512$ and $T = 1024$ are the time series lengths considered by Norwood and Killick (2018), and we consider $T = 128$ as well to evaluate the performance of the methods with a shorter time series length. For WCA with $\mathscr{M}_3$, we only consider $T = 512$. The sample lengths used by Norwood and Killick (2018) are dyadic as the R package `wavethresh` used to get the evolutionary wavelet spectra requires time series to be dyadic length. Norwood and Killick (2018) mention that this can be overcome by padding. For the information criterion methods whether the time series length is dyadic is not a concern, as they do not require the calculation of the evolutionary wavelet spectra.

2.5.2 Methods to compare

Information criterion methods

We use three different versions of BIC: BIC, and BIC which the weight on the number of change-points is higher than other free parameters. This includes:

Setting	Model	Parameters						
		λ	μ	ϕ_1	θ_1	ϕ_2	θ_2	d
1	$\mathscr{M}_1$	0.5	1	0.1	0.3	0.4	0.2	-
2		0.5	2	0.1	0.3	0.4	0.2	-
3		0.5	1	0.1	0.3	0.8	0.2	-
4		0.5	2	0.1	0.3	0.8	0.2	-
5		0.7	1	0.1	0.3	0.8	0.2	-
6		0.7	2	0.1	0.3	0.8	0.2	-
7	$\mathscr{M}_2$	-	-	-0.8	0.6	-	-	0.1
8		-	-	-0.8	0.6	-	-	0.2
9		-	-	-0.8	0.6	-	-	0.3
10		-	-	-0.8	0.6	-	-	0.4
11		-	-	0.1	-0.8	-	-	0.1
12		-	-	0.1	-0.8	-	-	0.2
13		-	-	0.1	-0.8	-	-	0.3
14		-	-	0.1	-0.8	-	-	0.4
15		-	-	0.1	0.8	-	-	0.1
16		-	-	0.1	0.8	-	-	0.2
17		-	-	0.1	0.8	-	-	0.3
18		-	-	0.1	0.8	-	-	0.4
19		-	-	0.6	-0.8	-	-	0.1
20		-	-	0.6	-0.8	-	-	0.2
21		-	-	0.6	-0.8	-	-	0.3
22		-	-	0.6	-0.8	-	-	0.4

Table 2.1 Model parameters for the simulation settings.

- BIC with the weight on the number of change-points is twice as the weight as the other free parameters (Kurozumi and Tuvaandorj, 2011, we refer the method as BIC2)
- BIC with the weight on the number of change-points is three times the weight of the other free parameters (Hall et al., 2013a, we refer the method as BIC3)

Following from Hall et al. (2013a) and Hall et al. (2015), we calculate BIC, BIC2, BIC3 as follows:

$$BIC_\gamma = \log(\widehat{\sigma^2})T + (p + \gamma m)\log(T) \tag{2.20}$$

where $\widehat{\sigma^2} = \frac{\sum_i^T \varepsilon_i^2}{T-(p+\gamma m)}$. $\gamma = 1,2,3$ for BIC, BIC2 and BIC3 respectively. With this calculation, the BIC is calculated in a way different from Yao (1988). The log-likelihood term calculated here is similar to the Equation (2.17) for sSIC from the original paper Liu et al. (1997).

Method from Yau and Davis (2012)

We also include the performance of the hypothesis testing procedure from Yau and Davis (2012) for reader's reference. We will refer the method as "Y&D" throughout the rest of the chapter. The performance of Y&D is directly quoted from the simulation results from Yau and Davis (2012) for the same settings considered by Yau and Davis (2012), which is the same as how it is done in Norwood and Killick (2018). Note that Y&D is a hypothesis testing, so the average "classification rate" for Y&D is in fact the empirical size and power of the testing at a certain significant level. Also, the length of time series considered in the simulation is slightly different ($T = 500$ in Yau and Davis, 2012 for Y&D and $T = 512$ for WCA (and BIC methods) here and for Norwood and Killick, 2018).

Change-point model with more than one change-point

For both WCA and information criterion methods, we consider one extra candidate - a more flexible model specification $\mathscr{M}_3$, which is a short memory change-point model with more than one change-point:

$$X_t \sim (\mu_i + ARMA(\phi_i, \theta_i))\mathbb{I}_{\tau_{i-1} < t \leq \tau_i} \tag{2.21}$$

with $i = 0, ..., m+1$, $m \geq 2$, $\tau_0 = 0$ and $\tau_{m+1} = T$.

We want to use $\mathscr{M}_3$ to see if the presence a more flexible model specification would affect the classification performance of WCA (and other methods). The fitted models from $\mathscr{M}_3$ has no upper limit on the number of parameters.

MSE method

This method selects the model specification that has the lowest MSE. This is a naive method which does not take into account the model complexity and only concerns about the difference in the fitted model and the observed data in the time domain. This method is used to show whether the poor simulated training data imply the poor model identification performance for methods using the ESC approach.

2.5.3 Implementation and specification

WCA

For WCA, the R package `LSWclassify` is kindly provided by Norwood and Killick (2018). The code for fitting and simulating $\mathscr{M}_3$ in WCA is added following the implementation of WCA for $\mathscr{M}_1$ and $\mathscr{M}_2$, but NOT is used instead to select the change-points.

Norwood and Killick (2018) suggest using PELT algorithm from Killick et al. (2012) to generalise WCA for multiple change-points models, but we find using PELT from `LSWclassify` to fit $\mathscr{M}_3$ is computationally too slow. NOT with BIC (or equivalently *sSIC* with $\alpha = 1, c = 1$) under the piecewise constant mean and variance setting is used instead to select the location of the change-points. Note that NOT is not designed for dependent data so the estimated location and the number of change-points may not be accurate. Nevertheless, $\mathscr{M}_3$ is supposed to be a misspecified model and given it has more than one change-point, $\mathscr{M}_3$ should give a good in-sample fit with low MSE even if the change-points are not estimated well. The algorithm of fitting the observed time series to $\mathscr{M}_3$ is in Algorithm 2.3.

A summary of the R packages and functions used for the implementation of the model fitting part of WCA is in Table 2.2. For the pseudo training data generation part of WCA, the summary of R packages and functions used is in Table 2.3.

Note The R function `arfima::arfima` for the ARFIMA fit used by Norwood and Killick (2018) does not always return a stationary fit. Norwood and Killick (2018) discard all the non-stationary ARFIMA fits in their implementation, even if those fits have lower BIC value. There is another R function `forecast::arfima` that fits ARFIMA model that always returns

Algorithm 2.3 Algorithm to get a fitted model from $\mathscr{M}_3$

```
Input: Observed time series $\{x_1,...,x_T\}$, order of ARMA(p,q) $p$ and $q$.
Output: fitted change-point model with more than one change-point and each segment
    between 2 consecutive change-points are ARMA(p,q).
1: Get the solution path of NOT for the time series $\{x_1,...,x_T\}$, under the piecewise constant
   mean and variance setting (S1).
2: Select the set of estimated change-points on the solution path from Step (1) such that
   BIC is minimised.. Denote it as $D^{(1)}$.
3: if the number of change-points in $D^{(1)} > 1$ then
4:     return $D^{(1)}$
5: else
6:     Among all the sets on the solution path from Step (1) that have the number of
       change-points greater than 1, select the set that has the highest threshold as $D^{(2)}$.
7:     return $D^{(2)}$
8: end if
```

Model specification	R package and function	Parameters	Remark
1 change-point with ARMA ($\mathscr{M}_1$)	`forecast::auto.arima`	`max.p = 1, max.q = 1` `ic = "bic", d = 0`	The change-point is chosen to maximise the sum of log-likelihood.
long memory (ARFIMA) ($\mathscr{M}_2$)	`arfima::arfima`	`order = c(i, 0, j)` with $i, j = \{0, 1\}$	Select the model that is stationary and with the lowest BIC.
2+ change-point with ARMA ($\mathscr{M}_3$)	`forecast::auto.arima` `not::not`	`constracts = "pcwsConstMeanVar"`	Follow the Algorithm 2.3.

Table 2.2 R functions used for the model fitting in Step 1 and 2 of WCA.

Model specification	R package and function
change-point with ARMA ($\mathscr{M}_1$)	`stats::arima.sim`
long memory (ARFIMA, $\mathscr{M}_2$)	`fracdiff::fracdiff.sim`

Table 2.3 R functions used for the pseudo training data generation in Step 3 of WCA and data generation in the simulation.

a stationary fit. We have tried using `forecast::arfima` for WCA with $T = 512$, but it does not improve the performance of WCA, so the original implementation from Norwood and Killick (2018) is kept for WCA in our simulation.

Information criteria methods

The fitted models from the Step 1 of WCA are used for the calculation of the information criterion values for the short memory change-point model specification, with the number of parameters, the number of change-points and the conditional residuals are extracted from the fitted models. For long memory model specification, `arfima::arfima` used by Norwood and Killick (2018) does not always return a stationary fit as discussed above. Therefore, `forecast::arfima` is used instead for BIC methods as it returns always return a stationary fit, and with this change both $\mathscr{M}_1$ and $\mathscr{M}_2$ use the same R package `forecast` for model fitting (Table 2.4).

Model specification	R package and function	Parameters	Remark
1 change-point with ARMA ($\mathscr{M}_1$)	`forecast::auto.arima`	`max.p = 1, max.q = 1` `ic = "bic", d = 0`	The change-point is chosen to maximise the sum of log-likelihood.
long memory (ARFIMA) ($\mathscr{M}_2$)	`forecast::arfima`	`max.p = 1, max.q = 1` `ic = "bic", drange = c(0,0.5)`	WCA uses `arfima::arfima`.
2+ change-point with ARMA ($\mathscr{M}_3$)	`forecast::auto.arima` `not::not`	`constracts =` `"pcwsConstMeanVar"`	Follow the Algorithm 2.3.

Table 2.4 R functions used for information criteria methods.

Simulated data

To unify the implementation, the R functions used to simulate time series for the simulation are the same as the functions used in the pseudo training data generation in WCA (Table 2.3).

2.5.4 Simulation results

The classification performance of WCA and the information criterion methods under the simulation settings are shown in Table 2.5. While our results for WCA are not exactly the same as the one reported from Norwood and Killick (2018), the results are similar and we do not see a systematic underestimation or overestimation of the classification performance of WCA between two studies. Also, our discussion below is valid regardless of the set of simulation results used out of the two studies. As there is no difference in the conclusion, and the implementation of WCA used here is from Norwood and Killick (2018), together with the randomness nature in the simulation study and the WCA algorithm, we do not investigate further on the discrepancy of the classification performance of WCA in the two studies.

Below we discuss the simulation results further.

Classification performance when considering only $\mathscr{M}_1$ and $\mathscr{M}_2$

When T is large ($T = 512$ and $T = 1024$), most of the methods perform well under the short memory change-point settings as they correctly classify the given time series as $\mathscr{M}_1$. The only exception is BIC3 for setting 1 when $T = 512$, but its performance is similar to other

Setting	Original																With $\mathscr{M}_3$				
	$T = 128$					$T = 500$	$T = 512$					$T = 1024$					$T = 512$				
	WCA	MSE	BIC	BIC2	BIC3	Y&D	WCA	MSE	BIC	BIC2	BIC3	WCA	MSE	BIC	BIC2	BIC3	WCA	MSE	BIC	BIC2	BIC3
1	85	98	53	12	2	99	100	100	95	80	39	100	100	100	100	85	43	12	94	80	39
2	94	100	86	60	30	95	100	100	100	100	99	100	100	100	100	100	34	7	100	100	99
3	91	100	83	59	37	97	99	100	100	100	99	99	100	100	100	100	57	0	100	100	99
4	97	100	97	83	59	94	100	100	100	100	100	100	100	100	100	100	61	0	100	100	100
5	96	100	88	72	51	94	99	100	100	100	99	100	100	100	100	100	65	2	100	100	99
6	99	100	97	86	67	91	100	100	100	100	100	100	100	100	100	100	63	1	100	100	100
7	11	0	84	97	100	63	34	0	90	100	100	50	1	97	100	100	13	0	90	100	100
8	11	0	76	98	100	97	60	1	91	100	100	82	9	99	100	100	34	1	91	100	100
9	18	2	86	99	100	98	81	4	97	100	100	98	41	100	100	100	63	4	97	100	100
10	22	1	83	99	100	96	89	33	96	100	100	99	86	100	100	100	79	33	96	100	100
11	35	1	83	97	100	-	84	4	90	100	100	88	4	94	99	100	54	0	90	100	100
12	34	3	93	99	99	-	86	8	97	99	100	99	28	99	100	100	53	1	97	99	100
13	37	4	87	99	99	-	89	19	98	100	100	99	63	100	100	100	46	0	98	100	100
14	43	2	92	97	99	-	93	22	99	100	100	100	76	100	100	100	50	0	99	100	100
15	24	4	71	97	100	-	40	5	82	98	100	64	1	84	99	100	9	5	82	98	100
16	29	9	76	98	100	-	63	1	88	96	100	74	2	92	100	100	26	1	88	96	100
17	23	10	83	96	99	-	59	2	93	99	100	81	6	95	99	100	24	2	93	99	100
18	32	6	77	98	100	-	54	6	96	99	100	86	32	97	99	100	27	6	96	99	100
19	9	5	88	98	99	-	39	0	89	99	100	45	1	94	100	100	25	0	89	99	100
20	23	4	89	94	99	-	44	1	95	100	100	65	4	93	100	100	38	0	95	100	100
21	32	6	94	100	100	-	53	3	92	100	100	78	7	99	100	100	48	3	92	100	100
22	45	11	95	97	98	-	52	9	92	98	100	69	20	97	99	100	46	9	92	98	100

Table 2.5 Average classification rate (%) under the simulation settings. Each simulation is run 100 times. The values shown for Y&D is directly quoted from Yau and Davis (2012) and they represent the empirical size and power at 5% significant level.

methods for other short memory settings, and has a better classification rate under setting 1 when $T = 1024$. As in our simulation the errors are normally distributed, large weight on change-point penalises too much on the change-point model. For long memory settings, WCA performs very poorly for some long memory settings like settings 19-22 especially when $T = 512$, as discussed in Section 2.3.1. For BIC methods, all of them outperform WCA under all long memory settings with high classification rate.

When T is small ($T = 128$), BIC2 and BIC3 perform quite poorly for short memory change-point settings and WCA has a very low classification rate for long memory settings. BIC, on the other hand, performs reasonably across all settings.

Although WCA performs poorly and worse than BIC methods for long memory settings, it still performs better than the MSE method. In our simulation settings, the simulated data from the more flexible short memory change-point model are very likely to be more similar to the observed time series in terms of MSE than the simulated data from the less flexible long memory model, but this does not necessarily to be the case in the distance metric that Norwood and Killick (2018) considered. For example in setting 14, the classification rate for WCA is 93% although it is just 22% for MSE when $T = 512$. While the poor performance of WCA in long memory settings raises the alarm to the ESC approach, it does not rule out the feasibility of the approach completely.

Classification performance in the presence of $\mathscr{M}_3$

For WCA, the presence of $\mathscr{M}_3$ worsens the classification performance not just for the long memory setting ($\mathscr{M}_2$), but also for the one change-point short memory setting ($\mathscr{M}_1$). In all settings, the classification rate drops with the presence of the flexible $\mathscr{M}_3$. This suggests that the poor performance of WCA may not be just due to "the variation within the wavelet spectrum of long memory series that could be interpreted as different levels and hence a change-point model would be more appropriate" as interpreted by Norwood and Killick (2018), but the difference in complexity of the model specification considered.

For all versions of BIC, the classification performance is not worsened (or worsened only by very little for BIC in setting 1) in the presence of $\mathscr{M}_3$, as the information criteria methods take into account both how well the model is fitting the data, and the model complexity of the model in terms of the number of parameters.

Number of parameters and how well model is fitted

In order to study whether the $\mathscr{M}_1$ is actually more flexible than $\mathscr{M}_2$, and whether the pseudo training data generated from the fitted models from $\mathscr{M}_1$ are more similar to the observed data than $\mathscr{M}_2$ even when $\mathscr{M}_2$ is the true model specification, we look at the number of parameters and the MSE of the fitted models for $T = 512$. We also calculate the Euclidean distance between the pseudo simulated data generated from the fitted models and the observed data. The results are summarised in Table 2.6 and below we discuss the results further.

Original WCA with $\mathscr{M}_1$ and $\mathscr{M}_2$

For the number of parameters, the fitted models from $\mathscr{M}_1$ has higher average number of parameters than $\mathscr{M}_2$ in all settings, which is what we have suspected in Section 2.3.3. For MSE, $\mathscr{M}_1$ has smaller MSE than $\mathscr{M}_2$ in all settings. Also, the pseudo training data generated by the fitted models from $\mathscr{M}_1$ has a smaller Euclidean distance to the observed data than the pseudo training data generated by the fitted models from $\mathscr{M}_2$ in the time domain in all settings. This means that the pseudo simulated data from $\mathscr{M}_1$ is more similar to the observed data in the Euclidean distance, even if the observed data is long memory (i.e. from $\mathscr{M}_2$).

Note that the average MSE of the fitted models under the model specification $\mathscr{M}_1$ is smaller than 1 under all long memory settings. As the actual $\sigma = 1$, this hints that fitted models with model specification $\mathscr{M}_1$ is overfitting the observed long memory time series. On the other hand, $\mathscr{M}_2$ has MSE higher than 1 in all short memory change-point settings, which hints that $\mathscr{M}_2$ is inadequate to fit the short memory change-point time series in the short memory change-point settings.

Setting	# param (fitted model)			MSE (fitted model)			Distance (pseudo data)		
	$\mathscr{M}_1$	$\mathscr{M}_2$	$\mathscr{M}_3$	$\mathscr{M}_1$	$\mathscr{M}_2$	$\mathscr{M}_3$	$\mathscr{M}_1$	$\mathscr{M}_2$	$\mathscr{M}_3$
1	4.51	3.00	11.01	0.99	1.05	0.96	1.61	1.76	1.56
2	4.34	3.02	12.18	0.99	1.09	0.95	1.61	2.09	1.55
3	4.30	3.04	36.70	1.00	1.15	0.86	2.27	2.48	1.70
4	4.73	3.01	36.07	0.99	1.14	0.85	2.17	2.84	1.74
5	4.01	3.02	22.68	1.01	1.16	0.92	2.06	2.19	1.65
6	4.41	3.04	23.03	1.00	1.16	0.91	2.00	2.47	1.66
7	5.47	4.00	7.77	0.97	0.99	1.07	3.38	4.39	3.51
8	5.67	4.00	7.97	0.98	1.00	1.06	3.16	3.15	3.26
9	6.13	4.00	8.01	0.98	1.00	1.07	2.95	4.39	3.05
10	6.03	4.00	7.81	0.99	0.99	1.07	2.77	2.77	2.87
11	5.11	3.06	14.70	0.97	0.99	0.94	2.01	2.03	1.88
12	5.71	3.14	28.71	0.98	1.00	0.92	2.20	2.26	1.89
13	6.03	3.21	40.43	0.99	1.00	0.89	2.42	2.64	1.93
14	6.49	3.25	52.80	0.98	0.99	0.87	2.75	3.50	2.03
15	4.32	3.22	6.38	0.98	1.00	1.01	1.66	1.66	1.67
16	4.60	3.32	6.32	0.98	0.99	1.01	1.60	1.60	1.60
17	5.05	3.44	6.28	0.98	1.00	1.01	1.53	1.53	1.53
18	5.08	3.40	6.47	0.98	0.99	1.00	1.48	1.49	1.48
19	5.44	3.59	58.35	0.97	1.01	0.97	3.20	6.18	2.40
20	5.61	3.74	62.23	0.98	1.02	1.03	3.74	10.54	2.68
21	5.83	3.92	67.12	0.98	1.01	1.09	4.45	8.73	3.09
22	5.94	3.97	68.21	0.98	1.00	1.18	5.58	7.21	3.75

Table 2.6 Average number of parameters and MSE of the fitted models, and the Euclidean distance between the pseudo simulated training data to the observed data from different model specifications in WCA under the simulation settings with $T = 512$.

The higher number of parameters and lower MSE in $\mathscr{M}_1$ shows that $\mathscr{M}_1$ has higher model complexity and is more flexible than $\mathscr{M}_2$. Under the long memory simulation settings, $\mathscr{M}_1$ is too flexible that it overfits observed time series.

WCA with $\mathscr{M}_3$

For $\mathscr{M}_3$, the fitted models under this model specification indeed have more parameters than $\mathscr{M}_1$ and $\mathscr{M}_2$ on average. It has many more number of parameters than the other two model specifications in all the short memory change-point models, and also the long memory settings 11-14 and 19-22. For MSE, however, its MSE value is greater than 1 and also greater than the MSE values of other two model specifications $\mathscr{M}_1$ and $\mathscr{M}_2$ in a number of settings (e.g. long memory settings 7-10). The high MSE for $\mathscr{M}_3$ usually occurs when the estimated number of change-points are very high or relatively low (e.g. when the number of change-points selected by NOT with sSIC is smaller than 2 and we force the algorithm to select at least 2 change-points from the solution path of NOT). After examining the plots of the difference between the squared residuals of $\mathscr{M}_3$ and $\mathscr{M}_2$ (or $\mathscr{M}_3$ and $\mathscr{M}_1$) against time, we found that the large positive difference in squared residual occurs near the estimated change-points, or in between a short estimated interval (i.e. two consecutive change-points are close to each other). For the Euclidean distance between the pseudo training data and the observed data, $\mathscr{M}_3$ in most cases has smaller distance than $\mathscr{M}_2$, even when the MSE of the fitted model of $\mathscr{M}_3$ is higher than $\mathscr{M}_2$.

2.5.5 Simulation performance of ES-ResNet on model identification

To evaluate the performance of ES-ResNet (Algorithm 2.2) on model identification and compare it with WCA and the BIC methods, the same simulation settings from Section 2.5.1 are used. Due to the long computation time to fit the ES-ResNet, we only consider $T = 512$ and each simulation setting is only run for 72 times (instead of 100 times shown in Table 2.5).

Simulation results

When comparing with WCA, the simulation results show that ES-ResNet performs better than (or the same as) the WCA in all long memory settings (setting 7-22). On the other hand, WCA performs better than (or the same as) the ES-ResNet in all short memory change-point

	ES-ResNet	WCA	BIC	BIC2	BIC3
1	85	100	93	76	42
2	100	100	100	100	100
3	89	99	100	100	99
4	97	100	100	100	100
5	99	99	100	100	100
6	100	100	100	100	100
7	75	32	89	100	100
8	71	61	90	100	100
9	74	79	97	100	100
10	90	89	94	100	100
11	75	82	94	100	100
12	89	88	96	99	100
13	97	90	99	100	100
14	100	92	100	100	100
15	58	46	82	99	100
16	69	64	88	96	100
17	72	58	93	99	100
18	58	53	97	99	100
19	100	38	86	99	100
20	100	49	97	100	100
21	100	53	93	100	100
22	100	54	94	99	100

Table 2.7 Average classification rate (%) under the simulation settings with model specification $\mathscr{M}_1$ and $\mathscr{M}_2$ and $T = 512$. Due to the computation time required for ES-ResNet, the results are only based on 72 simulated data sets for each setting. For fair comparison the performance of WCA and BIC methods shown here is corresponding to the same 72 simulated data sets. The numbers here therefore may not be the same as Table 2.5 which corresponds to 100 simulated data sets.

settings (setting 1-6). ES-ResNet generally performs better in the short memory change-point settings than the long memory one, but the difference in classification performance is not as big as WCA. In general, ES-ResNet performs better than WCA, but ES-ResNet still performs quite poorly on some long memory settings like setting 15 and 18.

When comparing the performance of ES-ResNet with BIC methods, BIC performs better than ES-ResNet in both the short memory change-point and the long memory settings, except for setting 19-22 which the ES-ResNet performs better than BIC. For BIC2 and BIC3, they outperform ES-ResNet except for setting 1.

2.5.6 Empirical study

Here we consider inflation data. We retrieve the change in the Consumer Price Index (CPI) from previous year from International Monetary Fund (IMF) via DBnomics on Group of Seven (G7) countries: Canada, Germany, France, the UK, Italy, Japan and the US. Figure 2.4 shows the time series. The data frequency is quarterly.

For the model specification to consider, we follow Norwood and Killick (2018) and use $p, q \leq 4$ instead of $p, q = 1$ for $\mathscr{M}_1$ for the long memory model. For the short memory change-point model we consider the number of change-point is ≥ 1 (i.e. union of $\mathscr{M}_2$ and $\mathscr{M}_3$) with $p, q \leq 4$ instead of $p, q = 1$. Here we use PELT for multiple change-point selection for both WCA and information criteria methods.

Empirical study results

WCA selects the short memory change-point model for all the inflation time series. This is anticipated as the empirical study from Norwood and Killick (2018) show that WCA returns a short memory change-point classification for the US quarterly inflation data. ES-ResNet also selects the short memory change-point model for all the inflation time series.

BIC methods, however, select the long memory model for all inflation time series. It is unsurprising that the two methods do not agree with each other. In literature, both long memory model and short memory change-point models have been used to model inflation.

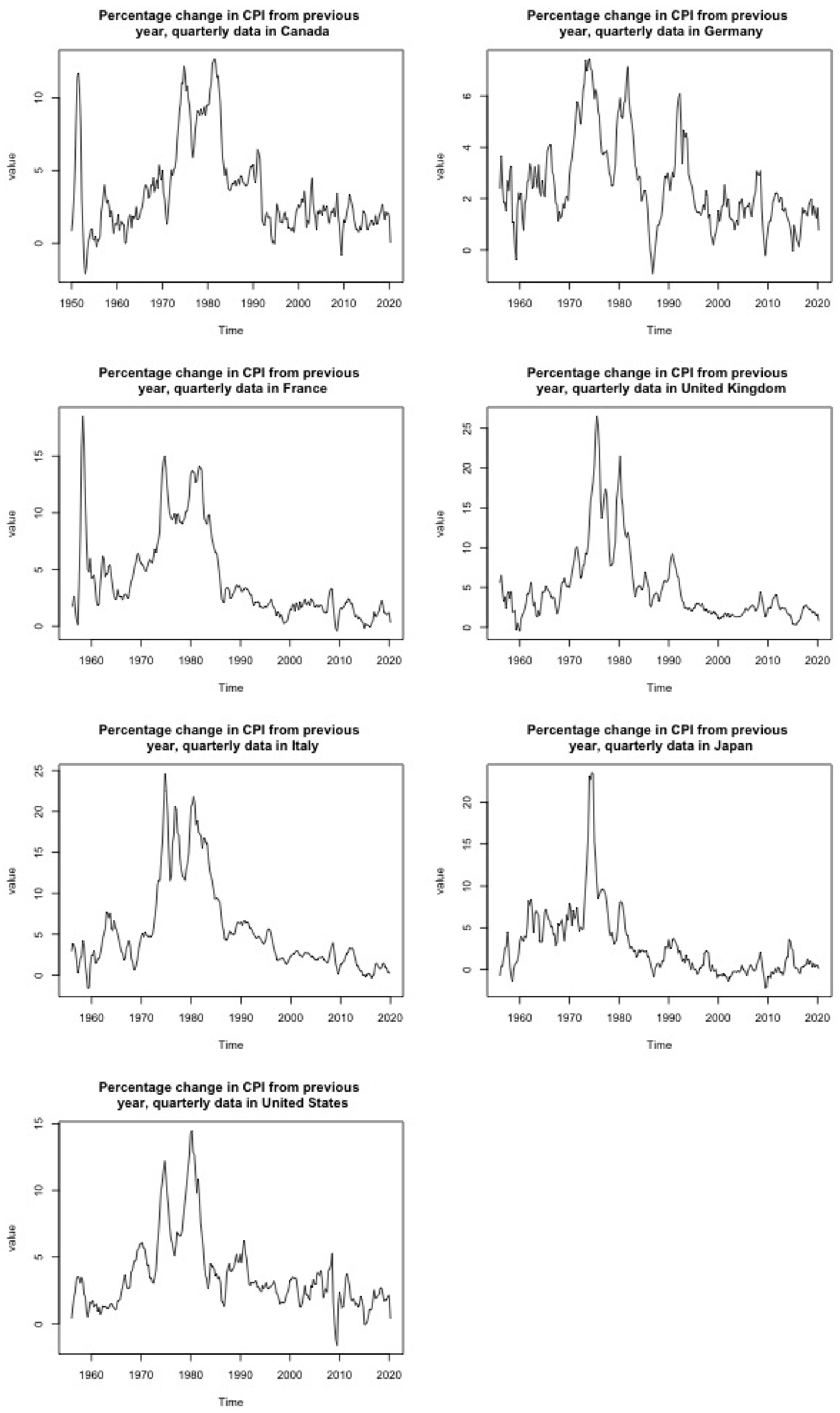

Fig. 2.4 Percentage change in the Consumer Price Index (CPI) from previous year (quarterly data) in G7.

Hsu (2005) also study the G7 inflation rates, although monthly data is used in their analysis. Their empirical results show that for Germany and Japan the long-memory appeared in the data may due to change-points and for other countries the inflation rates may have both long memory and change-points. Some other literature like Song and Shin (2015) and Hassler and Meller (2014) considers some hybrid models like change-point models with long memory in each segment to model inflation data.

2.6 Discussion and conclusion

In this chapter we explore the potential of the estimation-simulation-classification (ESC) approach for better time series model identification. We study the method proposed by Norwood and Killick (2018), which uses the ESC approach for model identification under the choice of the short memory change-point and the long memory model. We find that their method biases towards methods with higher model complexity, as the number of parameters is not taken into account properly.

The simulation results show that selecting the model specification by minimising BIC provides very good results in distinguishing the short memory change-point models and the long memory models under the simulation settings from Norwood and Killick (2018). This simple method which takes into account the model complexity, performs much better under the long memory settings than the more complex method using the ESC approach from Norwood and Killick (2018).

The poor classification results from Norwood and Killick (2018) do not rule out the use of the ESC approach for time series classification. We have shown that our proposed method ES-ResNet (ResNet with the ESC approach) provides reasonable performance and outperforms WCA in model identification, although it is not able to outperform the information criteria methods like BIC. Nevertheless, the poor simulation results on WCA remind us the possible issue when using the simulated 'training' data with the model complexity not taken into account properly. How this can be done on the ESC approach is another research topic.

Note that in this chapter we are not claiming that BIC or some versions of it can perform well in distinguishing the short memory change-point models and the long memory models in general settings. We notice for example how the number of parameters is counted and how the variance is calculated may affect the performance of identifying the right model specification. However, BIC or some other appropriate information criteria should at least be considered when determining the suitable model for a given data or they should at least be treated as a baseline when comparing with other methods in the numerical experiments.

Chapter 3

Forecasting time series with structural breaks and dependent noise using NOT

3.1 Introduction

In this chapter we focus on the forecasting performance on the time series in the following form:

$$X_t = f_t + \varepsilon_t \tag{3.1}$$

where f_t is a deterministic signal and ε_t is some stochastic noise. The signal f_t potentially has change-points. For example, f_t can be piecewise-constant or piecewise-linear. The noise ε_t is assumed to be dependent (e.g. in the form of ARMA). The structure of the dependent noise can also be changed at the change-points. The objective in this chapter is not to find the true model / all the change-points, but to find a change-point model with good forecasting performance in terms of the one-step ahead out-of-sample mean squared forecast error (MSFE). We assume no changes occur during the forecasting period.

We consider the Narrowest-Over-Threshold (NOT, Baranowski et al., 2019) as it is a generic and flexible methodology for change-point detection. We extend the use of

NOT for prediction on time series with dependent noise by proposing some new methods that make use of the change-points detected on the NOT solution path. We compare the prediction performance of the newly proposed procedures to the original NOT, the no-break model (i.e. model fitted using all available data) and some robust methods, and see if the newly proposed procedures provide some advantages in terms of prediction on the simulated and real data, and in which situations the newly proposed methods are preferred. The newly proposed methods are implemented in R and will be available online via https://github.com/christineyuen/NOT-ARMA.

The rest of the chapter is organised as follows. In Section 3.2 we review the related literature. In Section 3.3 we propose the new procedures incorporating the dependent noise structure into NOT. In Section 3.4 we suggest new methods to select the threshold of NOT that aims for better prediction. In Section 3.5 we evaluate the prediction performance of the newly proposed methods using the real and simulated data. We conclude the chapter with a discussion in Section 3.6.

3.2 Literature review

3.2.1 Time series forecasting on data with potential change-points

For time series that are suspected to have structure breaks or change-points, a straightforward approach to forecasting such time series is first detecting the most recent break and then use only the post-break data to estimate the forecasting model. Such a strategy implicitly assumes the observations before the latest change-point are not useful and we can safely discard the data. However, it is shown both theoretically and numerically such an approach may not provide an optimal forecasting performance (e.g. Pesaran and Timmermann, 2007, Pesaran and Timmermann, 2005). For example, even if the change-point can be detected accurately (or the location of the change-point is given), Pesaran and Timmermann (2007) show that theoretically the post-break model needs not to be optimal for forecasting when the time series is under multivariate regression model with structural breaks and exogenous regressors. Similar conclusion is drawn by Pesaran and Timmermann (2007) on bivariate VAR(1) (See Equation (3.2) in the review for Pesaran and Timmermann, 2007 for the definition of bivariate VAR(1)) time series with structural breaks via simulation. The objective of accurately detecting a set of change-points does not necessarily align with the goal of getting a model with the best forecasting performance.

In literature, many different methods have been proposed for forecasting time series with potential breaks. Eklund et al. (2010) argue that forecasting strategies can be summarised into two categories:

- Methods that monitor the changes and adjust the fitted model once a change-point has been detected.
- Methods that do not attempt to detect the change-points and instead use "robust" forecasting strategies which essentially downweight the older data as they are less relevant for the current and future prediction.

Below we review the theoretical results on forecasting time series with potential change-points, and different strategies proposed in literature. We mainly focus on the literature that works on the settings similar to ours, i.e. time series with deterministic change-points in the signal, and the noise has dependent structure. Nevertheless, we also review the works that are on some other related settings, as they provide some good insights on the advantages and shortcomings of different strategies (which usually fall into one of the two categories of strategies suggested Eklund et al., 2010).

Theoretical results and strategies proposed by Pesaran and Timmermann (2007) on multivariate regression model

Pesaran and Timmermann (2007) consider time series under the multivariate regression model with one or more change-points in the regression parameters β. They show theoretically that for univariate regression model with one change-point:

$$y_{t+1} = \begin{cases} \beta_1 x_t + \sigma_1 \varepsilon_{t+1} \text{ for } 1 \leq t \leq \tau \\ \beta_2 x_t + +\sigma_2 \varepsilon_{t+1} \text{ for } \tau < t \leq T \end{cases}$$

and assuming both the location and the size of the change-point is known, the regressor x_t is strictly exogenous and the parameters are estimated via ordinary least squared (OLS), it is optimal to use some pre-break data to estimate the parameters. To minimise the conditional MSFE, the optimal ratio of the pre-break observations useed is data dependent and it is higher if:

- the break of the mean parameter is smaller,
- the variance parameter increases at the point of the break ($\sigma_2 > \sigma_1$),
- the post-break window size $T - \tau$ is small.

Hence, including pre-break data for model fitting can be beneficial for future forecasting. Intuitively speaking, while including pre-break observations can incur bias, it can reduce the variance. By trading off the bias and variance, one can potentially improve the forecasting performance by including some pre-break observations.

Pesaran and Timmermann (2007) consider the following methods for forecasting:

- Post-break window: Estimate the change-points and use only observations after the last estimated change-point for forecasting.
- Trade-off: Calculate the optimal amount of pre-break observations used based on the theoretical results from Pesaran and Timmermann (2007). The calculation depends on the parameters like the size and the location of the change-point which are unknown, so the estimated values of these quantities are used instead.

- Cross-validation: Use the last $\tilde{\omega}$ samples to calculate the out-of-sample MSFE with respect to different starting points of observations used. Select the starting point with the smallest out-of-sample MSFE. In the simulation settings in Pesaran and Timmermann (2007), they use $\tilde{\omega} = 25$ when $T = 100$ and $\tilde{\omega} = 50$ when $T = 200$.

- Inverse MSFE weighted average: Take the weighted average of the forecasts with different starting points of observations used. The weights are set as the inverse of the out-of-sample MSFE calculated from the method above.

- Simple average combination / Averaging across estimation windows (AveW): Take the simple average of the forecasts with different starting points of observations used. This method is studied further by Pesaran and Pick (2011).

The last three methods do not require the detection of the location and the size of the change-points. Nevertheless, an estimated (last) change-point $\hat{\tau}$ can also be incorporated with these three methods by considering only the starting points before or at $\hat{\tau}$, as all post-break observations should be used for forecasting.

Pesaran and Timmermann (2007) demonstrate the performance of the methods above through simulation. They consider the bivariate VAR(1):

$$\begin{pmatrix} y_t \\ x_t \end{pmatrix} = \begin{pmatrix} \mu_{y_t} \\ \mu_{x_t} \end{pmatrix} + A_t \begin{pmatrix} y_{t-1} \\ x_{t-1} \end{pmatrix} + \begin{pmatrix} \varepsilon_{y_t} \\ \varepsilon_{x_t} \end{pmatrix} \tag{3.2}$$

with the coefficient matrices and variance can be changed at the change-points, but the unconditional (or long-run) mean is set so that it is not affected by the change-points. Forecasting using all the observations is used as the baseline. Simulation results from Pesaran and Timmermann (2007) show that, which method provides the best forecasting performance depends on the simulation setting, and also the size of the validation window $\tilde{\omega}$. In general the trade-off method performs worse than the post-break window method in the simulation, as the trade-off method does not only require the estimation of the change-point location but also the estimation of the parameters before and after the estimated change-point. In general cross-validation approach with the use of the estimated change-point seems to provide a robust prediction.

Results from Pesaran et al. (2013)

Pesaran et al. (2013) consider forecasting under both continuous and discrete structural breaks. In their work, breaks are continuous if parameter change occurs in every period by a relatively small amount and they are discrete if the parameters only change in a small number of distinct time points. Here we only review the results for discrete change-points as we only consider the discrete change-points setting in this chapter.

Pesaran et al. (2013) first consider the following model for discrete break:

$$y_t = \beta_t + \sigma_\varepsilon \varepsilon_t$$

where

$$\beta_t = \begin{cases} \beta_{(1)} \text{ for } 1 \leq t \leq \tau \\ \beta_{(2)} \text{ for } \tau < t \leq T \end{cases}$$

and ε_t is i.i.d (0,1). i.e. the time series considered has piecewise constant signal with i.i.d noise.

The forecast considered by Pesaran et al. (2013) is in the form $\hat{y}_{T+1} = \hat{\beta}_T(w)$ where

$$\hat{\beta}_T(w) = \sum_{t=1}^{T} w_t y_t$$

with $\sum_{t=1}^{T} w_t = 1$. Pesaran et al. (2013) show that, given the size of change and the location of the change-point, and assume there is only one change-point, the optimal weight for one step ahead forecast is:

$$w_t = \begin{cases} w_{(1)} = \frac{1}{T}\frac{1}{1+Tb(1-b)\lambda^2} \text{ for } 1 \le t \le \tau \\ w_{(2)} = \frac{1}{T}\frac{1+Tb\lambda^2}{1+Tb(1-b)\lambda^2} \text{ for } \tau < t \le T \end{cases}$$

i.e. $w_{(2)} = w_{(1)}(1+\tau\lambda^2)$, where $\lambda = (\beta_{(1)} - \beta_{(2)})/\sigma$ and $b = \tau/T$.

The above results show that the optimal weight is constant across the same segment but different between segments. The result implies that given the size of change and the location of the change-point, the alternative forecasting methods like post-break window, exponential weight, averaging across estimation windows (AveW, Pesaran and Timmermann, 2007, Pesaran and Pick, 2011) are not optimal. They show that when the break size is small, the difference in forecasting performance between the optimal weight and the post-break model can be large.

Similar results are also obtained for multiple regression model with the slope parameters and the error variance are subject to one change-point with exogenous regressors, and for multiple regressors with multiple breaks, given regressors are stationary process and under other conditions.

As the size of change and the location of the change-points are usually unknown and may be difficult to estimate, Pesaran et al. (2013) propose another optimal weights for both the location and the size of the change are uncertain, and the authors call it the robust optimal weight. The robust optimal weights is calculated by integrating the optimal weights with respect to uniformly distributed change-point locations within the range of possible change-points $s \in [\underline{\tau}, \bar{\tau}]$. For a discrete change-point, the robust optimal weights are:

$$w_t = \frac{w_t^*}{\sum_{s=1}^{T} w_s^*}$$

where

$$w_t^* = \begin{cases} 0 & \text{for } t < \underline{\tau} \\ \frac{-1}{(\bar{\tau}-\underline{\tau})} \log(\frac{T-t}{T-\underline{\tau}}) & \text{for } \underline{\tau} \leq t \leq \bar{\tau} \\ \frac{-1}{(\bar{\tau}-\underline{\tau})} \log(\frac{T-\bar{\tau}}{T-\underline{\tau}}) & \text{for } t > \bar{\tau} \end{cases}$$

Note that only the uncertainty of the location of the change-point is integrated as under the assumptions from Pesaran et al. (2013), the effect of the uncertainty of the location of the change-point is in the order T^{-1}, but the effect of the uncertainty of the size of the change in slope and variance are in the order T^{-2} and T^{-3}. If there is no prior knowledge about the range of the possible change-points, $\underline{\tau}$ and $\bar{\tau}$ can set to the 1 and $T-1$.

The simulation results in Pesaran et al. (2013) show that under the setting with one change-point in mean and no change in variance, which method provides the best forecasting performance depends on the simulation settings. For example, when the size of the change is large, then methods make use of the estimated change-point perform well. When the size of the change is small, however, robust methods (which include the robust optimal weight method) perform better than the methods require the estimation of the change-point.

Results from Pesaran and Timmermann (2005) on AR models

Pesaran and Timmermann (2005) consider AR models with change-points, with the parameters in the AR models are estimated via ordinary least squares (OLS). Similar to Pesaran and Timmermann (2007), Pesaran and Timmermann (2005) show that theoretically, including pre-break observations can reduce variance. Unlike the case with exogenous regressors in Pesaran and Timmermann (2007), including pre-break observations may reduce bias as well. It is because for an AR model with parameters estimated via OLS, there is small-sample bias in the estimates of the parameters. Including pre-break data in some situations can reduce such bias, as long as there is no change in mean (or intercept). Therefore, including pre-break data in the estimation of AR models may simultaneously reduce the bias and the variance of the forecast errors. Pesaran and Timmermann (2005) argue that this theoretical result may explain why empirically it is often difficult to improve forecasting performance over the model using all observations.

Pesaran and Timmermann (2005) consider the following methods:

- Expanding window: All available observations are used for forecasting.
- Rolling window: The last M available observations are used for forecasting. The value of M is set by the authors as 25 and 50 in the simulation.
- Post-break window: Only observations after the most recent estimated change-point are used for forecasting.

While Pesaran and Timmermann (2005) consider several settings with changes in different AR parameters, the only setting that involves the change in signal (which is the focus of this chapter) is the setting with change-points in mean. The simulation results in Pesaran and Timmermann (2005) show that with change in mean, the expanding window (i.e. the no-break model) provides the best forecasting performance in their single break and the mean reversion settings. The post-break window method also performs well. This contrasts with the results on other simulation settings with change-points in other AR parameters like change in the slope parameter. For the simulation settings with change-points in other AR parameters, using only the observations after the most recent estimated change-point often provide relatively poor forecasting performance.

Forecasting methods proposed by Eklund et al. (2010)

Eklund et al. (2010) focus on situations where change-points may occur during the forecasting period. For time series with the presence of a detected recent change-point, Eklund et al. (2010) propose a “monitoring” approach which provides a forecast that is some weighted average of the forecast from the no-break model and the post-break model. The strategy can be summarised as follows for time series with at most one change-point:

- Step 1: Find the change-point. Eklund et al. (2010) assume that the change-point occurs at the time when the change-point is detected.
- Step 2: If no change-point is detected, all available observations are used for forecasting (i.e. the no-break model). If a change-point $\hat{\tau}_1$ is detected, then the forecasting is based on both the no-break model and the post-break model as follows:
  - Prior to $\hat{\tau}_1 + \underline{\omega}$: Use the forecast from the no-break model only.
  - Between $\hat{\tau}_1 + \underline{\omega}$ and $\hat{\tau}_1 + \underline{\omega} + \bar{f}$: Use the linear interpolation of the forecasts from the no-break model and the post-break model.
  - After to $\hat{\tau}_1 + \underline{\omega} + \bar{f}$: Use the forecast from the post-break model only.

The parameters $\underline{\omega}$ and $\bar{f}$ are specified by the authors in their simulation.

Eklund et al. (2010) acknowledge that monitoring change-points may be difficult as change-points can be small, occur frequently, and the detection of change-points can have some delay, etc. They then propose three robust methods which the estimation of change-points is not required for change in mean:

- Rolling forecast: Use the last M observations for forecasting, $\tilde{y}_{t+1}(m) = \frac{\sum_{i=t-m+1}^{t} y_i}{m}$ with m is the size of the rolling window.
- Exponential weighted moving averages: Assign exponential weight on observations, with latest observations have higher weights, $\breve{y}_{t+1} = \sum_{i=1}^{t} \lambda(1-\lambda)^{t-i} y_i$ for some λ.
- Forecast averaging over the estimation periods: Combine forecasts using fitted models with all the possible estimation periods (i.e. different starting points are used), $\bar{y}_{t+1} = \frac{\sum_{m=1}^{t} \tilde{y}_{t+1}(m)}{t}$.

Eklund et al. (2010) show both theoretical and simulation results for the three methods they considered. In terms of the theoretical results, Eklund et al. (2010) look into the special case in which there is only change-point in mean and the noise is i.i.d. For deterministic break, which is the case we consider in this chapter, Eklund et al. (2010) conclude that there is no definite answer to which method performs better than others and it depends on the variance and bias trade-off. Such a theoretical result is similar to the one from Pesaran and Timmermann (2007).

For the simulation results, Eklund et al. (2010) consider the AR(1) settings. The parameters of the methods (e.g. the delay rate λ for exponential weight) are set by the authors. The no-break model is used as the benchmark. The results from Eklund et al. (2010) show that which method performs the best depends on the simulation setting (break and parameterisation). They observe that while the forecast averaging approach is not always the best, it often performs better than the full sample benchmark.

Results from Giraitis et al. (2013)

Giraitis et al. (2013) observe that Eklund et al. (2010) do not address how the parameters are set for their methods, and how to take care when the change is not monotonic (e.g. switching to old regime).

Giraitis et al. (2013) suggest new forecasting approaches based on the results from Eklund et al. (2010). They observe that some methods from Eklund et al. (2010) require setting the parameter for downweighting, and Giraitis et al. (2013) suggest choosing the parameter by minimising the out-of-sample MSFE. Giraitis et al. (2013) also consider the residual methods, which can be summarised as follows:

- Step 1: Fit the time series to AR(1).
- Step 2: Forecast the residual by parametric or nonparametric method, for example using exponential weight.

While Giraitis et al. (2013) consider several time series settings, the only setting relevant to this chapter is time series with change in mean. The simulation results show that for noise with AR structure, only the nonparametric residual method outperforms the no-break AR model in both two settings considered by Giraitis et al. (2013).

Results from Kley et al., 2019

Kley et al. (2019) focus on locally stationary time series, like time-varying AR model. While their work is not about time series with change-points, their observations on the relations between prediction and model specification are still related to the problem in this chapter. They find that even if the process generating the time series is time-varying, in some cases estimating the process by treating it as constant may provide a better prediction performance. They argue that "the wrong model" may be preferred when the objective of fitting a model is for better prediction. They propose a method that selects from different procedures based on the empirical MSFE. They observed from their simulation that when a large amount of data is not available it is often advisable to use a procedure derived from a simpler model.

Remark on the time series forecasting methods reviewed

In literature we review above, Pesaran and Timmermann (2007), Pesaran et al. (2013), Pesaran and Timmermann (2005) and Eklund et al. (2010) show theoretically that for time series with potential change-points, using some data prior to the last change-point may provide better prediction performance than using only the post-break data. The objective of accurately detecting a set of change-points does not necessarily align with the goal of getting a model with the best forecasting performance. Pesaran and Timmermann (2007) and Giraitis et al. (2013) propose using cross-validation to select the parameters (e.g. window size and last starting point). While the best prediction methods often depend on the settings, the relatively robust performance of the cross-validation method from Pesaran and Timmermann (2007) make us wonder if we can incorporate the cross-validation idea into NOT to improve its prediction performance. We want to see if using cross-validation to select a change-point model from the NOT solution path can improve the prediction performance of NOT. This is different from Pesaran and Timmermann (2007) and Giraitis et al. (2013), as the forecasting from their cross-validation method does not correspond to a change-point model.

3.2.2 Review of NOT with dependent noise and prediction

Here we review NOT only in terms of prediction and on time series with dependent noise, as the general review on NOT has already been presented in Section 2.2.4.

In Baranowski et al. (2019), the algorithm proposed and the corresponding theoretical properties studied are about the estimation of the number and the location of the change-points. While an accurate estimation of the number and the location of change-points may facilitate the forecasting of future observations, a good estimation of change-points does not necessarily imply good prediction performance directly.

The Corollary 1 and 2 in Baranowski et al. (2019) shows that under setting (S1) and (S2), if the noise is i.i.d or stationary with short memory Gaussian process, then NOT is consistent in selecting the right number and the location of change-points. However, the numerical study from Baranowski et al. (2019) shows that with finite samples, NOT with sSIC may not perform well with non-i.i.d noise, even if the signal is under the right signal setting. For example, for the 2 simulation settings under (S1) considered by Baranowski et al. (2019) with noise follows AR(1) with $\phi = 0.3$, the simulation results show that the estimated number of change-points is likely to be different from the true number of change-points.

For the real data, we observe that NOT with sSIC tends to return a large number of estimated change-points when we assume the underlying signal is piecewise linear. The detection of a large number of change-points by NOT may because the real data is not actually piecewise linear, and the error is not actually independent as assumed by NOT. Figure 3.1 shows an example NOT fit using piecewise continuous linear contrast function with sSIC on one of the real data sets used in our empirical study in Section 3.5. There are some observable “breaks” in the real data with some seemingly but not exactly linear trends.

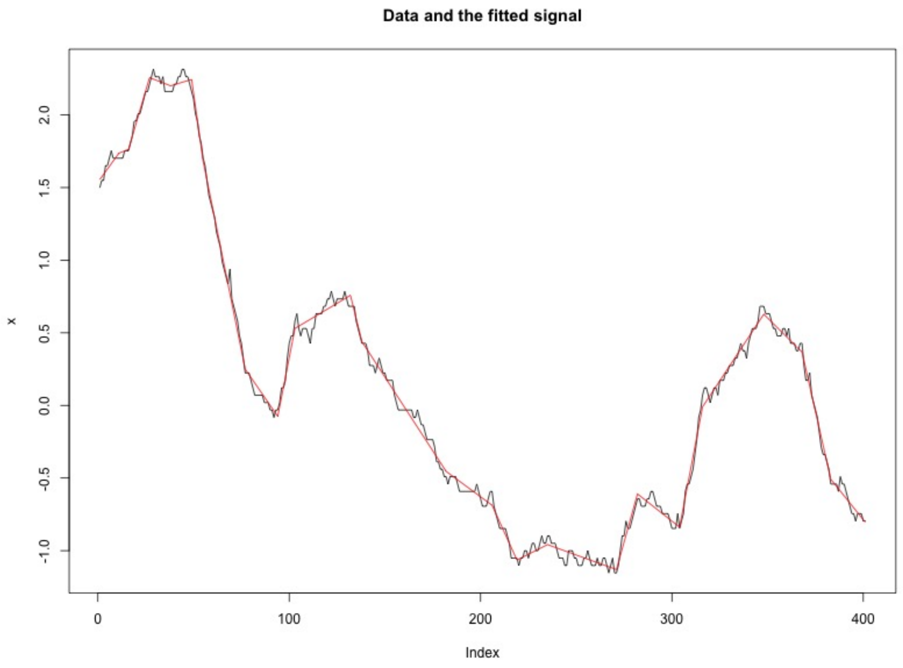

Fig. 3.1 Real time series (black in coloured version) and NOT fit using sSIC (piecewise linear, red in coloured version)

While NOT with sSIC does a very good job in capturing the structure presented in the data using piecewise linear continuous function, it seems to retain too many details. For example from time point 130 to 210 there are three change-points detected by NOT with sSIC, and two change-points from 350 to the end of the time series. The non-linear structure in the real data requires more change-points in order to capture the dynamic of the real data. While those local change-points may still be useful for interpretation, they may not be useful for prediction.

3.3 Incorporating dependent structure into NOT

In order to make NOT applicable to forecasting time series with dependent noise, we first propose two new NOT procedures that incorporate ARMA fit.

Algorithm 3.1 NOT-ARMA-1: Fitting ARMA on each segment between change-points estimated on the NOT solution path

```
Input: Observed data $\mathbf{x}_T = x_1,...,x_T$, NOT contrast function.
Output: Solution path of NOT with fitted change-point models with ARMA fitted at each
    segment and the corresponding thresholds.
1: Get the NOT solution path using Algorithm 2 of NOT in Baranowski et al. (2019) and
   the given contrast function. Record the thresholds $\zeta_T = \{\zeta_T^1,...,\zeta_T^{N_T}\}$ and the sets of
   change-points $\hat{\mathscr{T}}_T = \{\hat{\mathscr{T}}_T(\zeta_T^1),...,\hat{\mathscr{T}}_T(\zeta_T^{N_T})\}$ on the solution path, where $N_T$ is the length
   of the solution path and $\hat{\mathscr{T}}_T(\zeta_T^k)$ is a set of estimated change-points with respect to the
   threshold $\zeta_T^k$.
2: for each set of change-points $\hat{\mathscr{T}}_T(\zeta_T^k)$, $k = 1,...,N_T$ do
3:     Fit ARMA on each segment of time series between two consecutive estimated
       change-points. For piecewise linear signal, the time index of the observations is used as a
       regressor for the estimation of the trend of the signal. Denote such a fitted change-point
       model as $\hat{M}_T^{(1)}(\zeta_T^k)$.
4: end for
5: return $S_T^{(1)} = \{\hat{M}_T^{(1)}(\zeta_T^1),...,\hat{M}_T^{(1)}(\zeta_T^{N_T});\ \zeta_T\}$.
```

3.3.1 NOT-ARMA-1: fit ARMA into each segment of time series

The first proposed procedure is NOT-ARMA-1, which fits ARMA to each segment between two change-points estimated by NOT on its solution path. The details of the procedure are shown in Algorithm 3.1. Note that in Algorithm 3.1, only the change-points estimation from NOT is used but not the signal estimated. The signal is instead estimated by ARMA via the intercept for piecewise constant signal, or the intercept and the slope of the regressor for piecewise linear signal.

Algorithm 3.2 NOT-ARMA-2: Fitting ARMA on the whole time series after the signal estimated by NOT is removed

```
Input: Observed data $\mathbf{x}_T = x_1, ..., x_T$, NOT contrast function.
Output: Solution path of NOT with fitted ARMA models with the corresponding thresholds.
1: Get the solution path of NOT using Algorithm 2 of NOT in Baranowski et al. (2019)
   and given contrast function. Record the thresholds $\zeta_T = \{\zeta_T^1, ..., \zeta_T^{N_T}\}$ and the sets of
   change-points $\hat{\mathscr{T}}_T = \{\hat{\mathscr{T}}_T(\zeta_T^1), ...\hat{\mathscr{T}}_T(\zeta_T^{N_T})\}$ on the solution path, where $N_T$ is the length
   of the solution path and $\hat{\mathscr{T}}_T(\zeta_T^k)$ is a set of estimated change-points with respect to the
   threshold $\zeta_T^k$.
2: for each threshold $\zeta_T^k$, $k = 1, ..., N_T$ do
3:     Estimate the signal $\hat{\mu}_T(\zeta_T^k)$ by NOT.
4:     Calculate the residuals by removing the estimated signal from the original data:
           $\mathbf{e}_T(\zeta_T^k) = \mathbf{x}_T - \hat{\mu}_T(\zeta_T^k)$
5:     Fit ARMA on the residual $\mathbf{e}_T(\zeta_T^k)$.
6:     Record the fitted model, which is the fitted ARMA plus the estimated signal. Denote
   such a fitted model as $\hat{M}_T^{(2)}(\zeta_T^k)$.
7: end for
8: return $S_T^{(2)} = \{\hat{M}_T^{(2)}(\zeta_T^1), ..., \hat{M}_T^{(2)}(\zeta_T^{N_T});\ \zeta_T\}$.
```

3.3.2 NOT-ARMA-2: fit ARMA on the whole time series after the signal estimated by NOT is removed

We also consider another algorithm NOT-ARMA-2, which makes use of the signal estimated by NOT. The estimation of the ARMA model is done after the signal estimated by NOT is removed. The details of the procedure are in Algorithm 3.2. Note that for Step (5) of Algorithm 3.2, the ARMA is fitted on the whole time series but not on each segment. The NOT-ARMA-1 and NOT-ARMA-2 are implemented in R and will be available online via https://github.com/christineyuen/NOT-ARMA.

The Algorithm 3.1 and 3.2 only return a solution path. The final model can be selected by a given threshold value or find the model that has the minimum BIC/sSIC value. As our goal in this chapter is to provide a model with good prediction performance, in the next section we consider a prediction-driven way to select the final model on the solution path.

Before we continue, we use the notation NOT-ARMA-1(BIC) to denote using BIC to select the final model from the solution path generated from NOT-ARMA-1. Similarly, if we use NOT-CV-min (which we will introduce in Section 3.4) to select the final model from the solution path generated from NOT-ARMA-2, we write such procedure as NOT-ARMA-2(NOT-CV-min).

3.4 New algorithm NOT-CV: Using cross-validation to select the threshold

3.4.1 Introduction

Currently, NOT uses sSIC (or BIC) to select the final change-point model from the solution path. Baranowski et al. (2019) show the consistency of NOT using sSIC in detecting the number and the location of change-points. While NOT with sSIC may be able to estimate the location of the change-points accurately, this does not guarantee good prediction performance, as Pesaran and Timmermann (2007) and other works reviewed in Section 3.2.1 show that the optimal starting point of the time series for forecasting does not necessarily the last change-point. Following from Pesaran and Timmermann (2007) and Giraitis et al. (2013) using cross-validation to select the parameters for prediction performance, we propose two new procedures that make use of cross-validation (CV) to select a threshold on the solution path from NOT-ARMA-1 or NOT-ARMA-2 to attempt to improve the prediction performance. We call the procedure "NOT-CV". While we use the word "cross-validation", in this chapter we only consider the holdout method of cross-validation, which is the method used in literature we review in Section 3.2 (Pesaran and Timmermann, 2007, Giraitis et al., 2013). This is because our observations are time series and other types of cross-validation may alter the time order of the data.

The rationale of using cross-validation to select threshold is as follows: Not all change-points are useful for prediction, and ignoring some small change-points may boost the prediction performance. By selecting the threshold through cross-validation, we hope the algorithm can retain only the change-points that are useful for prediction. The cross-validation is on the thresholds instead of on the change-point locations (or the starting point of the observation window like Pesaran and Timmermann, 2007). Even if the last change-point (or the "optimal" starting point for the observation window) is located within the validation set, it may still be selected.

3.4.2 Cross-validation and time series prediction with possible change-points

Cross-validation for prediction on change-point data is not a new idea. For example, cross-validation is used in Pesaran and Timmermann (2007) to select the starting point of the observation window for model fitting and in Giraitis et al. (2013) to select the decaying factor for exponential weights. The main difference between these approaches and the NOT-CV is that NOT-CV still attempts to use the change-points detected to fit a model for the prediction purpose. In Pesaran and Timmermann (2007), the starting point selected by minimising the validation MSFE is not necessarily the same as one of the change-points detected. In Giraitis et al. (2013), the selection of a down weighting parameter does not even require the estimation of the change-points. On the contrary, the change-points used in NOT-CV must be one of the sets of change-points estimated on the NOT solution path. While our objective is to get a fitted model with good prediction performance, we hope the resulting model can still have the change-point model structure and provide a good interpretation of the structural change in the observed time series.

We consider two different NOT-CV algorithms in this chapter. The first algorithm selects the threshold that minimises the validation MSFE (NOT-CV-min). The second algorithm selects the threshold using the one standard error (SE) rule (NOT-CV-1SE). We observe that NOT-CV-min tends to select quite a lot of change-points. The NOT-CV-1SE uses the one standard error rule, which always selects a larger threshold (often means a smaller set of change-points) than the one from NOT-CV-min. The one standard error rule on validation MSE is a popular rule of thumb used in selecting the tuning parameter for Lasso (Tibshirani, 1996).

Below we give more details about the two algorithms.

3.4.3 NOT-CV-min

NOT-CV-min selects the threshold that minimises the validation MSFE. The details of NOT-CV-min is shown in Algorithm 3.3.

Re-partition of squared error

When fitting with different sets of observed time series x_t, the sets of thresholds associated with different threshold-indexed solution paths from the `not::not` are different. The threshold-indexed solution paths can be thought as a step function mapping the threshold to a set of change-points, with its value (i.e. the set of change-point) only at discrete thresholds $0 = \zeta_T^0 < ... < \zeta_T^N$. Such mapping is data specific and so as the set of discrete thresholds. Therefore, the set of thresholds when fitted using all the data ζ_T is different from the set of thresholds when fitted using only the training data $\zeta_{n'-1}, n' = T - n'_v + 1, ..., T$. In order to use the validation squared error to find the threshold for the final fit, the validation error is re-partitioned to have the same sets of thresholds as the thresholds on the solution path of

Algorithm 3.3 NOT-CV-min: Finding the threshold minimising the validate MSFE

```
Input: Observed data $\mathbf{x}_T = x_1,...,x_T$, the solution path $S_T^{(1)}$ from Algorithm 3.1 (or $S_T^{(2)}$ from Algorithm 3.2. If $S_T^{(2)}$ is used, all the superscript $(1)$ below in this algorithm should be replaced by $(2)$), NOT contrast function, the validation set size $n_v$.
Output: Final fitted model
 1: for i in $1 : n_v$ do
 2:     Set $\mathbf{x}_{n'-1} = x_1,...,x_{n'-1}$ as the training data and $x_{n'}$ as the validation data point, where $n' = T - n_v + i$. Get the solution path $S_{n'-1}^{(1)}$ from Algorithm 3.1 using the training data $\mathbf{x}_{n'-1}$.
 3:     for For each threshold $\zeta_{n'-1}^k \in \zeta_{n'-1}$ from $S_{n'-1}^{(1)}$ do
 4:         Calculate the prediction squared error corresponding to each threshold using the validation data point $x_{n'}$:
                $$[e_{n'}^{(1)}(\zeta_{n'-1}^k)]^2 = (x_{n'} - \hat{x}_{n'}^{(1)})^2$$
            where is $\hat{x}_{n'}^{(1)}$ is the one step ahead forecast from $\hat{M}_{n'-1}^{(1)}(\zeta_{n'-1}^k)$ with respect to threshold $\zeta_{n'-1}^k$.
 5:     end for
 6:     Re-partition the squared error $[e_{n'}^{(1)}(\zeta_{n'-1}^1)]^2,...,[e_{n'}^{(1)}(\zeta_{n'-1}^{N_{n'-1}})]^2$ calculated from Step (3) to get $[\tilde{e}_{n'}^{(1)}(\zeta_T^1)]^2,...,[\tilde{e}_{n'}^{(1)}(\zeta_T^{N_T})]^2$ according to the Algorithm 3.4. The re-partitioning aligns the squared error from Step (4) threshold in the solution path from the input.
 7: end for
 8: Calculate the validation MSFE along the solution path of $S_T^{(1)}$ to get $MSFE_T^{(1)}(\zeta_T^1),...,MSFE_T^{(1)}(\zeta_T^{N_T})$, where
        $$MSFE_T^{(1)}(\zeta_T^k) = \frac{1}{n_v} \sum_{n'=T-n_v+1}^{T} [\tilde{e}_{n'}^{(1)}(\zeta_T^k)]^2$$
 9: Select the largest threshold that has the lowest validation MSFE and denote it as $\zeta_T^*$:
        $$\zeta_T^* = max(\arg\min_{\zeta_T^k} MSFE_T^{(1)}(\zeta_T^k))$$
10: return $\hat{M}_T^{(1)}(\zeta_T^*)$
```

the final fit. The details for the re-partition procedure is in Algorithm 3.4. Basically, linear interpolation is used to construct the validation squared error at the thresholds in ζ_T.

3.4.4 NOT-CV-1SE: find the threshold using the one SE rule

NOT-CV-1SE is similar to NOT-CV-min except the threshold is chosen using the one SE rule. The one SE rule is often used in the Lasso as an alternative to using minimised cross-validation MSE to select the regularisation parameter. For the Lasso, the minimised cross-validation MSE is often used to select the regularisation parameter for the prediction purpose, and the one SE rule is used to select the regularisation parameter for model specification purpose as the one SE rule is less prone to overfitting.

While our focus is on prediction but not model specification, we hope that the one SE rule can reduce overestimating change-points observed the NOT-CV-min and be able to return a better threshold. The details of NOT-CV-1SE is in Algorithm 3.5.

Algorithm 3.4 Re-partition: Matching the thresholds from the validation fit and the solution path fitted on the whole time series

```
Input: Thresholds $\zeta_T^1, ..., \zeta_T^{N_T}$ from the solution path $S_T^{(1)}$ from Algorithm 3.1 (or $S_T^{(2)}$ from Algorithm 3.2. If $S_T^{(2)}$ is used instead, all the superscript $(1)$ should be replaced by $(2)$ in this algorithm), thresholds $\zeta_{n'-1}^1, ..., \zeta_{n'-1}^{N_{n'-1}}$ from the solution path $S_{n'-1}^{(1)}$, squared error $[e_{n'}^{(1)}(\zeta_{n'-1}^1)]^2, ..., [e_{n'}^{(1)}(\zeta_{n'-1}^{N_{n'-1}})]^2$ on validation data point $x_{n'}$.
Output: Re-partitioned validation squared error on the validation data $x_{n'}$ along the solution path $S_T^{(1)}$.
1: for For each threshold $\zeta_T^k$, $k = 1, ..., N_T$ do
2:     Find the thresholds $\zeta_{n'-1}^{(lower)}$ and $\zeta_{n'-1}^{(upper)}$ from $\zeta_{n'-1}^1, ..., \zeta_{n'-1}^{N_{n'-1}}$ such that they are the closest to $\zeta_T^k$ and just cover it:
           $$\zeta_{n'-1}^{(lower)} = max(\{\zeta_{n'-1}^{k'} | \zeta_{n'-1}^{k'} \leq \zeta_T^k\})$$
       and
           $$\zeta_{n'-1}^{(upper)} = min(\{\zeta_{n'-1}^{k'} | \zeta_{n'-1}^{k'} \geq \zeta_T^k\})$$
3:     Calculate the re-partition squared error:
           $$[\tilde{e}_{n'}^{(1)}(\zeta_T^k)]^2 = w[e_{n'}^{(1)}(\zeta_{n'-1}^{(lower)})]^2 + (1-w)[e_{n'}^{(1)}(\zeta_{n'-1}^{(upper)})]^2$$
       where
           $$w = \begin{cases} 1 & \text{if } \zeta_{n'-1}^{(lower)} = \zeta_{n'-1}^{(upper)} \\ \frac{\zeta_{n'-1}^{(upper)} - \zeta_T^k}{\zeta_{n'-1}^{(upper)} - \zeta_{n'-1}^{(lower)}} & \text{otherwise} \end{cases}$$
4: end for
5: return $[\tilde{e}_{n'}^{(1)}(\zeta_T^1)]^2, ..., [\tilde{e}_{n'}^{(1)}(\zeta_T^{N_T})]^2$
```

Algorithm 3.5 NOT-CV-1SE: Finding the threshold using the one SE rule

```
Input: Observed data $\mathbf{x}_T = x_1, ..., x_T$, the solution path $S_T^{(1)}$ from Algorithm 3.1 (or $S_T^{(2)}$
    from Algorithm 3.2. If $S_T^{(2)}$ is used, all the superscript $(1)$ below in this algorithm should
    be replaced by $(2)$), NOT contrast function, the validation set size $n_v$.
Output: Final fitted model
 1: Same as the Step (1)-(8) of Algorithm 3.3 for NOT-CV-min.
 9: Select the largest threshold such that the corresponding average validate MSFE is not
    greater than the minimum MSFE + one SE:
        $\zeta_T^{**} = max(\{\zeta_T^k | MSFE_T^{(1)}(\zeta_T^k) \leq MSFE_T^{(1)}(\zeta_T^*) + se_T^{(1)}(\zeta_T^*)\})$
    where
        $\zeta_T^* = max(\arg\min_{\zeta_T^k} MSFE_T^{(1)}(\zeta_T^k))$
    and $se_T^{(1)}(\zeta_T^*)$ is the standard error of the MSFE with respect to the minimum MSFE.
10: return $\hat{M}_T^{(1)}(\zeta_T^{**})$
```

The NOT-CV-min and NOT-CV-1SE are implemented in R and will be available online via https://github.com/christineyuen/NOT-ARMA.

3.5 Numerical study

3.5.1 Objectives

We evaluate the prediction performance of the newly proposed methods using both real and simulated data. We would like to find out:

- If the proposed methods improve the prediction performance of the original NOT on time series with dependent noise.

- In what settings incorporating the detected changes points (via the proposed methods) can provide a better prediction performance than a fitted model ignoring any potential change-points or the robust methods.

3.5.2 Implementation and specification

Methods to compare

In order to achieve the objectives stated in Section 3.5.1, we compare the performance of the proposed methods with:

- Original NOT with the final model selected by BIC (which is equivalent to sSIC with $\alpha = 1$ used in Baranowski et al., 2019).
- No-break model, which is the ARMA fit with all available data. It serves as the baseline in our simulation.
- Robust methods that do not incorporate the detected changes points:
  - Average across estimation windows: Take the simple average of the forecasts with different starting points of observations using ARMA. This is similar to the AveW considered by Pesaran and Timmermann (2007) and Eklund et al. (2010) except we use the ARMA instead of the VAR model.
  - Rolling window: the last M_1 available observations are used for forecasting using ARMA. This is similar to the rolling window considered by Pesaran and Timmermann (2005) and Eklund et al. (2010), except we use ARMA instead of AR.
  - Optimal break point chosen by cross-validation: Use the last M_2 samples to calculate the out-of-sample MSFE with respect to different starting points of observations to use. Select the starting point with the smallest out-of-sample MSFE.

This is similar to the cross-validation method from Pesaran and Timmermann (2007), except we use ARMA instead of AR.

– Optimal rolling window chosen by cross-validation: Use the last M_3 samples to calculate the out-of-sample MSFE with respect to different rolling window used. Select the rolling window size with the smallest out-of-sample MSFE. This is similar to the rolling window method with the parameter selected by cross-validation by Giraitis et al. (2013), except that in Giraitis et al. (2013) both the window size and M_3 are chosen to minimise the out-of-sample MSFE, and we consider ARMA.

For simulated data for which the true change-points are known, the prediction performance of the "oracle" model is also included. "Oracle" model is the ARMA model fitted using only the post-break observations. For the robust methods except the averaging, they require a given parameter M_1, M_2 or M_3. For simplicity, we set the numbers $M_1 = M_2 = M_3$ and they are equal to the number we use for validation for our methods, which is 15% of the data. Our simulated time series excluding the testing segment are mostly with length of 190 or 340, which means the validation data set is at the length of 29 or 51. These two numbers are similar to the ones used in literature we are referencing to. For example, Eklund et al. (2010) consider M_1 to be 20 and 60, Pesaran and Timmermann (2007) consider M_2 to be 25 or 30 and Giraitis et al. (2013) use M_3 to be 20 or 30. We only consider window sizes that are ≥ 5 to make sure there are enough data points for ARMA to fit.

Prediction performance measure

Prediction performance is measured via empirical MSFE. All MSFE are calculated via one step ahead prediction. For simple comparison, the relative MSFE with respective to the baseline method (i.e. the no-break model) are reported. The relative MSFE for method k is:

$$relative\ MSFE_k = \frac{MSFE_k}{MSFE_{baseline}}$$

The relative MSFE for the baseline method is always 1.

In order to estimate the prediction performance, the last 10 observations are used as the test data.

R implementation

The R function `not::not` is used to fit NOT. For simulated data, the NOT contrast function is set according to the corresponding simulation setting. For example, if the signal in the simulated data is piecewise linear, then the contrast is set to “pcwsConstMean”. For real data, the contrast is set as “pcwsLinContMean” to capture change-point in slope with no jumps.

ARMA in each segment between two consecutive change-points is fitted using the R function `stats::ARIMA`. The order is set to $(1,0,1)$.

For methods that require the calculation of the validation MSFE (i.e. NOT-CV-min, NOT-CV-1SE and some robust methods), the last 15% of the non-test data is set as the validation data.

3.5.3 Simulation study

Simulation settings

Below we consider different simulation settings. We generate observations with 100 realisations of time series using the model specifications below. We consider four sets of simulation settings: simple, settings similar to Baranowski et al. (2019), settings with change in noise structures and settings with exaggerated signal change. Previous work in literature show that which time series prediction methods perform well depends on different settings like the size of jumps and the location of the last change-point, and we want to see how different time series prediction methods behave by considering different sets of settings.

Simple settings

We first consider very simple settings, with the time series has a constant signal with at most one change-point. All the times series have length 200 with noise $ARMA(1,1)$.

- Model 1s (no change): No change-point, $ARMA(1,1)$ with $\phi = 0.1$ and $\theta = 0.3$.
- Model 2s (noise change): $ARMA(1,1)$ changes from $(0.4, 0.2)$ to $(0.1, 0.3)$ at $t = 150$.
- Model 3s (signal change): No change-point in noise but signal changes from mean 1 to mean -9 at $t = 150$.
- Model 4s (signal and noise change): same as Model 3s except the noise structure changes as well from $(0.4, 0.2)$ to $(0.1, 0.3)$ at $t = 150$.
- Model 5s (earlier change-point): same as Model 4s except the change-point is $t = 50$ for both signal and noise change.
- Model 6s (smaller signal change): same as Model 4s except the change in signal is from 1 to 0.
- Model 7s (different noise change): same as Model 6s except the change in ARMA is from $(-0.8, 0.7)$ to $(0.1, 0.3)$.

For all the ARMA models represented in the form of (x, y), the first number in the parenthesis is the AR parameter and the second number is the MA parameter. Figure 3.2 shows one realisation from Model 1s-7s. Some settings like Model 3s, 4s and 5s have a very large jump.

Settings similar to Baranowski et al. (2019)

We consider the simulation settings are similar to the ones in Baranowski et al. (2019), except

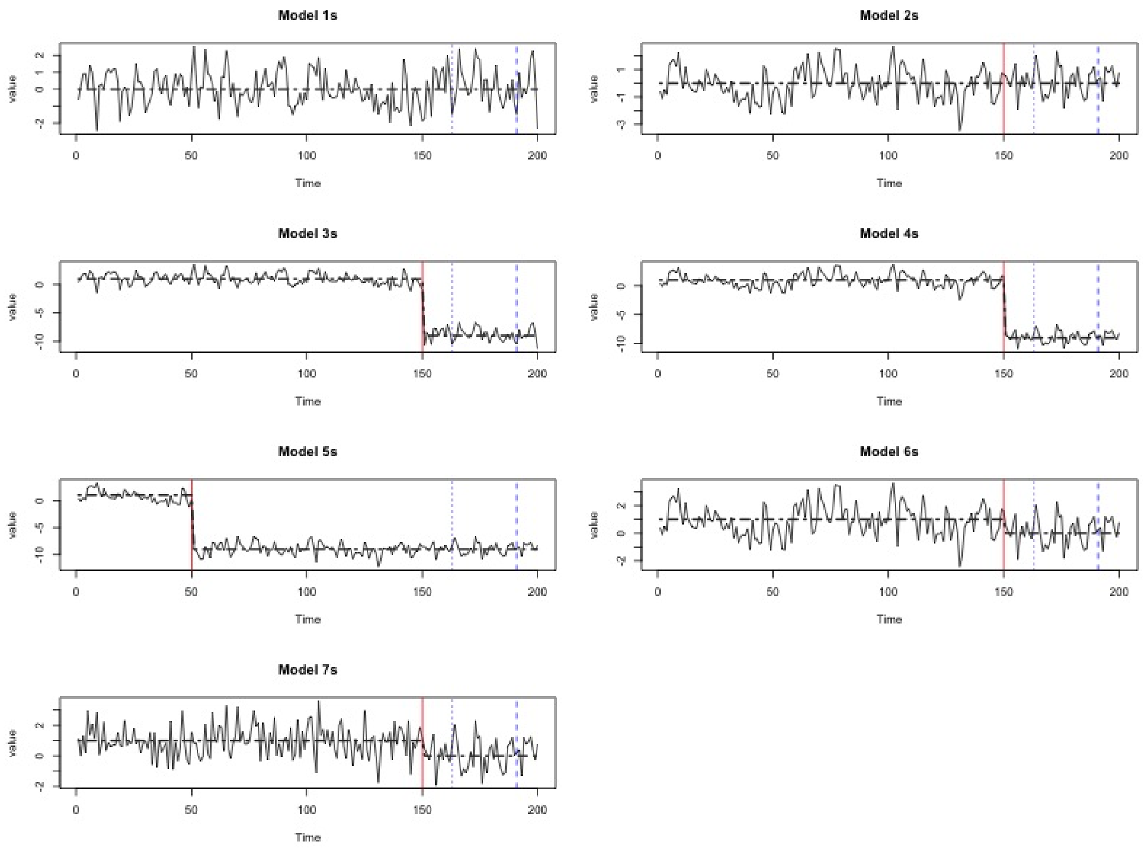

Fig. 3.2 One time series realisation from Model 1s-7s. Solid lines (grey in the coloured version) are the simulated time series and the dotted line (black in the coloured version) are the signals. Vertical solid lines (red in the coloured version) are the change-points. Dotted and dashed vertical lines (blue in coloured version) indicate the position of the start of the validation and test set.

the length of simulated data here is much shorter (around $192-500$ vs around $512-2000$ in Baranowski et al., 2019), and the noise used is $ARMA(1,1)$ with $\phi=0.4$ and $\theta=0.2$ instead of the i.i.d or $AR(1)$ noise considered in Baranowski et al. (2019). The modification aims to make our simulated data to be more similar to the time series observed in the real world, than the one from Baranowski et al. (2019). Also, note that there is no change-point in the noise structure for the Model 1-5 considered here. Figure 3.3 shows one realisation from each model. Here the simulation settings include the underlying signal with piecewise constant mean, piecewise linear continuous mean and piecewise linear mean (not necessarily continuous) with one or more change-points.

- Model 1 (`teeth`): piecewise constant signal (S1) with 2 jumps at $\tau=64,128$ with sizes $-2,2$. Initial mean 1 and $T=192$. This is similar to the M1 in Baranowski et al. (2019) except the time series is shorter.

- Model 2 (`block`): piecewise constant signal (S1) with 3 jumps at $\tau=205,267,308$ with sizes $1.464,-1.830,1.098$. Initial mean 0 and $T=350$. This is similar to the M2 in Baranowski et al. (2019) except the time series is shorter.

- Model 3 (`wave1`): piecewise linear continuous signal (S2), with 3 change-points at $\tau=91,182,273$ with changes in slopes $-3\times2^{-6}, 4\times2^{-6}, -5\times2^{-6}$. Starting intercept 1, starting slope 2^{-8} and $T=350$. This has similar wave signal as the M3 in Baranowski et al. (2019).

- Model 4 (`wave2`): piecewise linear signal without jumps (S2), with 1 change-point at $\tau=100$ with change in slope -2^{-5}. Starting intercept 2^{-1}, starting slope $=2^{-6}$ and $T=200$. This is similar to the M4 in Baranowski et al. (2019) except the time series is shorter and each interval between the change-points are shorter.

- Model 5 (`mix`): piecewise linear with possible jumps at change-points (S3) with 4 change-points at $\tau=100,200,300,400$, jump sizes $0,-1,2,-1$ and changes in the

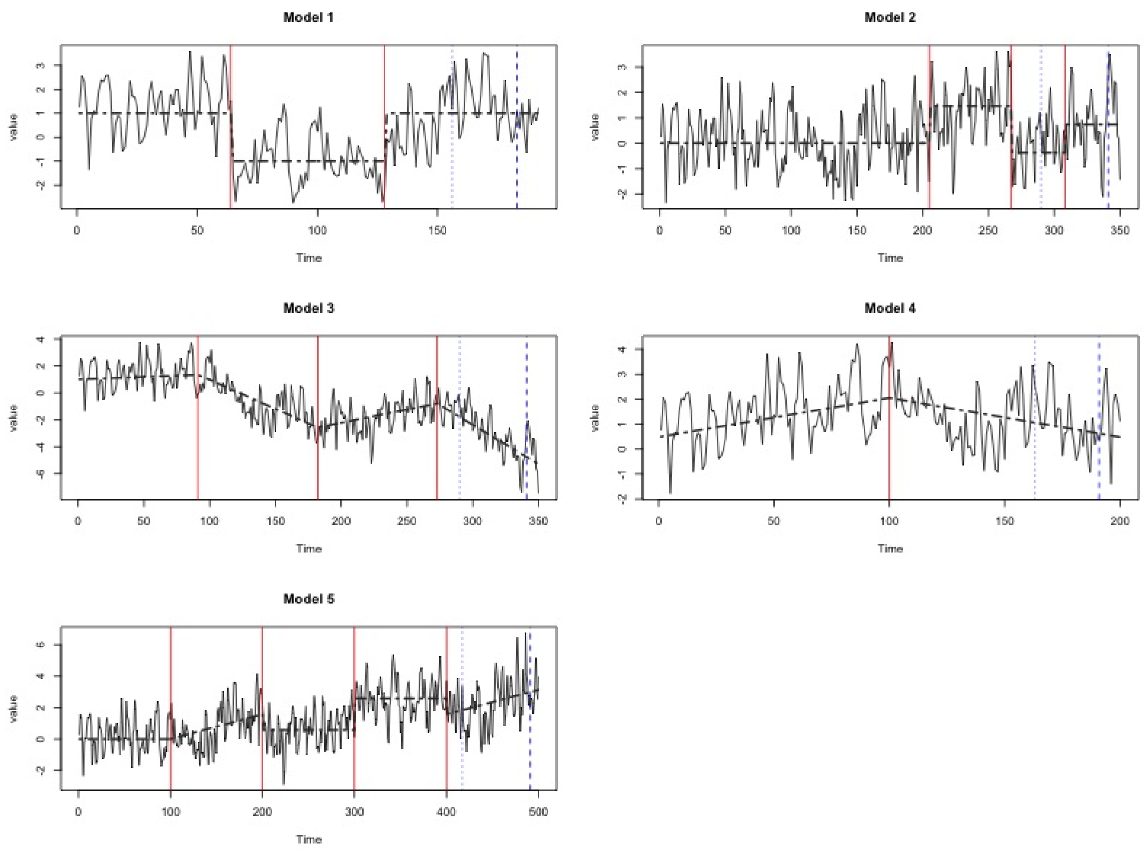

Fig. 3.3 One time series realisation from Model 1-5. Solid lines (grey in the coloured version) are the simulated time series and the dotted line (black in the coloured version) are the signals. Vertical solid lines (red in the coloured version) are the change-points. Dotted and dashed vertical lines (blue in coloured version) indicate the position of the start of the validation and test set.

slope $2-6, -2^{-6}, 0, 2^{-6}$. Starting intercept 0, starting slope 0 and $T = 500$. This has similar signal as the M5 in Baranowski et al. (2019).

Settings with exaggerated signal change

We consider the same settings as Model 1-5 but the change in signal (in terms of both change in slope and jump) is 5 times the one in Model 1-5, and we call the exaggerated settings as Model 1e-5e. Figure 3.4 shows one realisation from Model 1e-5e. Note that in these settings, the location of the change-points can be detected easily by human eyes. Here we want to see whether which prediction method performs the best depends on the jump size.

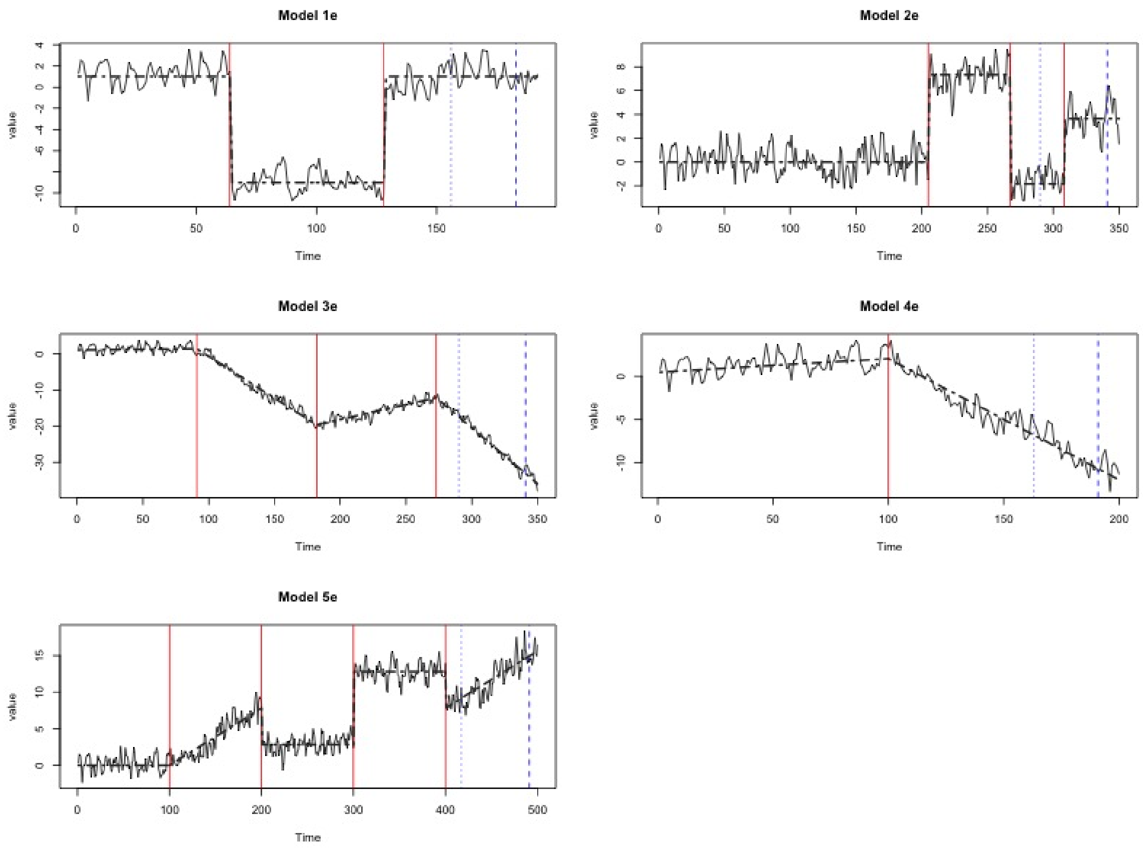

Fig. 3.4 One time series realisation from Model 1e-5e. Solid lines (grey in the coloured version) are the simulated time series and the dotted line (black in the coloured version) are the signals. Vertical solid lines (red in the coloured version) are the change-points. Dotted and dashed vertical lines (blue in coloured version) indicate the position of the start of the validation and test set.

Settings with change in the signal and the noise structures

We consider the same settings as Model 1-5 but the noise structure can be changed as well. In the model specification below, the ARMA structure is represented in the form of (x, y), with the first number in the parenthesis is the AR parameter and the second number is the MA parameter. Figure 3.5 shows one realisation from Model 1a-5a. We want to see whether having the change in noise structures affects the performance of different time series methods.

- Model 1a (`teeth`): same as Model 1 except the ARMA in each segment is: $(0.8, 0.2), (0.1, 0.3), (0.4, 0.2)$.
- Model 2a (`block`): same as Model 2 except the ARMA in each segment is: $(0.1, 0.3), (0.8, 0.2), (0.1, 0.3), (0.4, 0.2)$.
- Model 3a (`wave1`): same as Model 3 except the ARMA in each segment is: $(0.1, 0.3), (0.8, 0.2), (0.1, 0.3), (0.4, 0.2)$.
- Model 4a (`wave2`): same as Model 4 except the ARMA in each segment is: $(0.1, 0.3), (0.4, 0.2)$.
- Model 5a (`mix`): same as Model 5 except the ARMA in each segment is: $(0.4, 0.2), (0.1, 0.3), (0.8, 0.2), (0.1, 0.3), (0.4, 0.2)$.

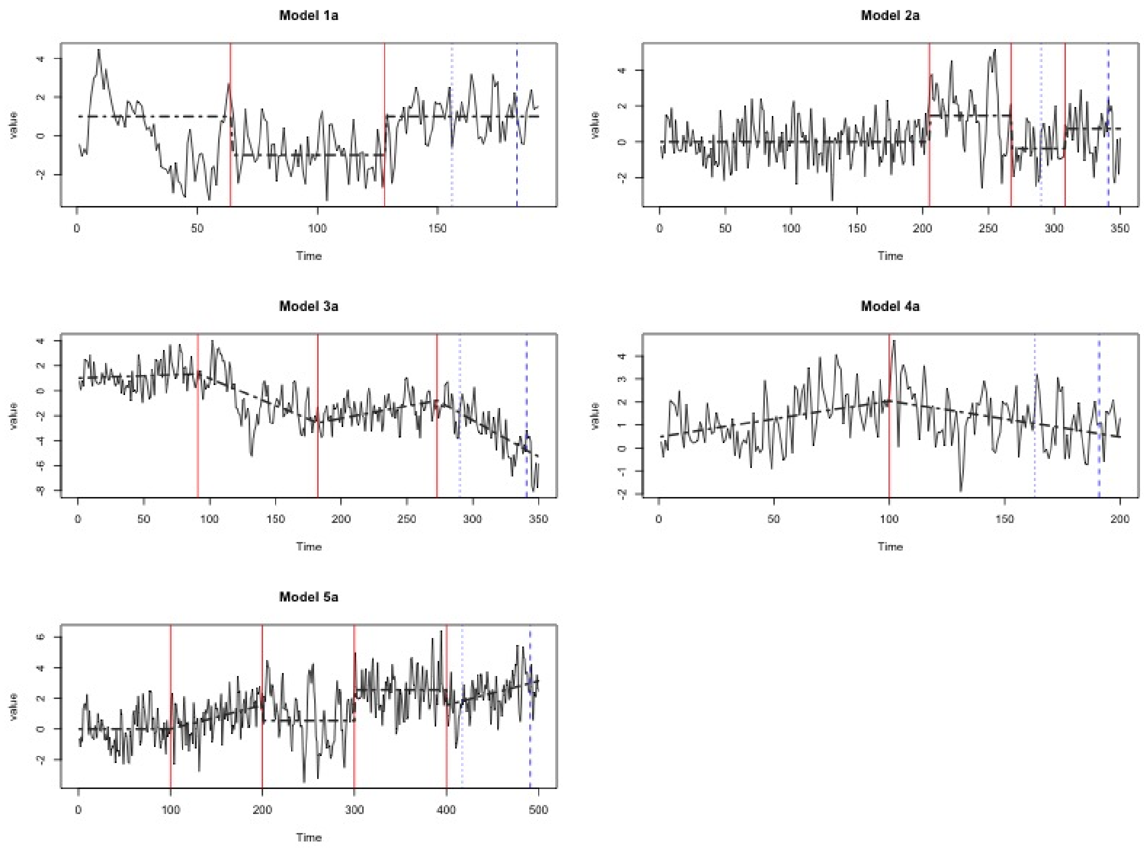

Fig. 3.5 One time series realisation from Model 1a-5a, where the noise signals can be changed. Solid lines (grey in the coloured version) are the simulated time series and the dotted line (black in the coloured version) are the signals. Vertical solid lines (red in the coloured version) are the change-points. Dotted and dashed vertical lines (blue in coloured version) indicate the position of the start of the validation and test set.

Simulation results

The prediction performance of NOT-ARMA-1 and NOT-ARMA-2 with different ways to select the threshold on the solution path (NOT-CV-min, NOT-CV-1SE and BIC) as well as the original NOT, the oracle and the robust methods on the simulated data is shown in Table 3.1 to Table 3.4 (corresponding to different sets of simulation settings). Table 3.5 to Table 3.8 (corresponding to different sets of simulation settings) show the number of change-points selected by each method except the robust methods. Below we discuss the performance of the proposed methods and the robust methods in details.

First, notice that oracle is not always able to outperform the baseline method. For Model 2s when there is only change in noise, the oracle which uses only the post-break data performs worse than the baseline method which uses all the available data. For the robust methods, the average across estimation windows method provides very good prediction performance. In most of the settings it outperforms the baseline. This is consistent with the findings in literature that we have reviewed. The other robust methods also perform quite well, although their performance is not as good as the average window method.

Overall, the newly proposed methods with BIC for thresholding (i.e. NOT-ARMA-1(BIC) and NOT-ARMA-2(BIC)) provide the best performance among all the methods proposed in this chapter. For settings where the change is relatively large like Model 1s-7s and 1e-5e, NOT-ARMA-1(BIC) and NOT-ARMA-2(BIC) have the relative MSFE less than 1 (or equal to 1 for Model 1s and 2s), indicating that they outperform the baseline method in these settings. In these settings they are also likely to perform better than the robust methods. For Model 1-5, NOT-ARMA-1(BIC) seldom selects any change-points (Table 3.6) so its performance is very similar to the baseline method. For Model 1a-5a, for some settings NOT-ARMA-2(BIC) performs better than the baseline in some settings and worse in other settings with a relatively small margin, but the robust averaging method is likely to perform better than NOT-ARMA-1(BIC) and NOT-ARMA-2(BIC). When comparing with the original NOT, the NOT-ARMA-1(BIC) and NOT-ARMA-2(BIC) has a smaller relative MSFE than the original NOT in all settings, showing that the proposed method improves the prediction performance of the original NOT on dependent data under the simulation settings we considered. NOT-ARMA-1(BIC) and NOT-ARMA-2(BIC) often underestimate the number of change-points, except for the settings where the change is large. In those cases they often select the right number of change-points.

For the original NOT, it usually performs worse than the baseline except for the Model 3s-7s where the change is large. Also, it often has a relative MSFE higher than the NOT-ARMA-2 with different thresholding methods. For all model settings, it overestimates the number of change-points.

For NOT-ARMA-1 with thresholds selected by NOT-CV-min or NOT-CV-1SE, it selects too many change-points and performs poorly in most cases, even when the jump size is large. For NOT-ARMA-1(NOT-CV-min), in quite a number of settings it performs even worse than the original NOT, and in no case NOT-ARMA-1(NOT-CV-min) outperforms the baseline. While NOT-ARMA-1(NOT-CV-1SE) performs better than NOT-ARMA-1(NOT-CV-min), it seldom outperforms the baseline.

For NOT-ARMA-2 with thresholds selected by NOT-CV-min and NOT-CV-1SE, for the Model 1e to 5e and 3s to 6s where the change is large, it often outperforms the baseline. For other settings, however, NOT-ARMA-2 is less likely to outperform the baseline method.

Model	Oracle	NOT	NOT-ARMA-1			NOT-ARMA-2			Average	Rolling	CV fixed pt	CV window
			(CV-min)	(CV-1SE)	(BIC)	(CV-min)	(CV-1SE)	(BIC)				
1s	1.00	1.17	1.52	1.20	1.00	1.17	1.05	1.00	1.02	1.09	1.04	1.11
2s	1.03	1.21	1.43	1.09	1.00	1.13	1.01	1.00	1.00	1.07	1.03	1.08
3s	0.76	0.85	1.07	1.06	0.76	0.86	0.85	0.73	0.91	0.84	0.80	0.84
4s	0.73	0.86	1.12	0.90	0.73	0.81	0.80	0.72	0.88	0.76	0.75	0.82
5s	0.75	0.91	1.37	0.93	0.75	0.90	0.87	0.75	0.77	0.82	0.79	0.80
6s	0.88	1.06	1.21	1.05	0.99	0.96	0.97	0.97	0.91	0.91	0.92	0.96
7s	0.77	0.94	1.14	0.99	0.85	1.10	1.01	0.97	0.84	0.80	0.80	0.84

Table 3.1 Average relative MSFE for the simple settings with Model 1s-7s. Results are normalised with the baseline method so that the baseline method always have the relative MSFE equals to 1.

Model	Oracle	NOT	NOT-ARMA-1			NOT-ARMA-2			Average	Rolling	CV fixed pt	CV window
			(CV-min)	(CV-1SE)	(BIC)	(CV-min)	(CV-1SE)	(BIC)				
1	0.98	1.43	1.37	1.13	1.00	1.08	1.00	0.96	0.98	1.02	1.01	1.04
2	1.07	1.64	1.42	1.15	1.00	1.13	1.03	1.05	1.00	1.07	1.02	1.03
3	0.84	1.33	1.47	1.18	1.00	0.97	1.01	0.95	0.91	0.87	0.90	0.95
4	0.91	1.45	1.61	1.20	1.00	1.06	0.99	1.00	0.93	1.08	0.94	0.95
5	1.04	1.71	1.39	1.09	1.00	1.11	1.02	1.04	1.01	1.05	1.03	1.02

Table 3.2 Average relative MSFE for the simulation settings with Model 1-5. Results are normalised with the baseline method so that the baseline method always have the relative MSFE equals to 1.

Model	Oracle	NOT	NOT-ARMA-1			NOT-ARMA-2			Average	Rolling	CV fixed pt	CV window
			(CV-min)	(CV-1SE)	(BIC)	(CV-min)	(CV-1SE)	(BIC)				
1e	0.85	1.22	1.24	1.01	0.85	0.98	0.94	0.81	0.91	0.89	0.90	0.92
2e	0.93	1.34	1.22	1.08	0.93	1.00	0.99	0.87	0.97	1.04	1.01	1.02
3e	0.78	1.14	1.39	1.12	1.00	0.99	1.00	0.97	0.89	0.81	0.79	0.88
4e	0.78	1.16	1.63	1.12	0.78	0.82	0.93	0.78	0.83	0.92	0.80	0.81
5e	0.87	1.23	1.33	1.09	0.87	0.95	1.01	0.85	0.96	0.88	0.90	0.98

Table 3.3 Average relative MSFE for the exaggerate settings with Model 1e-5e. Results are normalised with the baseline method so that the baseline method always have the relative MSFE equals to 1.

3.5.4 Real data analysis

We analyse the performance of the methods further using real economics data.

Model	Oracle	NOT	NOT-ARMA-1			NOT-ARMA-2			Average	Rolling	CV fixed pt	CV window
			(CV-min)	(CV-1SE)	(BIC)	(CV-min)	(CV-1SE)	(BIC)				
1a	0.94	1.46	1.22	1.07	1.01	0.99	0.99	0.96	0.97	0.99	0.98	1.05
2a	1.01	1.57	1.31	1.09	1.04	1.11	1.01	1.05	0.98	1.04	1.02	1.03
3a	0.86	1.44	1.50	1.17	1.00	0.94	1.01	0.98	0.92	0.88	0.92	0.94
4a	0.87	1.53	1.58	1.27	1.00	1.07	1.01	1.01	0.88	0.97	0.90	0.90
5a	1.02	1.85	1.47	1.16	1.02	1.15	1.01	1.02	1.00	1.04	1.04	1.06

Table 3.4 Average relative MSFE for the settings with Model 1a-5a, where the ARMA structure can be changed. Results are normalised with the baseline method so that the baseline method always have the relative MSFE equals to 1.

Model	Oracle	NOT	NOT-ARMA-1			NOT-ARMA-2		
			(CV-min)	(CV-1SE)	(BIC)	(CV-min)	(CV-1SE)	(BIC)
1s	0.00	0.70	10.20	3.59	0.00	6.72	1.07	0.02
2s	1.00	3.01	9.88	2.97	0.01	7.68	0.92	0.02
3s	1.00	1.89	12.56	4.63	1.00	8.07	1.65	1.02
4s	1.00	4.25	11.34	4.20	1.00	8.43	1.46	1.01
5s	1.00	2.26	9.07	2.84	1.00	7.23	1.24	1.04
6s	1.00	3.91	10.44	3.53	0.09	7.47	1.37	0.43
7s	1.00	1.16	9.40	3.32	0.68	6.02	1.81	1.07

Table 3.5 Average number of change-points selected for the simple settings with Model 1s-7s.

Model	Oracle	NOT	NOT-ARMA-1			NOT-ARMA-2		
			(CV-min)	(CV-1SE)	(BIC)	(CV-min)	(CV-1SE)	(BIC)
1	2.00	5.39	9.22	3.19	0.13	5.75	1.05	1.71
2	3.00	6.66	17.16	6.27	0.00	6.18	0.69	1.07
3	3.00	3.39	12.16	4.38	0.00	0.90	0.17	1.15
4	1.00	1.73	7.37	2.88	0.00	3.09	0.64	0.34
5	4.00	3.83	13.77	4.57	0.02	3.77	0.50	0.28

Table 3.6 Average number of change-points selected for the simulation settings with Model 1-5.

Real data

We gather the monthly UK economics data from Eurostat via DBnomics. Below we give the details on how we select the datasets. Eurostat has many different databases and we reduce the scope with the following criteria:

Model	Oracle	NOT	NOT-ARMA-1			NOT-ARMA-2		
			(CV-min)	(CV-1SE)	(BIC)	(CV-min)	(CV-1SE)	(BIC)
1e	2.00	5.49	10.37	3.87	2.00	7.53	1.77	2.06
2e	3.00	6.64	18.84	7.31	2.98	9.20	1.84	3.02
3e	3.00	3.03	9.96	4.48	0.35	0.27	0.04	0.70
4e	1.00	1.67	6.63	2.83	0.98	0.81	0.42	1.00
5e	4.00	4.66	16.55	5.66	3.54	3.63	0.77	4.01

Table 3.7 Average number of change-points selected for the exaggerate settings with Model 1e-5e.

Model	Oracle	NOT	NOT-ARMA-1			NOT-ARMA-2		
			(CV-min)	(CV-1SE)	(BIC)	(CV-min)	(CV-1SE)	(BIC)
1a	2.00	7.08	9.42	3.05	0.40	5.72	1.24	1.44
2a	3.00	6.25	15.88	5.86	0.60	6.69	0.89	1.75
3a	3.00	6.96	12.99	5.36	0.01	1.36	0.21	0.51
4a	1.00	1.34	7.07	2.99	0.04	2.79	0.73	0.41
5a	4.00	6.42	15.84	6.08	0.17	3.44	0.35	0.51

Table 3.8 Average number of change-points selected for the settings with Model 1a-5a, where the ARMA structure can be changed.

- Use only datasets that have the name "monthly". This makes sure that data series in the datasets have only monthly data. We use monthly data so that the data series are long enough, and change-points are less likely to occur during the forecasting window.
- For each dataset, only select the first data series satisfying some criteria (next bullet point). Only one data series is used in each dataset because the data series in each dataset can be very similar. This reduces the correlation of the prediction performance among the real time series considered.
- Each data series has to satisfy the following:
  - $length > 200$.
  - standard deviation for the 101^{th} to 200^{th} has to be greater than 0.01. This prevents the data to be constant for a prolonged period of time.

– if the dataset contains both seasonal adjusted and non-seasonal adjusted data series, choose the one that is seasonal adjusted.

We select the data with these rather rigid rules to avoid cherry-picking and attempt to get real-life time series with different behaviours. Using the above criteria, we have 46 data series. We further remove the data series that are dominated by the seasonal effects or spikes. We are left with 41 time series. The full list of real data used is in Appendix Table B.1 and the corresponding plots are in Figure 3.6-3.12.

The sets of time series used here include price indexes / inflation, production (e.g meat), exchange rate, interest rates, unemployment and sales and trade, etc. Prediction of these time series is important for government policy and company planning. For example, forecasting of the unemployment rate impacts the government fiscal policy. For some of the time series like inflation, we have pointed out in Chapter 2 that change-point models are considered to model them. Some other time series may be predicted with other predictors and with other models based on what we know about the time series. For example, Askitas and Zimmermann (2009) show there is a strong correlation between Google keyword searches and the unemployment rates, and Fondeur and Karamé (2013) show that including Google search data improves the accuracy of the unemployment prediction. Nonetheless, pure time series models may still be used as a baseline (e.g. D'Amuri and Marcucci 2010).

Some of the real time series we use in the empirical study appear to undergo some structure changes. For example, time series 11 (Figure 3.7 bottom left) and 29 (Figure 3.10 bottom left) appear to have a change in variability in around 2008. Time series 1 (Figure 3.6 top left) and 8 (Figure 3.7 top right) appear to have change in slope and possibly change in variability as well. Some time series like time series 17 (Figure 3.8 bottom left) and 20 (Figure 3.9 top right) appear to be dominated by the linear trend. Some time series are clearly related in the data sets. For example, there are a few time series that are related to HICP.

Nevertheless, they still show different interesting patterns and we do not filter out these related time series.

The approach of using a variety of UK economics data to study the prediction performance of different methods on time series with potential change-points is similar to the empirical study in Eklund et al. (2010). Eklund et al. (2010) use many different UK economics time series like unemployment, manufacturing, GDP, etc in their empirical study. Quarterly data is used in their analysis but we use monthly data here. Eklund et al. (2010) use the economics data to compare the performance of the methods that monitor changes to robust methods that do not monitor changes.

Real data analysis results

The prediction performance of NOT-ARMA-1 and NOT-ARMA-2 with different ways to select the thresholds on the solution path (NOT-CV-min, NOT-CV-1SE and BIC) as well as the original NOT and the robust methods on the real data is shown in Table 3.9.

The relative performance of the methods on the real datasets is similar to the results on the simulated datasets. NOT-ARMA-1(BIC) provides the best performance on the real datasets. Note that it often has the relative MSFE as 1, except for a few datasets like dataset 1, 8, 16, etc. This means NOT-ARMA-1(BIC) often has the same performance as the no-break model. Such result is not surprising as NOT-ARMA-1(BIC) often selects no change-points (see Table 3.10). When NOT-ARMA-1(BIC) does select the change-point, it performs better than the baseline no-break model. In fact, it is the only method that consistently outperforms the baseline method.

For NOT-ARMA-2 with three different ways to select the thresholds, while for some datasets they outperform the baseline method, for other datasets they perform worse than the baseline method. The NOT-ARMA-2 is not able to provide better performance than the no-break method for the real datasets we considered. For the original NOT and the NOT-ARMA-1 with thresholds selected by NOT-CV-min or NOT-CV-1SE, they often perform

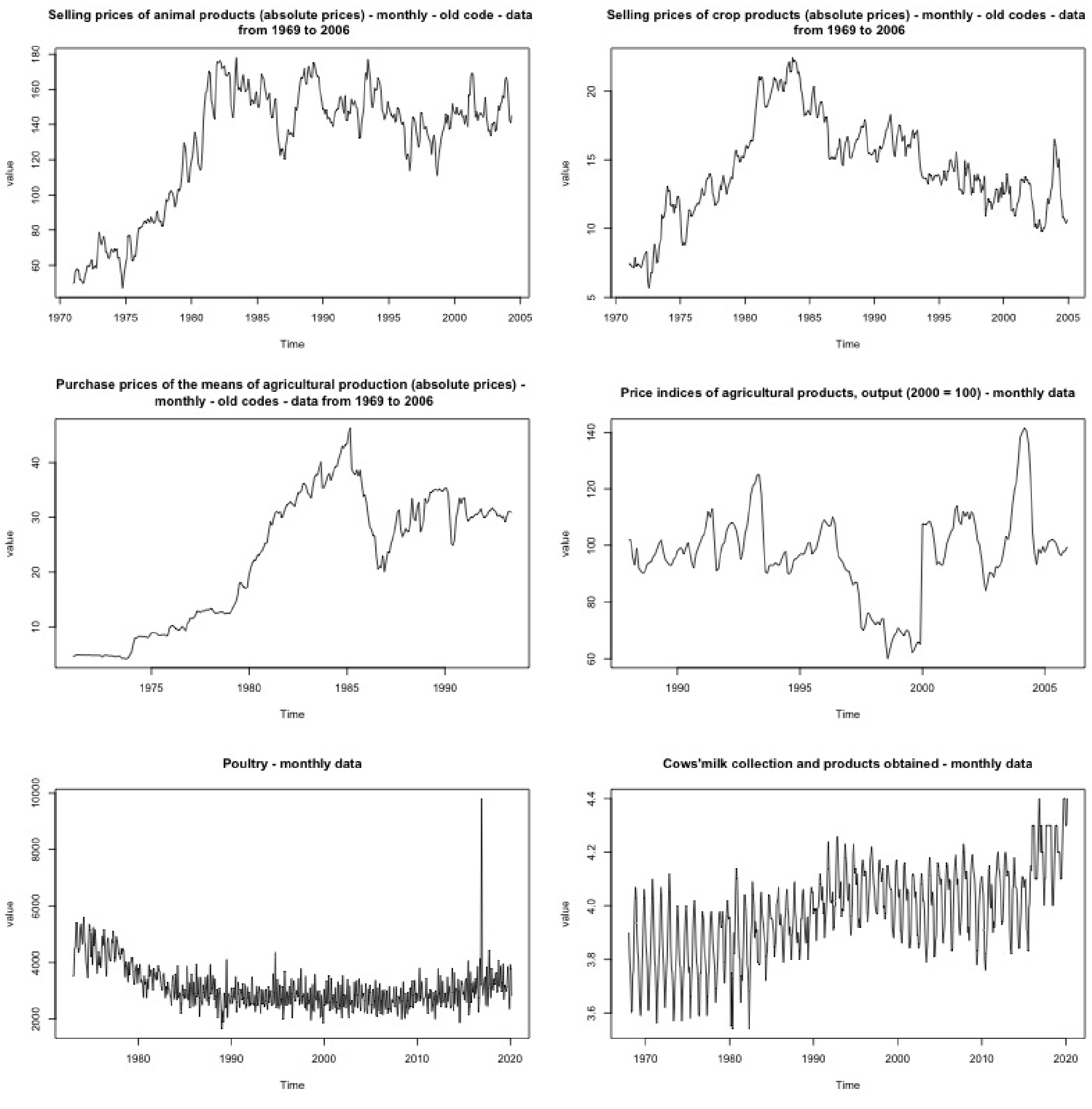

Fig. 3.6 Real time series used in empirical study (time 1-6).

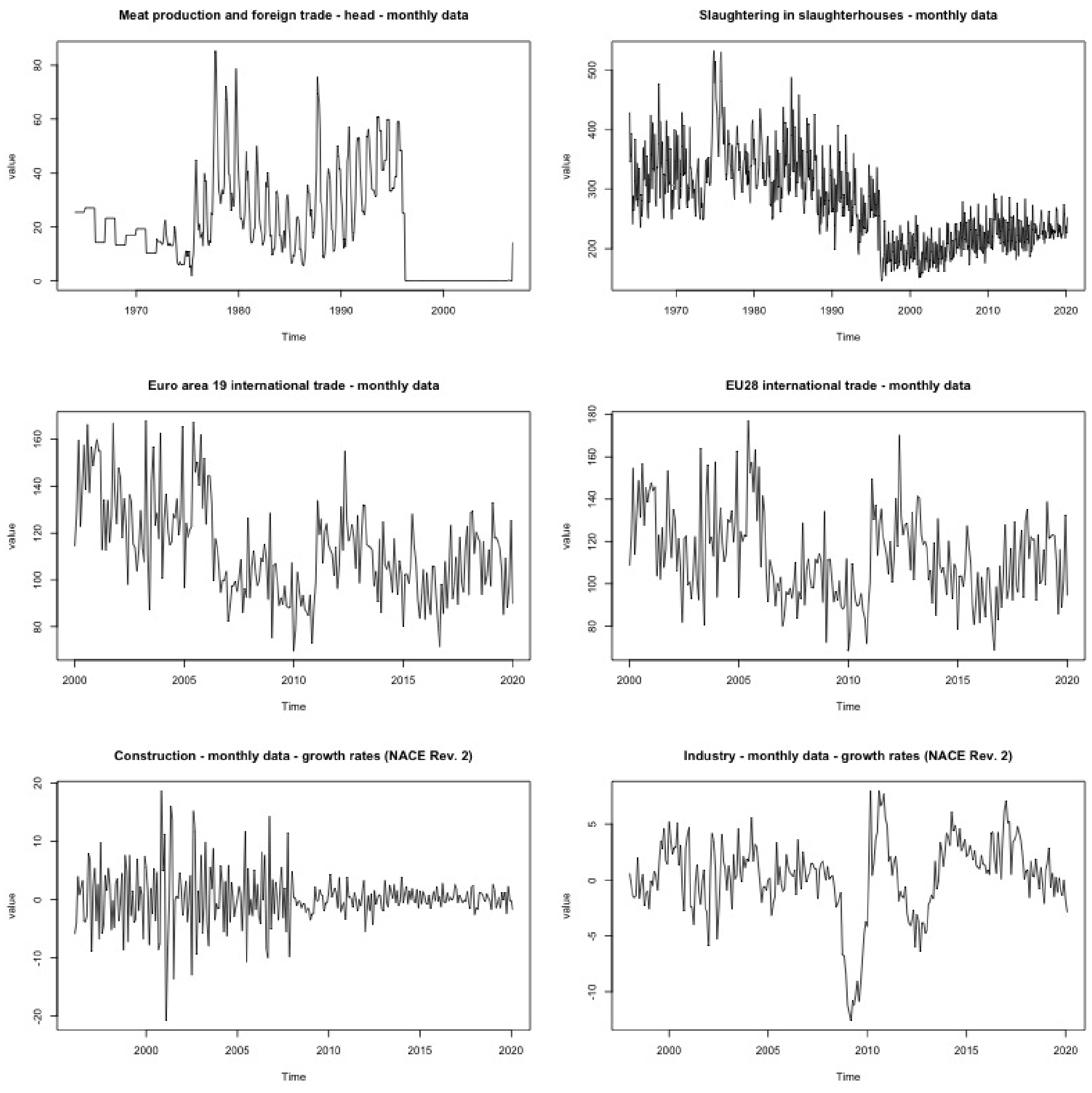

Fig. 3.7 Real time series used in empirical study (setting 7-12).

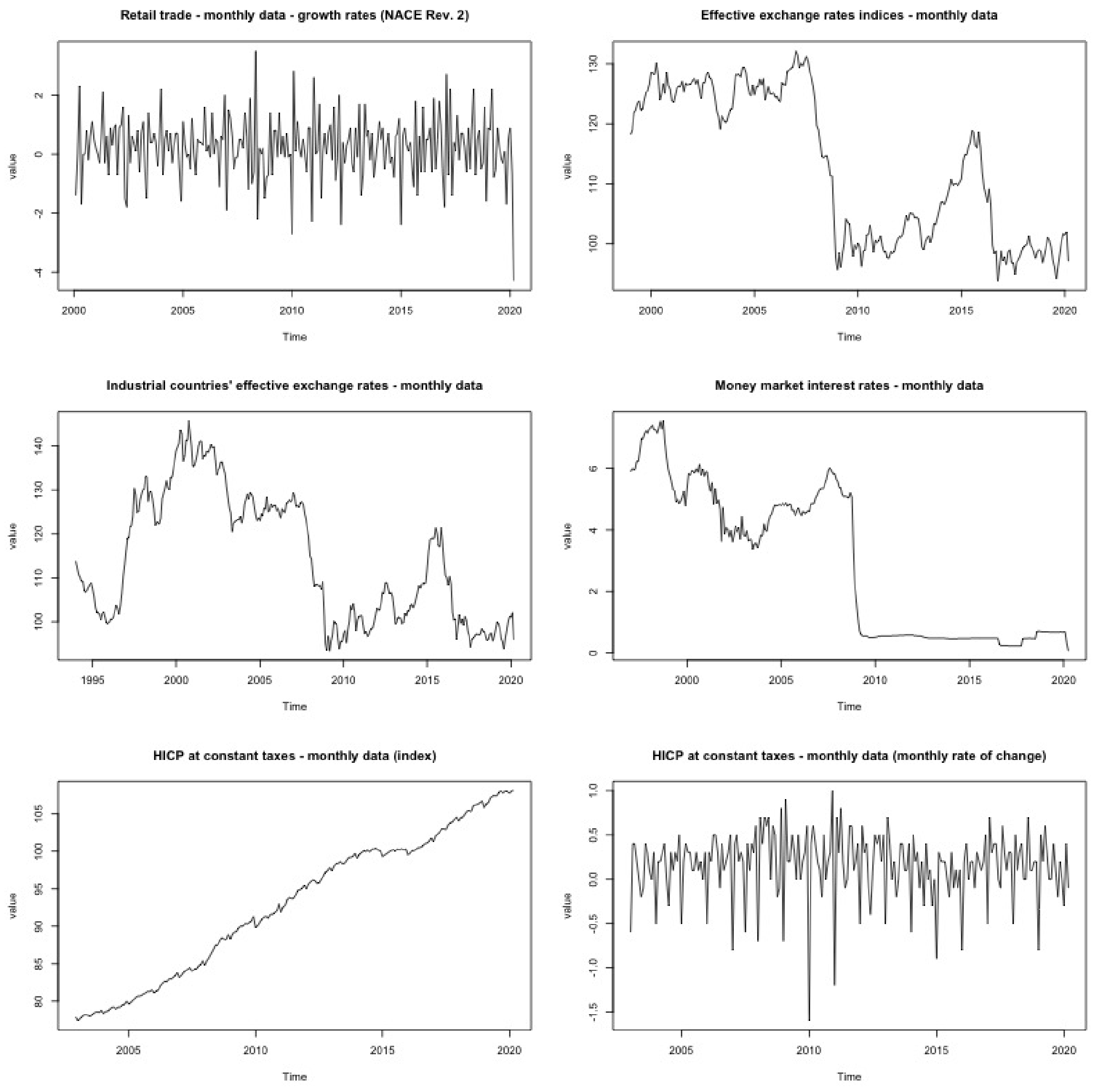

Fig. 3.8 Real time series used in empirical study (setting 13-18).

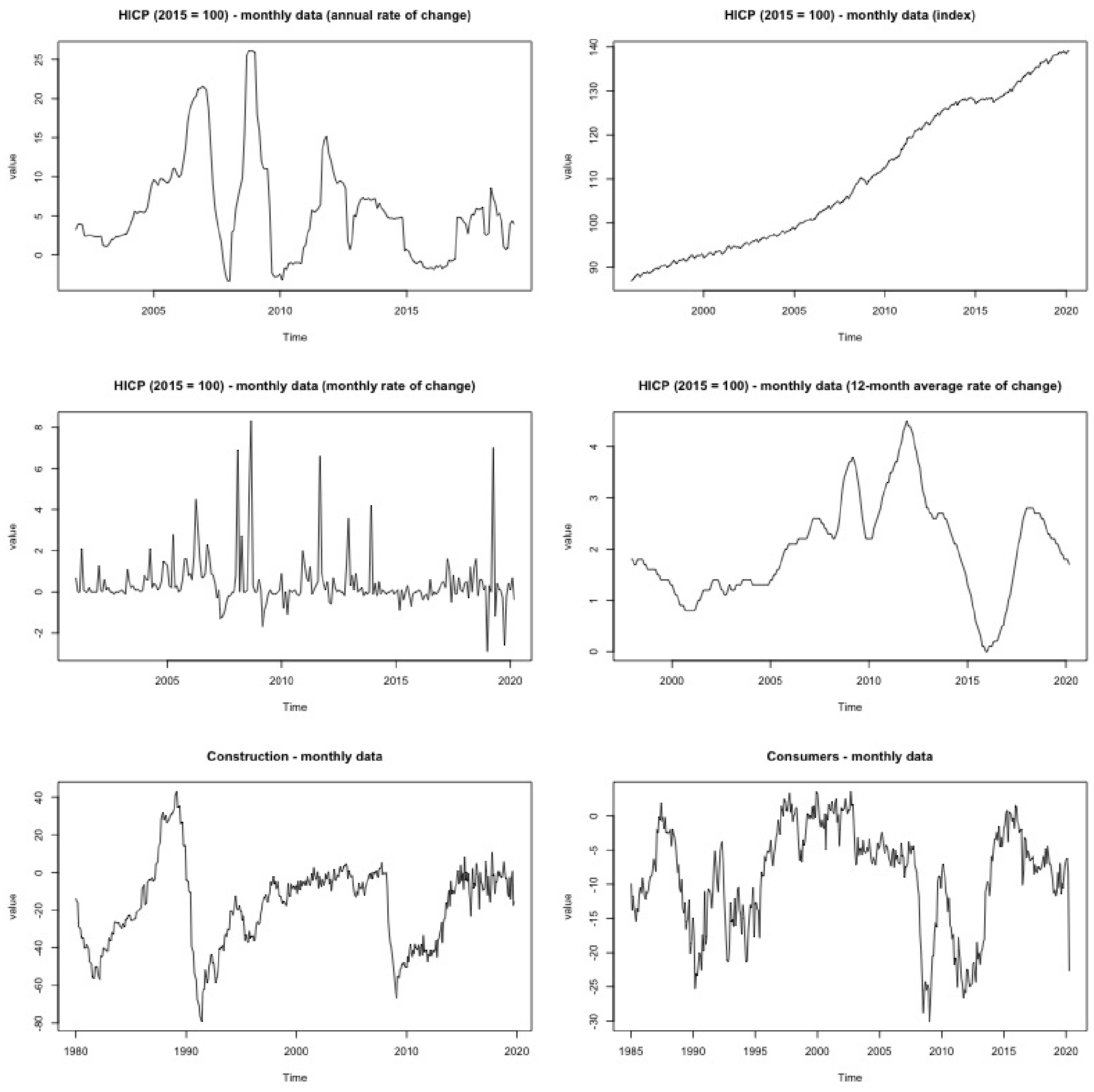

Fig. 3.9 Real time series used in empirical study (setting 19-24).

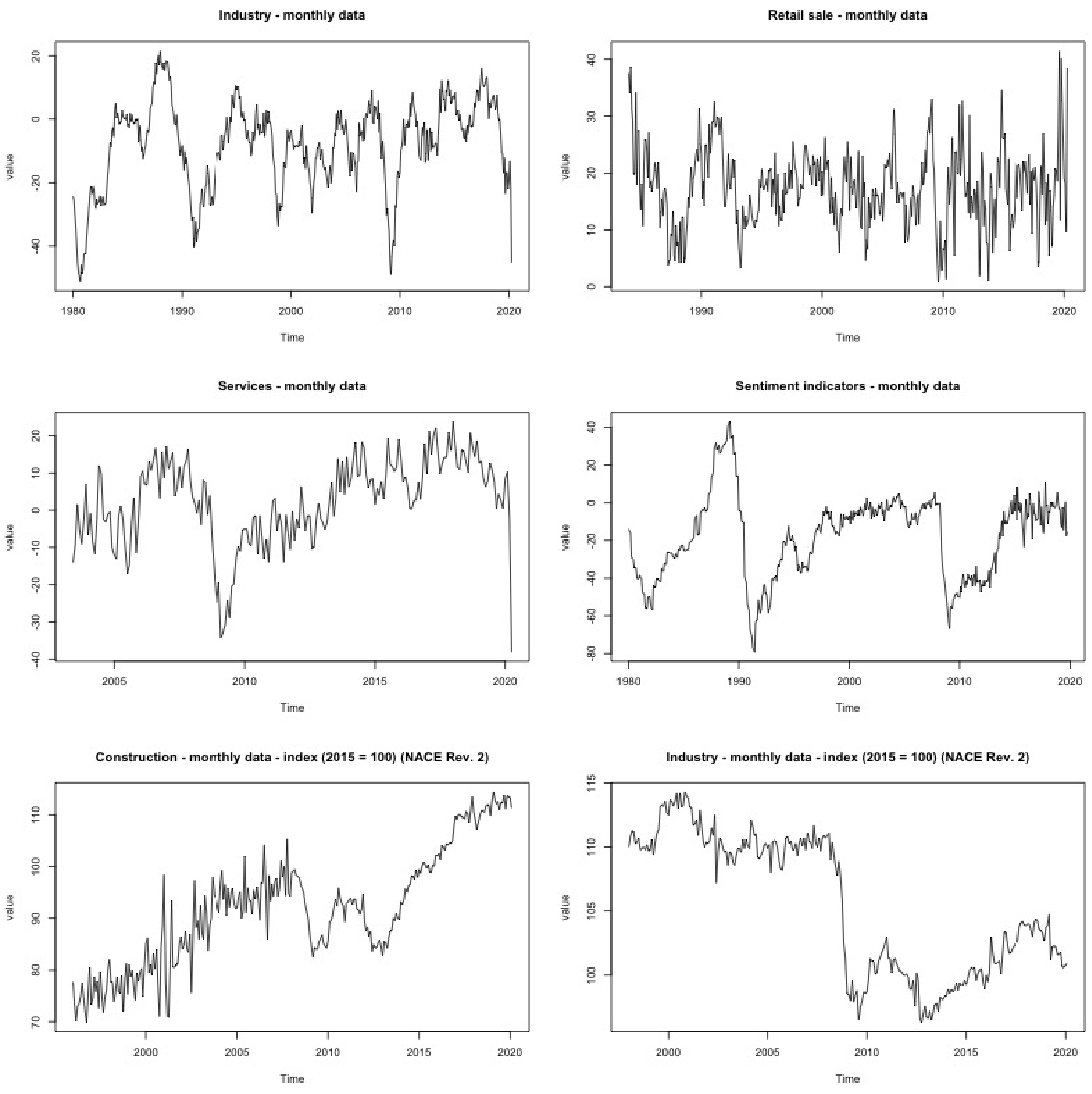

Fig. 3.10 Real time series used in empirical study (setting 25-30).

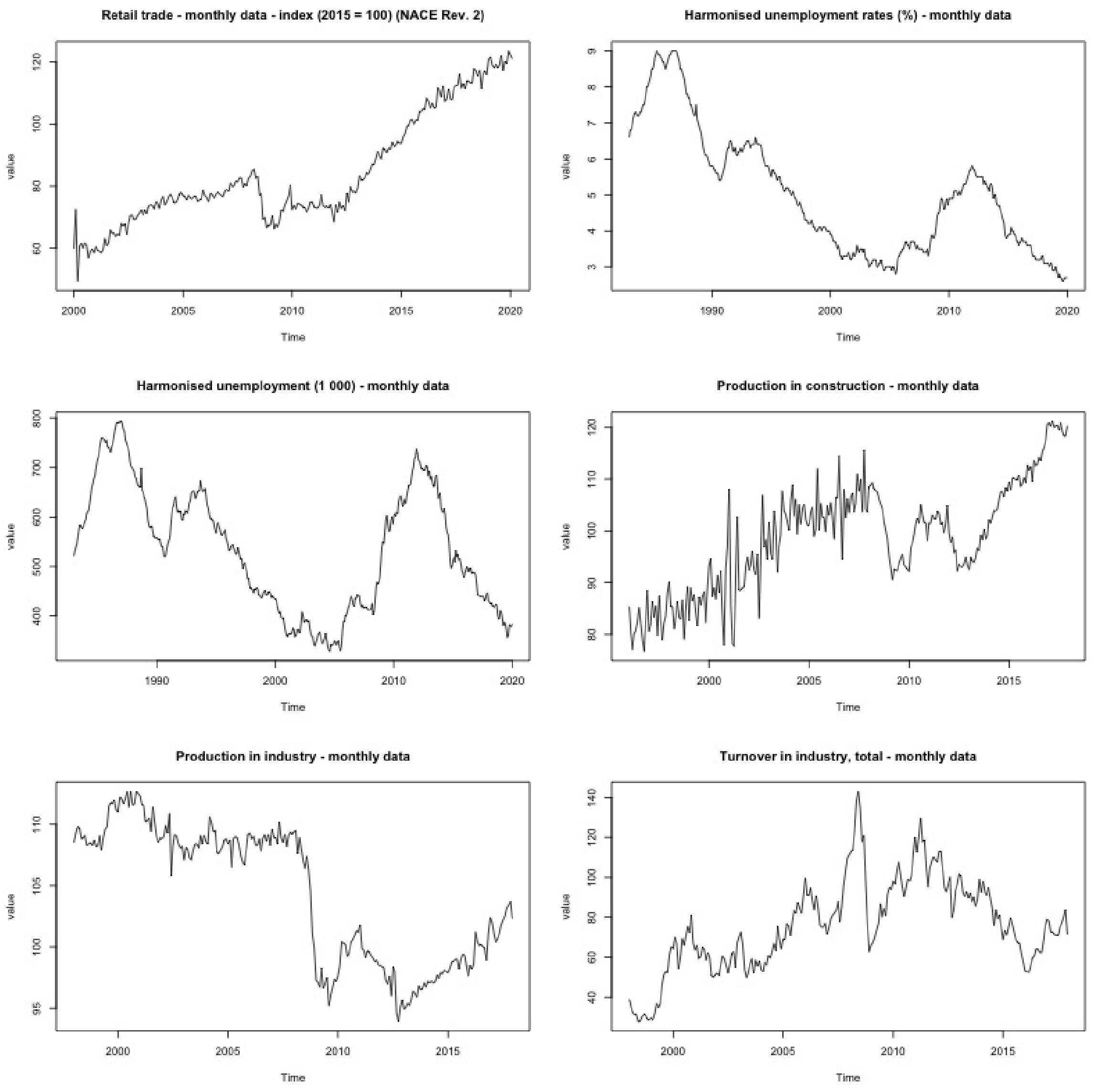

Fig. 3.11 Real time series used in empirical study (setting 31-36).

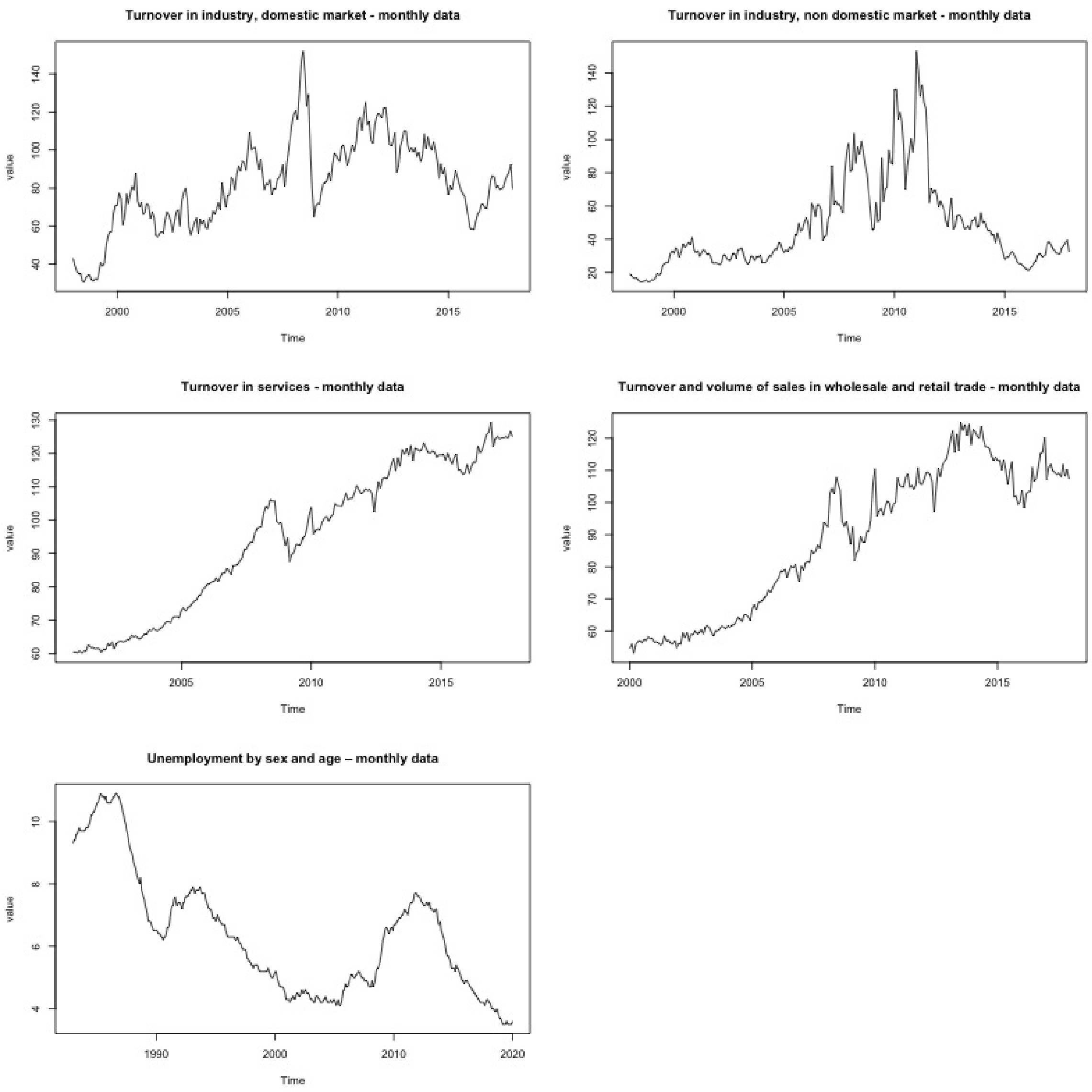

Fig. 3.12 Real time series used in empirical study (setting 37-41).

worse than the baseline. For robust methods, for some datasets they outperform the baseline method, for other datasets they perform worse than the baseline method. For some datasets that do not have an obvious change-point like datasets 17 and 20, the averaging window method has a high relative MSFE, indicating that it performs worse than the no change-point baseline model. NOT-ARMA-1(BIC) and NOT-ARMA-2(BIC), on the other hand, have the same performance as the baseline model, as they select no change-points.

real datasets	NOT	NOT-ARMA-1			NOT-ARMA-2			Average	Rolling	CV fixed pt	CV window
		(CV-min)	(CV-1SE)	(BIC)	(CV-min)	(CV-1SE)	(BIC)				
1	7.62	2.92	1.18	0.98	0.95	1.00	0.93	0.97	0.97	0.97	1.01
2	8.31	2.77	2.05	1.00	0.66	1.00	0.70	0.86	0.92	0.86	0.87
3	2.61	1.02	1.00	1.00	1.16	1.00	1.00	0.89	0.99	0.85	0.85
4	5.31	10.97	7.90	1.00	2.40	1.75	1.00	1.10	1.24	1.09	1.04
5	1.10	2.78	2.38	1.00	0.99	0.99	0.97	0.91	1.73	1.11	1.21
6	2.10	0.70	1.02	1.00	0.97	1.05	1.00	0.93	0.70	0.99	1.05
7	1.14	1.09	1.09	1.00	1.09	1.09	1.07	1.09	1.09	1.09	1.09
8	0.34	0.25	0.45	0.28	0.65	0.81	0.36	0.48	0.31	0.33	0.26
9	1.91	1.74	1.13	1.00	1.16	1.00	1.00	1.08	1.81	1.08	1.13
10	1.42	1.71	1.00	1.00	1.27	0.99	1.00	1.05	1.50	1.05	1.04
11	1.52	1.22	1.03	1.00	1.68	1.75	1.00	1.13	1.58	1.24	1.18
12	1.44	1.11	1.03	1.00	1.64	1.30	1.13	0.98	0.82	0.99	1.18
13	0.85	1.37	1.17	1.00	0.98	1.00	1.00	1.05	1.23	1.09	0.85
14	1.90	1.02	1.25	1.00	1.02	1.00	1.00	1.01	0.98	1.00	1.04
15	1.65	1.01	1.00	1.00	0.99	1.00	1.00	0.99	0.98	0.97	0.98
16	3.09	1.04	1.11	0.72	1.00	1.00	1.00	0.98	1.41	1.07	1.10
17	0.87	1.98	1.77	1.00	0.70	1.00	1.00	2.12	2.01	1.02	0.98
18	1.12	5.28	1.71	1.00	1.29	1.17	1.00	0.98	1.15	0.97	0.89
19	8.47	3.29	1.94	1.00	1.07	1.05	1.01	1.01	1.83	1.44	0.97
20	1.65	2.06	0.88	1.00	0.90	0.94	1.00	3.40	2.26	1.18	1.09
21	1.07	19.79	13.98	1.00	1.09	1.01	1.00	1.19	1.73	1.73	1.62
22	0.73	0.62	0.69	1.00	1.25	0.92	1.55	0.92	1.79	1.05	1.06
23	0.96	1.91	1.60	1.00	1.00	1.00	1.00	0.91	1.07	1.06	0.86
24	0.91	0.85	0.99	1.00	0.99	1.00	1.00	1.00	0.90	1.00	1.01
25	1.11	0.74	0.76	1.00	1.01	1.00	1.00	1.01	0.99	0.99	1.03
26	1.85	1.92	1.15	1.00	0.99	1.00	1.00	0.99	1.04	0.98	0.99
27	1.12	1.51	1.47	1.00	0.95	1.05	1.00	1.03	0.91	0.99	0.88
28	0.96	1.91	1.60	1.00	1.00	1.00	1.00	0.91	1.07	1.06	0.86
29	12.05	0.86	0.95	1.00	1.00	1.00	1.00	1.30	2.24	1.69	2.07
30	8.83	7.38	1.58	1.00	1.07	1.00	1.00	1.36	3.93	3.86	3.80
31	1.11	1.08	1.02	1.00	1.06	1.00	1.00	0.99	0.92	0.87	1.05
32	1.51	0.86	0.80	1.00	1.00	1.00	1.00	0.94	1.05	0.98	1.16
33	1.70	0.69	0.85	1.00	1.00	1.00	1.00	0.98	1.01	1.05	1.00
34	3.19	2.39	2.79	0.89	1.00	1.00	1.00	0.89	1.20	1.32	1.43
35	2.22	2.19	1.85	1.00	1.00	1.00	1.00	0.86	0.82	0.89	0.88
36	0.60	0.85	1.60	1.00	1.36	1.37	1.18	0.87	0.97	0.85	1.04
37	0.58	0.80	1.39	1.00	1.28	0.95	1.00	0.88	1.00	0.87	1.27
38	7.16	1.43	1.42	0.97	2.27	2.46	2.51	0.70	0.76	0.89	1.23
39	1.56	1.12	0.95	1.00	0.76	1.00	1.00	0.88	0.88	0.86	0.87
40	5.82	0.93	0.77	1.00	1.00	1.00	1.00	0.76	0.62	0.59	0.64
41	4.16	1.40	1.37	1.00	1.00	1.00	1.00	1.03	1.94	1.13	1.02

Table 3.9 Relative MSFE for the real time series. Results are normalised with the no-break model so that the no-break model always have the relative MSFE equals to 1.

real datasets	NOT	NOT-ARMA-1			NOT-ARMA-2		
		(CV-min)	(CV-1SE)	(BIC)	(CV-min)	(CV-1SE)	(BIC)
1	15.50	16.80	2.90	0.80	0.50	0.00	1.00
2	11.50	27.50	11.10	0.00	0.60	0.00	1.00
3	8.60	8.40	0.00	0.00	0.50	0.00	0.00
4	15.00	7.60	5.30	0.00	0.90	0.50	0.00
5	1.00	20.00	11.00	0.00	0.50	0.20	1.00
6	2.00	7.80	2.30	0.00	2.00	0.30	0.00
7	11.00	3.90	3.90	0.00	0.60	0.60	0.50
8	5.00	56.10	14.60	1.50	0.80	0.60	1.40
9	6.00	13.00	4.70	0.00	3.70	0.00	0.00
10	5.00	11.40	2.70	0.00	4.60	0.80	0.00
11	0.00	14.40	1.60	1.00	5.90	3.00	0.00
12	5.00	5.30	0.20	0.00	3.40	0.90	1.50
13	0.00	6.30	1.70	0.00	3.60	1.00	0.00
14	14.20	11.50	3.20	0.00	0.20	0.00	0.00
15	15.50	5.80	0.00	0.00	0.90	0.00	0.00
16	18.20	9.20	5.90	0.90	0.40	0.00	0.00
17	21.60	3.30	0.20	0.00	6.70	3.60	0.00
18	0.00	12.70	3.70	0.00	6.10	3.10	0.00
19	9.20	20.00	11.60	0.00	3.30	0.60	0.80
20	19.80	2.30	0.30	0.00	0.40	0.10	0.00
21	0.00	12.20	6.10	0.00	1.60	0.20	0.00
22	12.80	21.90	18.40	0.00	0.30	0.20	1.00
23	14.60	41.20	23.00	0.00	0.20	0.00	0.00
24	19.00	10.50	2.10	0.00	0.20	0.00	0.00
25	24.60	8.30	3.10	0.00	0.50	0.00	0.00
26	12.90	25.40	5.30	0.00	1.10	0.00	0.00
27	8.20	12.30	5.00	0.00	1.30	0.50	0.00
28	14.60	41.20	23.00	0.00	0.20	0.00	0.00
29	4.00	17.00	11.60	0.00	0.00	0.00	0.00
30	12.90	17.00	10.60	0.00	0.30	0.00	0.00
31	10.80	2.00	0.30	0.00	0.50	0.00	0.00
32	24.30	34.90	26.30	0.00	0.00	0.00	0.00
33	22.90	33.60	13.70	0.00	0.00	0.00	0.00
34	4.00	25.90	19.70	1.00	0.00	0.00	0.00
35	22.90	23.10	22.80	0.00	0.00	0.00	0.00
36	9.00	12.00	3.40	0.00	1.40	0.70	0.70
37	9.00	12.10	6.00	0.00	1.20	0.10	0.00
38	8.90	7.50	6.60	0.90	0.90	0.70	0.70
39	17.90	9.10	6.60	0.00	3.00	0.00	0.00
40	9.40	12.60	10.70	0.00	0.00	0.00	0.00
41	23.60	29.20	20.40	0.00	0.00	0.00	0.00

Table 3.10 The average number of change-points estimated, when the change-point models are fitted on the real datasets.

3.6 Conclusion and discussion

In this chapter, we extend the use of NOT for prediction on time series with dependent noise by proposing some new methods that make use of the change-points detected on the NOT solution path. We compare the prediction performance of the newly proposed procedures to the original NOT, the no-break model (i.e. model fitted using all available data) and some robust methods like averaging window. The NOT-ARMA-1(BIC) provides the best prediction performance on both simulated and real datasets among all the methods proposed in this chapter. It often outperforms the no-break models, showing the benefit of incorporating estimated change-points for the prediction purpose. The NOT-CV methods proposed for the threshold selection in this chapter usually perform worse than the baseline method. Further analysis and modification of the methodology are needed to improve their performance for prediction.

The robust method averaging window provides good prediction performance (relative to the baseline no change point model) in the simulation settings, and the relative performance of the robust methods are consistent with the findings from the previous works. NOT-ARMA-1(BIC) and NOT-ARMA-2(BIC) outperform the averaging window when the size of jumps is larger, but it is not the case when the jump size is smaller. This is again consistent with previous work - which methods perform the best depends on the size of the jumps. When the size of jumps is relatively large, using methods that take into account the change-points often perform better. When the size of jumps is relatively small, robust methods like averaging often provide a better prediction.

3.6.1 Rules of thumb for time series forecasting in the presence of change-points

Given the findings from the literature and observations from our simulation results, we suggest the rules of thumb for time series forecasting in the presence of change-points as follows:

- Time series with easy-to-detect change-points: As our simulation results for models 1s-7s and 1e-5e show, it is better to incorporate the estimated change-points for forecasting
- Long time series with hard-to-detect change-points. As our simulation results for models 1a-5a show, if the change-points are given, incorporating the change-points still provide good prediction. However, given the change-points are difficult to be detected accurately, it is better to use the robust methods in reality.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE transactions on automatic control 19*(6), 716–723. pages 87

Aminikhanghahi, S. and D. J. Cook (2017). A survey of methods for time series change point detection. *Knowledge and information systems 51*(2), 339–367. pages 77

Andreou, E. and E. Ghysels (2002). Detecting multiple breaks in financial market volatility dynamics. *Journal of Applied Econometrics 17*(5), 579–600. pages 75

Andreou, E. and E. Ghysels (2009). Structural breaks in financial time series. In *Handbook of financial time series*, pp. 839–870. Springer. pages 75

Askitas, N. and K. F. Zimmermann (2009). Google econometrics and unemployment forecasting. pages 166

Bach, F. R. (2008). Bolasso: model consistent lasso estimation through the bootstrap. In *Proceedings of the 25th international conference on Machine learning*, pp. 33–40. ACM. pages 34

Baek, C. and V. Pipiras (2012). Statistical tests for a single change in mean against long-range dependence. *Journal of Time Series Analysis 33*(1), 131–151. pages 90, 91

Baek, C., V. Pipiras, et al. (2014). On distinguishing multiple changes in mean and long-range dependence using local whittle estimation. *Electronic Journal of Statistics 8*(1), 931–964. pages 90, 91

Bagnall, A., A. Bostrom, J. Large, and J. Lines (2016). The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version. *arXiv preprint arXiv:1602.01711*. pages 92

Bai, J. (1997). Estimation of a change point in multiple regression models. *Review of Economics and Statistics 79*(4), 551–563. pages 74

Bai, J. et al. (2000). Vector autoregressive models with structural changes in regression coefficients and in variance-covariance matrices. Technical report, China Economics and Management Academy, Central University of Finance and Economics. pages 84

Bai, J. and P. Perron (2006). Multiple structural change models: a simulation analysis. *Econometric theory and practice: Frontiers of analysis and applied research 1*, 212–237. pages 81, 85

Baillie, R. T. (1996). Long memory processes and fractional integration in econometrics. *Journal of econometrics 73*(1), 5–59. pages 73

Baranowski, R., Y. Chen, and P. Fryzlewicz (2018). Ranking-based variable selection for high-dimensional data. *Stat. Sin*, 1–32. pages 34

Baranowski, R., Y. Chen, and P. Fryzlewicz (2019). Narrowest-over-threshold detection of multiple change points and change-point-like features. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 81*(3), 649–672. pages 77, 78, 80, 81, 85, 128, 140, 142, 143, 144, 151, 153, 154, 156, 157

Barkoulas, J. T., C. F. Baum, and N. Travlos (2000). Long memory in the greek stock market. *Applied Financial Economics 10*(2), 177–184. pages 75

Beadle, E. R. and P. M. Djuric (1997). Uniform random parameter generation of stable minimum-phase real arma (p, q) processes. *IEEE signal processing letters 4*(9), 259–261. pages 95

Beale, E., M. Kendall, and D. Mann (1967). The discarding of variables in multivariate analysis. *Biometrika 54*(3-4), 357–366. pages 27

Beran, J., R. J. Bhansali, and D. Ocker (1998). On unified model selection for stationary and nonstationary short-and long-memory autoregressive processes. *Biometrika 85*(4), 921–934. pages 83

Berkes, I., L. Horváth, P. Kokoszka, Q.-M. Shao, et al. (2006). On discriminating between long-range dependence and changes in mean. *The annals of statistics 34*(3), 1140–1165. pages 68, 90, 91

Braun, J. V., R. Braun, and H.-G. Müller (2000). Multiple changepoint fitting via quasilikelihood, with application to dna sequence segmentation. *Biometrika 87*(2), 301–314. pages 75

Brooks, R. J. and A. M. Tobias (1996). Choosing the best model: Level of detail, complexity, and model performance. *Mathematical and computer modelling 24*(4), 1–14. pages 105

Caiado, J., N. Crato, and D. Peña (2006). A periodogram-based metric for time series classification. *Computational Statistics & Data Analysis 50*(10), 2668–2684. pages 92

Candes, E. and T. Tao (2007). The Dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, 2313–2351. pages 44

Chatterjee, A. and S. N. Lahiri (2011). Bootstrapping Lasso estimators. *Journal of the American Statistical Association 106*(494), 608–625. pages 28

Chen, J. and Z. Chen (2008). Extended Bayesian information criteria for model selection with large model spaces. *Biometrika*, 759–771. pages 30, 34, 42, 56, 63

Chen, J., A. Gupta, and J. Pan (2006). Information criterion and change point problem for regular models. *Sankhyā: The Indian Journal of Statistics*, 252–282. pages 79, 86

Chen, J. and R. B. Randall (2016). Intelligent diagnosis of bearing knock faults in internal combustion engines using vibration simulation. *Mechanism and Machine Theory 104*, 161–176. pages 97

Dau, H. A., E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML (2018, October). The ucr time series classification archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/. pages 92

Davies, P. L. (2014). *Data analysis and approximate models: Model choice, Location-Scale, analysis of variance, nonparametric regression and image analysis*. CRC Press. pages 70, 71, 99, 102

Davis, R. A., S. A. Hancock, and Y.-C. Yao (2016). On consistency of minimum description length model selection for piecewise autoregressions. *Journal of Econometrics 194*(2), 360–368. pages 79, 89

Davis, R. A., T. C. Lee, and G. A. Rodriguez-Yam (2008). Break detection for a class of nonlinear time series models. *Journal of Time Series Analysis 29*(5), 834–867. pages 89

Diebold, F. X. and A. Inoue (2001). Long memory and regime switching. *Journal of econometrics 105*(1), 131–159. pages 65

D'Amuri, F. and J. Marcucci (2010). 'google it!'forecasting the us unemployment rate with a google job search index. pages 166

Eklund, J., G. Kapetanios, and S. Price (2010). Forecasting in the presence of recent structural change. pages 130, 136, 137, 138, 139, 151, 152, 167

Elkordy, M. F., K.-C. Chang, and G. C. Lee (1993). Neural networks trained by analytically simulated damage states. *Journal of Computing in Civil Engineering 7*(2), 130–145. pages 96

Er-raoudi, M., M. Diany, H. Aissaoui, and M. Mabrouki (2016). Gear fault detection using artificial neural networks with discrete wavelet transform and principal component analysis. *J Mech Eng Sci 10*, 2016–2029. pages 97

Fan, J. and R. Li (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association 96*(456), 1348–1360. pages 27, 42, 55

Fan, J. and J. Lv (2010). A selective overview of variable selection in high dimensional feature space. *Statistica Sinica 20*(1), 101. pages 28

Fawaz, H. I., G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery 33*(4), 917–963. pages 68, 92, 93, 94, 109, 110

Fondeur, Y. and F. Karamé (2013). Can google data help predict french youth unemployment? *Economic Modelling 30*, 117–125. pages 166

Frid-Adar, M., E. Klang, M. Amitai, J. Goldberger, and H. Greenspan (2018). Synthetic data augmentation using gan for improved liver lesion classification. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pp. 289–293. IEEE. pages 98

Fryzlewicz, P. (2020). Detecting possibly frequent change-points: Wild binary segmentation 2 and steepest-drop model selection. *Journal of the Korean Statistical Society*, 1–44. pages 80

Fryzlewicz, P. et al. (2014). Wild binary segmentation for multiple change-point detection. *The Annals of Statistics 42*(6), 2243–2281. pages 80, 85

Fryzlewicz, P. and H. Ombao (2009). Consistent classification of nonstationary time series using stochastic wavelet representations. *Journal of the American Statistical Association 104*(485), 299–312. pages 93, 99

Fukuda, K. (2009). Distribution switching of stock returns: international evidence. *Applied Financial Economics 19*(5), 371–377. pages 83

Gecgel, O., S. Ekwaro-Osire, J. P. Dias, A. Nispel, F. M. Alemayehu, and A. Serwadda (2018). Machine learning in crack size estimation of a spur gear pair using simulated vibration data. In *International Conference on Rotor Dynamics*, pp. 175–190. Springer. pages 97

Giraitis, L., G. Kapetanios, and S. Price (2013). Adaptive forecasting in the presence of recent and ongoing structural change. *Journal of Econometrics 177*(2), 153–170. pages 138, 139, 140, 144, 145, 152

Granger, C. W. and N. Hyung (2004). Occasional structural breaks and long memory with an application to the s&p 500 absolute stock returns. *Journal of empirical finance 11*(3), 399–421. pages 65, 84

Granger, C. W. and R. Joyeux (1980). An introduction to long-memory time series models and fractional differencing. *Journal of time series analysis 1*(1), 15–29. pages 74

Granger, C. W. J. and Z. Ding (1995). Some properties of absolute return: An alternative measure of risk. *Annales d'Economie et de Statistique*, 67–91. pages 65

Greene, M. T. and B. D. Fielitz (1977). Long-term dependence in common stock returns. *Journal of Financial Economics 4*(3), 339–349. pages 65, 74

Hall, A., D. Osborn, and N. Sakkas (2013a). The asymptotic expectation of the residual sum of squares in linear models with multiple break points. *Unpublished Mimeo, Department of Economics, University of Manchester, Manchester, UK*. pages 88, 108, 112

Hall, A. R., D. R. Osborn, and N. Sakkas (2013b). Inference on structural breaks using information criteria. *The Manchester School 81*, 54–81. pages 81, 82, 85, 88

Hall, A. R., D. R. Osborn, and N. Sakkas (2015). Structural break inference using information criteria in models estimated by two-stage least squares. *Journal of Time Series Analysis 36*(5), 741–762. pages 112

Hansen, M. H. and B. Yu (2001). Model selection and the principle of minimum description length. *Journal of the American Statistical Association 96*(454), 746–774. pages 89

Harrison Jr, D. and D. L. Rubinfeld (1978). Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management 5*(1), 81–102. pages 58

Hassler, U. and B. Meller (2014). Detecting multiple breaks in long memory the case of us inflation. *Empirical Economics 46*(2), 653–680. pages 126

Hassler, U. and J. Wolters (1995). Long memory in inflation rates: International evidence. *Journal of Business & Economic Statistics 13*(1), 37–45. pages 65, 75

Hawkins, D. M. (1977). Testing a sequence of observations for a shift in location. *Journal of the American Statistical Association 72*(357), 180–186. pages 77

He, K., X. Zhang, S. Ren, and J. Sun (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778. pages 68, 92, 93, 94

Henry, M. and P. Zaffaroni (2003). The long-range dependence paradigm for macroeconomics and finance. *Theory and applications of long-range dependence*, 417–438. pages 74

Henry, O. T. (2002). Long memory in stock returns: some international evidence. *Applied financial economics 12*(10), 725–729. pages 65

Hinkley, D. V. (1970). Inference about the change-point in a sequence of random variables. pages 77

Hosking, J. (1981). Fractional differencing. biometrika 68 165–176. *Mathematical Reviews (MathSciNet): MR614953 Zentralblatt MATH 464*. pages 74

Hsu, C.-C. (2005). Long memory or structural changes: An empirical examination on inflation rates. *Economics Letters 88*(2), 289–294. pages 126

Huang, H.-Y., H. Ombao, and D. S. Stoffer (2004). Discrimination and classification of nonstationary time series using the slex model. *Journal of the American Statistical Association 99*(467), 763–774. pages 93

Hyung, N. and P. H. Franses (2001). Structural breaks and long memory in us inflation rates: Do they matter for forecasting? Technical report. pages 76

Jackson, B., J. D. Scargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. Sangtrakulcharoen, L. Tan, and T. T. Tsai (2005). An algorithm for optimal partitioning of data on an interval. *IEEE Signal Processing Letters 12*(2), 105–108. pages 79

Javanmard, A., A. Montanari, et al. (2018). Debiasing the Lasso: Optimal sample size for gaussian designs. *The Annals of Statistics 46*(6A), 2593–2622. pages 28

Killick, R. and I. Eckley (2014). changepoint: An r package for changepoint analysis. *Journal of statistical software 58*(3), 1–19. pages 77

Killick, R., P. Fearnhead, and I. A. Eckley (2012). Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association 107*(500), 1590–1598. pages 79, 114

Kley, T., P. Preuß, P. Fryzlewicz, et al. (2019). Predictive, finite-sample model choice for time series under stationarity and non-stationarity. *Electronic Journal of Statistics 13*(2), 3710–3774. pages 139

Koepcke, L., G. Ashida, and J. Kretzberg (2016). Single and multiple change point detection in spike trains: comparison of different cusum methods. *Frontiers in systems neuroscience 10*, 51. pages 75

Kovács, S., H. Li, P. Bühlmann, and A. Munk (2020). Seeded binary segmentation: A general methodology for fast and optimal change point detection. *arXiv preprint arXiv:2002.06633*. pages 80

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105. pages 94

Krzemieniewska, K., I. A. Eckley, and P. Fearnhead (2014). Classification of non-stationary time series. *Stat 3*(1), 144–157. pages 93, 99

Kurozumi, E. and P. Tuvaandorj (2011). Model selection criteria in multivariate models with multiple structural changes. *Journal of Econometrics 164*(2), 218–238. pages 87, 88, 108, 112

LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *nature 521*(7553), 436–444. pages 68

Lee, J. D., D. L. Sun, Y. Sun, J. E. Taylor, et al. (2016). Exact post-selection inference, with application to the lasso. *The Annals of Statistics 44*(3), 907–927. pages 28

Levin, A. T. and J. Piger (2002). Is inflation persistence intrinsic in industrial economies? *FRB of St. Louis Working Paper No.* pages 68, 76

Lim, C. and B. Yu (2016). Estimation stability with cross-validation (escv). *Journal of Computational and Graphical Statistics 25*(2), 464–492. pages 56

Liu, H., B. Yu, et al. (2013). Asymptotic properties of lasso+ mls and lasso+ ridge in sparse high-dimensional linear regression. *Electronic Journal of Statistics 7*, 3124–3169. pages 28

Liu, J., S. Wu, and J. V. Zidek (1997). On segmented multivariate regression. *Statistica Sinica*, 497–525. pages 79, 81, 85, 86, 113

Lloyd, J. R., D. Duvenaud, R. Grosse, J. Tenenbaum, and Z. Ghahramani (2014). Automatic construction and natural-language description of nonparametric regression models. In *Twenty-eighth AAAI conference on artificial intelligence*. pages 84

Loftus, J. R. and J. E. Taylor (2014). A significance test for forward stepwise model selection. *arXiv preprint arXiv:1405.3920*. pages 53

Mares, M. A., S. Wang, and Y. Guo (2016). Combining multiple feature selection methods and deep learning for high-dimensional data. *Transactions on Machine Learning and Data Mining 9*, 22–45. pages 36

McLeod, A. I. and K. W. Hipel (1978). Preservation of the rescaled adjusted range: 1. a reassessment of the hurst phenomenon. *Water Resources Research 14*(3), 491–508. pages 73

Meinshausen, N. (2007). Relaxed lasso. *Computational Statistics & Data Analysis 52*(1), 374–393. pages 29, 55

Meinshausen, N. and P. Bühlmann (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 72*(4), 417–473. pages 34, 42

Mirza, M. and S. Osindero (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*. pages 70, 97

Moon, J., M. B. Hossain, and K. H. Chon (2021). Ar and arma model order selection for time-series modeling with imagenet classification. *Signal Processing*, 108026. pages 95

Muggeo, V. M. and G. Adelfio (2011). Efficient change point detection for genomic sequences of continuous measurements. *Bioinformatics 27*(2), 161–166. pages 75

Murphey, Y. L., M. A. Masrur, Z. Chen, and B. Zhang (2006). Model-based fault diagnosis in electric drives using machine learning. *IEEE/ASME Transactions On Mechatronics 11*(3), 290–303. pages 70, 96

Myung, I. J. (2000). The importance of complexity in model selection. *Journal of mathematical psychology 44*(1), 190–204. pages 105

Nan, Y. and Y. Yang (2014). Variable selection diagnostics measures for high-dimensional regression. *Journal of Computational and Graphical Statistics 23*(3), 636–656. pages 35, 36

Ninomiya, Y. (2005). *Statistics & Probability Letters 72*(3), 237–247. pages 87, 88

Norwood, B. and R. Killick (2018). Long memory and changepoint models: a spectral classification procedure. *Statistics and Computing 28*(2), 291–302. pages 8, 68, 69, 70, 71, 75, 76, 97, 98, 99, 100, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 113, 114, 116, 117, 119, 124, 126

Pan, J. and J. Chen (2006). Application of modified information criterion to multiple change point problems. *Journal of multivariate analysis 97*(10), 2221–2241. pages 87

Pan, J.-N. and S.-T. Chen (2008). Monitoring long-memory air quality data using arfima model. *Environmetrics: The official journal of the International Environmetrics Society 19*(2), 209–219. pages 75

Perez, L. and J. Wang (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*. pages 97

Pesaran, M. H. and A. Pick (2011). Forecast combination across estimation windows. *Journal of Business & Economic Statistics 29*(2), 307–318. pages 132, 134

Pesaran, M. H., A. Pick, and M. Pranovich (2013). Optimal forecasts in the presence of structural breaks. *Journal of Econometrics 177*(2), 134–152. pages 133, 134, 135, 139

Pesaran, M. H. and A. Timmermann (2004). How costly is it to ignore breaks when forecasting the direction of a time series? *International Journal of Forecasting 20*(3), 411–425. pages 76

Pesaran, M. H. and A. Timmermann (2005). Small sample properties of forecasts from autoregressive models under structural breaks. *Journal of Econometrics 129*(1-2), 183–217. pages 129, 135, 136, 139, 151

Pesaran, M. H. and A. Timmermann (2007). Selection of estimation window in the presence of breaks. *Journal of Econometrics 137*(1), 134–161. pages 129, 130, 131, 132, 134, 135, 138, 139, 140, 144, 145, 151, 152

Pivetta, F. and R. Reis (2007). The persistence of inflation in the united states. *Journal of Economic dynamics and control 31*(4), 1326–1358. pages 65

Pohjalainen, J., O. Räsänen, and S. Kadioglu (2015). Feature selection methods and their combinations in high-dimensional classification of speaker likability, intelligibility and personality traits. *Computer Speech & Language 29*(1), 145–171. pages 36

Pötscher, B. (1990). Estimation of autoregressive moving-average order given an infinite number of models and approximation of spectral densities. *Journal of Time Series Analysis 11*(2), 165–179. pages 83

Powers, D. (2011). Evaluation: From precision, recall and fmeasure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies 2*, 37–63. pages 57

Reeves, J., J. Chen, X. L. Wang, R. Lund, and Q. Q. Lu (2007). A review and comparison of changepoint detection techniques for climate data. *Journal of applied meteorology and climatology 46*(6), 900–915. pages 75

Schröder, A. L. and P. Fryzlewicz (2013). Adaptive trend estimation in financial time series via multiscale change-point-induced basis recovery. pages 75

Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics 6*(2), 461–464. pages 69, 82

Serrà, J., S. Pascual, and A. Karatzoglou (2018). Towards a universal neural network encoder for time series. In *CCIA*, pp. 120–129. pages 92

Shah, R. D. and R. J. Samworth (2013). Variable selection with error control: another look at stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 75*(1), 55–80. pages 34, 42

Simonyan, K. and A. Zisserman (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. pages 94

Sobie, C., C. Freitas, and M. Nicolai (2018). Simulation-driven machine learning: Bearing fault classification. *Mechanical Systems and Signal Processing 99*, 403–419. pages 70, 96

Song, H. and D. W. Shin (2015). Long-memories and mean breaks in realized volatilities. *Applied Economics Letters 22*(16), 1273–1280. pages 69, 83, 126

Stamey, T. A., J. N. Kabalin, J. E. McNeal, I. M. Johnstone, F. Freiha, E. A. Redwine, and N. Yang (1989). Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. ii. radical prostatectomy treated patients. *The Journal of urology 141*(5), 1076–1083. pages 29

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9. pages 94

Tang, W. H. and A. Röllin (2018). Model identification for arma time series through convolutional neural networks. *arXiv preprint arXiv:1804.04299*. pages 95

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288. pages 27, 55, 58, 146

Tibshirani, R. J., A. Rinaldo, R. Tibshirani, L. Wasserman, et al. (2018). Uniform asymptotic inference and the bootstrap after model selection. *The Annals of Statistics 46*(3), 1255–1287. pages 28

Tibshirani, R. J., J. Taylor, R. Lockhart, and R. Tibshirani (2016). Exact post-selection inference for sequential regression procedures. *Journal of the American Statistical Association 111*(514), 600–620. pages 28, 53

Tsai, C.-F. and Y.-C. Hsiao (2010). Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. *Decision Support Systems 50*(1), 258–269. pages 36

Van de Geer, S., P. Bühlmann, Y. Ritov, R. Dezeure, et al. (2014). On asymptotically optimal confidence regions and tests for high-dimensional models. *The Annals of Statistics 42*(3), 1166–1202. pages 28

Wang, X., P. Peng, and D. B. Dunson (2014). Median selection subset aggregation for parallel inference. In *Advances in Neural Information Processing Systems*, pp. 2195–2203. pages 34

Wang, X. L., Q. H. Wen, and Y. Wu (2007). Penalized maximal t test for detecting undocumented mean change in climate data series. *Journal of Applied Meteorology and Climatology 46*(6), 916–931. pages 75

Wang, Z., W. Yan, and T. Oates (2017). Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pp. 1578–1585. IEEE. pages 68, 92, 93, 109

Warren, D. L. and S. N. Seifert (2011). Ecological niche modeling in maxent: the importance of model complexity and the performance of model selection criteria. *Ecological applications 21*(2), 335–342. pages 105

Yang, W. and Y. Yang (2017). Toward an objective and reproducible model choice via variable selection deviation. *Biometrics 73*(1), 20–30. pages 36, 56

Yang, Y. (2001). Adaptive regression by mixing. *Journal of the American Statistical Association 96*(454), 574–588. pages 35, 42

Yao, Y.-C. (1988). Estimating the number of change-points via schwarz'criterion. *Statistics & Probability Letters 6*(3), 181–189. pages 69, 79, 83, 84, 86, 88, 89, 113

Yao, Y.-C. and S.-T. Au (1989). Least-squares estimation of a step function. *Sankhyā: The Indian Journal of Statistics, Series A*, 370–381. pages 79

Yau, C. Y. and R. A. Davis (2012). Likelihood inference for discriminating between long-memory and change-point models. *Journal of Time Series Analysis 33*(4), 649–664. pages 17, 65, 68, 91, 99, 103, 113, 118

Ye, C., Y. Yang, and Y. Yang (2018). Sparsity oriented importance learning for high-dimensional linear regression. *Journal of the American Statistical Association 113*(524), 1797–1812. pages 36

Ye, L. and E. Keogh (2009). Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 947–956. pages 92

Yuan, M. and Y. Lin (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 68*(1), 49–67. pages 27

Yuan, Z. and Y. Yang (2005). Combining linear regression models: When and how? *Journal of the American Statistical Association 100*(472), 1202–1214. pages 35, 42

Zhang, C.-H. et al. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics 38*(2), 894–942. pages 27, 55

Zhang, C.-H. and S. S. Zhang (2014). Confidence intervals for low dimensional parameters in high dimensional linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 76*(1), 217–242. pages 28, 53

Zhang, Y. and Y. Yang (2015). Cross-validation for selecting a model selection procedure. *Journal of Econometrics 187*(1), 95–112. pages 30, 34, 42, 58, 63

Zhao, P. and B. Yu (2006). On model selection consistency of lasso. *Journal of Machine learning research 7*(Nov), 2541–2563. pages 28

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association 101*(476), 1418–1429. pages 44

Zou, H. and T. Hastie (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67*(2), 301–320. pages 27, 55

Appendix A

Appendix for Chapter 1

A.1 Proof of Proposition 1.1

Proof.

$$
\begin{aligned}
& \arg\min_{M} \frac{1}{|\mathscr{M}|} \sum_{\{k|M_k \in \mathscr{M}\}} Hamming(M_k, M) \\
\Longleftrightarrow\ & \arg\min_{M} \frac{1}{|\mathscr{M}|} \sum_{\{k|M_k \in \mathscr{M}\}} \sum_{j=1}^{p} (\mathbb{1}_{\beta_j^k=0} \mathbb{1}_{j\in M} + \mathbb{1}_{\ beta_j^k \neq 0} \mathbb{1}_{j \notin M}) \\
\Longleftrightarrow\ & \arg\min_{M} \frac{1}{|\mathscr{M}|} \sum_{j=1}^{p} \sum_{\{k|M_k \in \mathscr{M}\}} [(1-\mathbb{1}_{\beta_j^k\neq 0})\mathbb{1}_{j\in M} + \mathbb{1}_{\beta_j^k \neq 0}(1-\mathbb{1}_{j\in M})] \\
\Longleftrightarrow\ & \arg\min_{M} \frac{1}{|\mathscr{M}|} \sum_{j=1}^{p} \sum_{\{k|M_k \in \mathscr{M}\}} [\mathbb{1}_{j\in M}(1-2\mathbb{1}_{\beta_j^k \neq 0})] \\
\Longleftrightarrow\ & \arg\min_{M} \sum_{j=1}^{P} \mathbb{1}_{j\in M}(1-2\frac{\sum_{\{k|M_k\in\mathscr{M}\}} \mathbb{1}_{\beta_j^k\neq 0}}{|\mathscr{M}|}) \\
\Longleftrightarrow\ & \arg\min_{M} \sum_{j=1}^{P} \mathbb{1}_{j\in M}(1-2\tau_j)
\end{aligned}
$$

□

A.2 Proof of Proposition 1.2

Proof. Let the number of fitted models for covariance j to be estimated with positive sign to be n_j^+ and those estimated with negative sign to be n_j^-.

$$\begin{aligned}
&\arg\min_{M} \frac{1}{|\mathscr{M}|} \sum_{\{k|M_k \in \mathscr{M}\}} dist(M_k, M) \\
\iff &\frac{1}{|\mathscr{M}|} \arg\min_{\{s_j^M|j=1,\ldots,p\}} \sum_{\{k|M_k \in \mathscr{M}\}} |s_j^M - sign(\beta_j^k)| \\
\iff &\frac{1}{|\mathscr{M}|} \arg\min_{\{s_j^M|j=1,\ldots,p\}} \sum_{\{k|M_k \in \mathscr{M}\}} [(|\mathscr{M}| - n_j^+ + n_j^-)\mathbb{1}_{s_j^M=+1} + (n_j^+ + n_j^-)\mathbb{1}_{s_j^M=0} + (|\mathscr{M}| + n_j^+ - n_j^-)\mathbb{1}_{s_j^M=-1}] \\
\iff &\arg\min_{\{s_j^M|j=1,\ldots,p\}} \sum_{\{k|M_k \in \mathscr{M}\}} [(1 - \tau_j^+ + \tau_j^-)\mathbb{1}_{s_j^M=+1} + (\tau_j^+ + \tau_j^-)\mathbb{1}_{s_j^M=0} + (1 - \tau_j^- + \tau_j^+)\mathbb{1}_{s_j^M=-1}]
\end{aligned}$$

which is equivalent to setting

$$s_j^M = \begin{cases} +1 \text{ if } \tau_j^+ \geq 1/2 \\ -1 \text{ if } \tau_j^- \geq 1/2 \\ 0 \text{ otherwise} \end{cases}$$

□

A.3 Illustration of selection disagreement via Jaccard distance heatmap

The heat maps in Figure A.1-A.5 show the average Jaccard distance ($dist_{Jaccard}(A,B) = \frac{|A \triangle B|}{|A \cup B|}$, where $A \triangle B = (A \cup B) - (A \cap B)$) among selected sets from five different variable selection methods under simulation setting 1-5 (see Section 5 more details). The Jaccard distance shows the relative size of selection disagreement. It ranges from 0 to 1 and larger distance means larger dissimilarity between two selected sets. The Jaccard distance among some selection methods can be quite large, showing that relatively large number of covariates are selected by one method but not by another. Even for methods that are relatively similar like the Lasso and elastic Lasso, the difference among them is still not negligible for most settings.

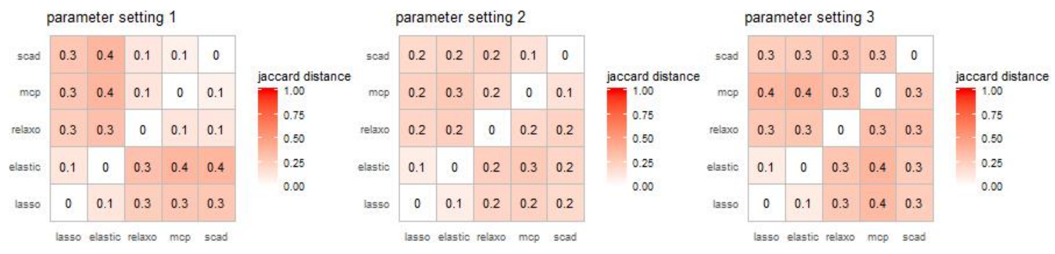

Fig. A.1 Average Jaccard distance among fitted models from different methods using simulated data from model 1

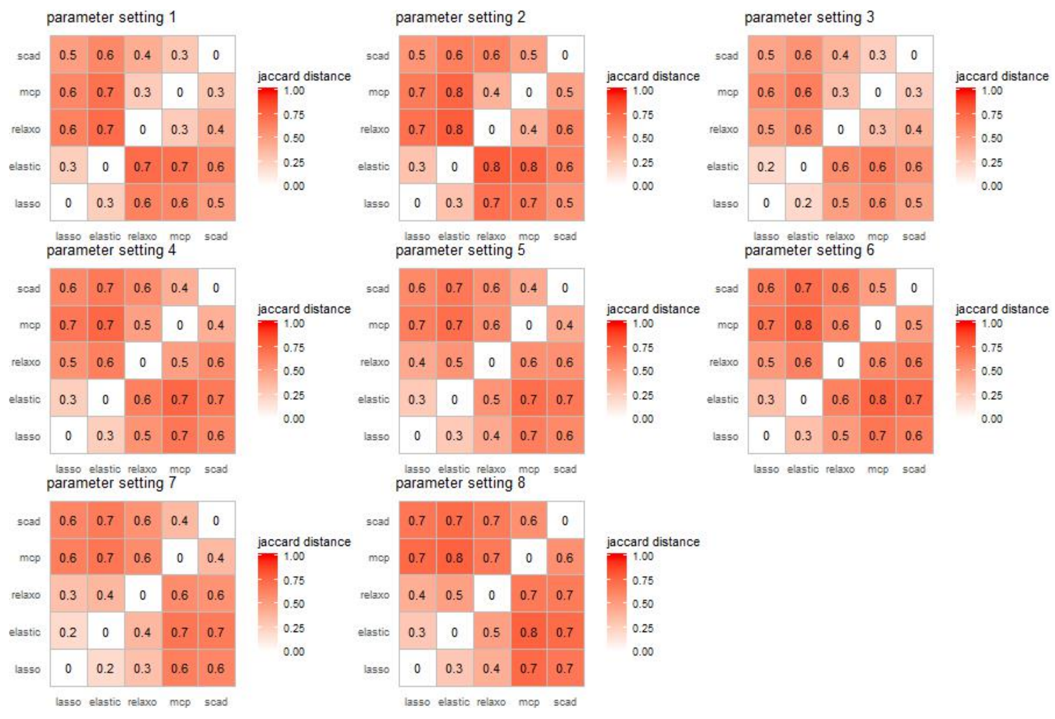

Fig. A.2 Average Jaccard distance among fitted models from different methods using simulated data from model 2

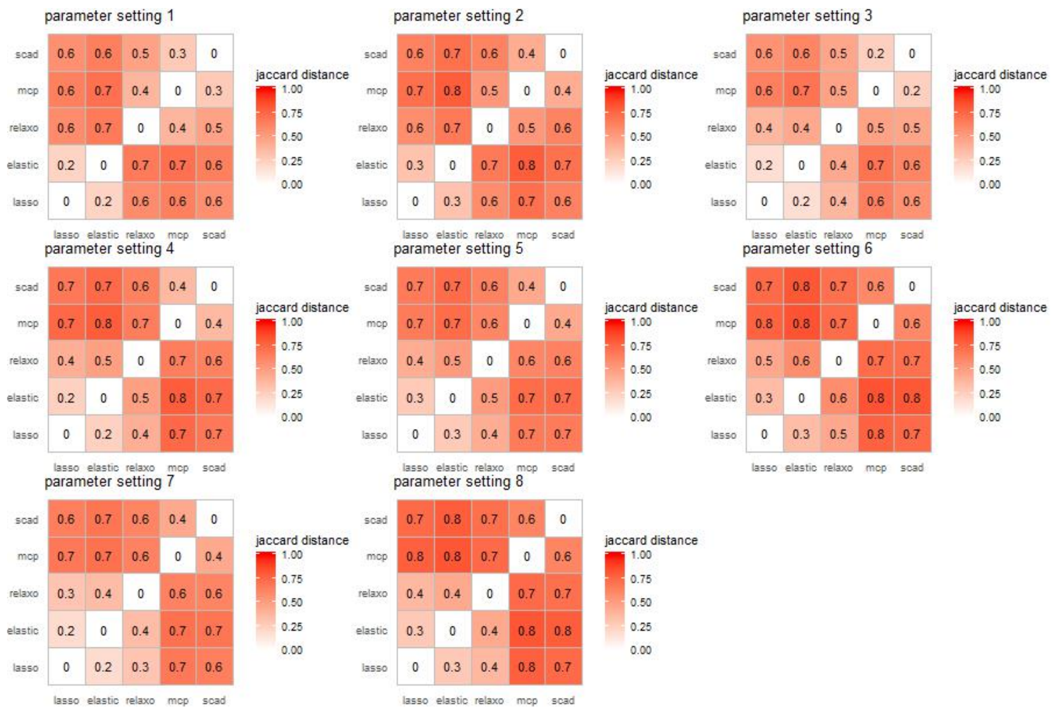

Fig. A.3 Average Jaccard distance among fitted models from different methods using simulated data from model 3

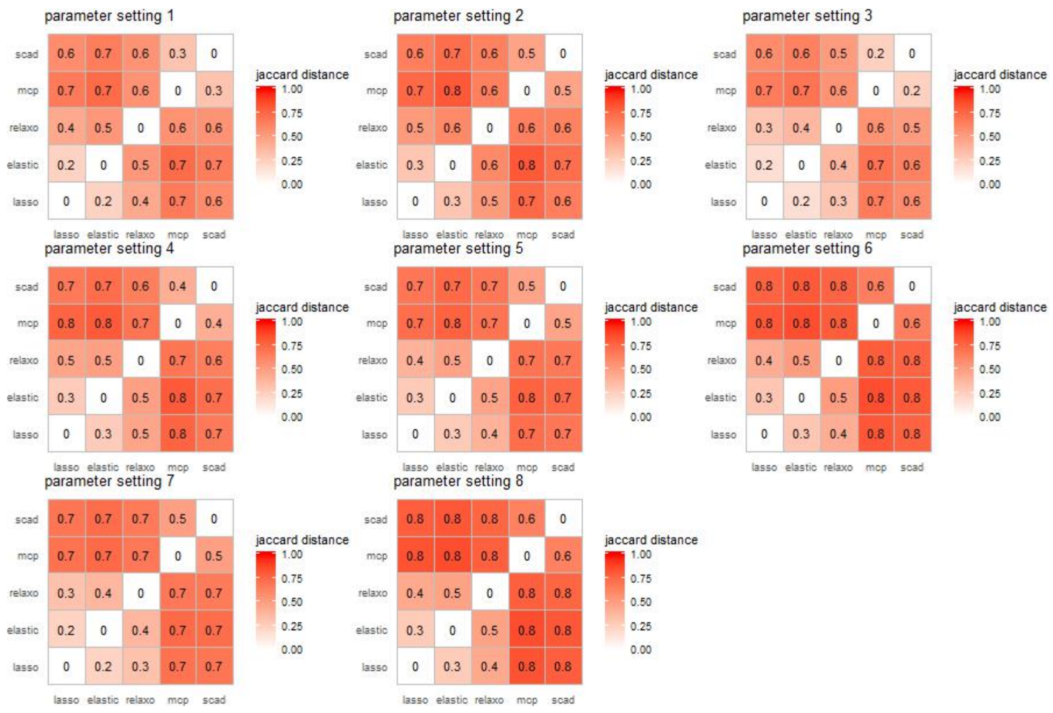

Fig. A.4 Average Jaccard distance among fitted models from different methods using simulated data from model 4

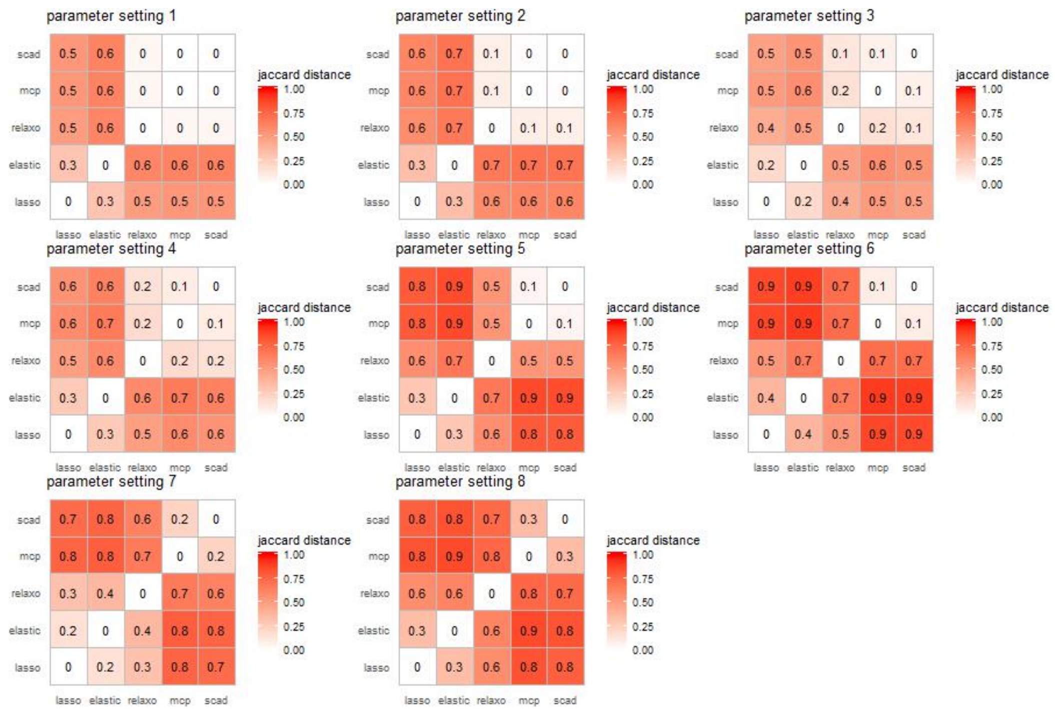

Fig. A.5 Average Jaccard distance among fitted models from different methods using simulated data from model 5

A.4 Tables of the simulation results

parameters	methods	f	FP+FN	FP	FN	pred.err	l1.diff	l2.diff	size
setting 1	lasso	0.76 (0.01)	2.11 (0.14)	2.11	0	1.13 (0.02)	0.75 (0.04)	0.39 (0.02)	5.11 (0.14)
	elastic net	0.71 (0.01)	2.67 (0.14)	2.67	0	1.15 (0.03)	0.85 (0.04)	0.42 (0.02)	5.67 (0.14)
rho = 0.5	relaxed lasso	0.91 (0.01)	0.7 (0.1)	0.7	0	1.11 (0.03)	0.6 (0.04)	0.34 (0.02)	3.7 (0.1)
s = 3	mcp	0.94 (0.01)	0.47 (0.1)	0.47	0	1.1 (0.02)	0.53 (0.04)	0.32 (0.02)	3.47 (0.1)
p = 8	scad	0.94 (0.01)	0.53 (0.11)	0.53	0	1.09 (0.02)	0.51 (0.04)	0.31 (0.02)	3.53 (0.11)
	vsd	0.98 (0)	0.11 (0.03)	0.11	0	1.08 (0.02)	0.43 (0.02)	0.28 (0.01)	3.11 (0.03)
	bic	0.96 (0.01)	0.35 (0.08)	0.35	0	1.09 (0.02)	0.5 (0.03)	0.31 (0.02)	3.35 (0.08)
	ebic	**0.96** (0.01)	0.35 (0.08)	0.35	0	**1.09** (0.02)	**0.5** (0.03)	**0.31** (0.02)	3.35 (0.08)
	cv	0.94 (0.01)	0.48 (0.11)	0.48	0	**1.09** (0.03)	0.51 (0.04)	**0.31** (0.02)	3.48 (0.11)
	csuv.m.0	**0.96** (0.01)	**0.27** (0.06)	**0.27**	0	**1.09** (0.02)	**0.5** (0.03)	**0.31** (0.02)	3.27 (0.06)
	csuv.s.0	0.94 (0.01)	0.42 (0.06)	0.42	0	1.1 (0.02)	0.55 (0.03)	0.33 (0.02)	3.42 (0.06)
	csuv.m.5	0.96 (0.01)	0.27 (0.06)	0.27	0	1.09 (0.02)	0.5 (0.03)	0.31 (0.02)	3.27 (0.06)
	csuv.s.5	0.94 (0.01)	0.42 (0.06)	0.42	0	1.1 (0.02)	0.55 (0.03)	0.33 (0.02)	3.42 (0.06)
	csuv.m.0.all	0.95 (0.01)	0.38 (0.07)	0.38	0	1.1 (0.02)	0.53 (0.03)	0.33 (0.02)	3.38 (0.07)
	csuv.s.0.all	0.92 (0.01)	0.56 (0.06)	0.56	0	1.11 (0.02)	0.59 (0.03)	0.35 (0.02)	3.56 (0.06)
	csuv.m.0.mcp	0.97 (0.01)	0.21 (0.06)	0.21	0	1.09 (0.02)	0.47 (0.03)	0.29 (0.02)	3.21 (0.06)
	csuv.s.0.mcp	0.97 (0.01)	0.2 (0.05)	0.2	0	1.08 (0.02)	0.46 (0.03)	0.29 (0.02)	3.2 (0.05)
setting 2	lasso	0.76 (0.01)	2.13 (0.14)	2.11	0.02	10.12 (0.21)	2.26 (0.11)	1.17 (0.05)	5.09 (0.14)
	elastic net	0.73 (0.01)	2.49 (0.14)	2.48	0.01	10.25 (0.23)	2.39 (0.11)	1.2 (0.04)	5.47 (0.14)
rho = 0.5	relaxed lasso	0.87 (0.01)	1.01 (0.12)	0.96	0.05	10.11 (0.23)	1.99 (0.12)	1.12 (0.05)	3.91 (0.12)
s = 3	mcp	0.84 (0.01)	1.29 (0.13)	1.15	0.14	10.38 (0.23)	2.3 (0.14)	1.28 (0.06)	4.01 (0.15)
p = 8	scad	0.82 (0.01)	1.53 (0.13)	1.48	0.05	10.31 (0.23)	2.26 (0.13)	1.26 (0.06)	4.43 (0.14)
	vsd	0.87 (0.01)	0.65 (0.07)	0.08	0.57	10.79 (0.27)	2.18 (0.13)	1.42 (0.08)	2.51 (0.06)
	bic	0.86 (0.01)	1.04 (0.11)	0.92	0.12	10.24 (0.22)	2.11 (0.12)	1.22 (0.06)	3.8 (0.12)
	ebic	0.86 (0.01)	1.06 (0.11)	0.94	0.12	10.22 (0.22)	2.11 (0.12)	1.22 (0.06)	3.82 (0.13)
	cv	0.76 (0.01)	2.12 (0.14)	2.09	**0.03**	10.13 (0.22)	2.28 (0.11)	1.18 (0.05)	5.06 (0.14)
	csuv.m.0	**0.91** (0.01)	**0.65** (0.09)	**0.56**	0.09	10.09 (0.22)	**1.83** (0.11)	**1.09** (0.06)	3.47 (0.08)
	csuv.s.0	0.9 (0.01)	0.72 (0.08)	0.64	0.08	**10.08** (0.21)	1.88 (0.1)	1.11 (0.05)	3.56 (0.08)
	csuv.m.5	0.91 (0.01)	0.65 (0.09)	0.56	0.09	10.09 (0.22)	1.83 (0.11)	1.09 (0.06)	3.47 (0.08)
	csuv.s.5	0.9 (0.01)	0.72 (0.08)	0.64	0.08	10.08 (0.21)	1.88 (0.1)	1.11 (0.05)	3.56 (0.08)
	csuv.m.0.all	0.91 (0.01)	0.63 (0.09)	0.57	0.06	10.09 (0.22)	1.83 (0.11)	1.09 (0.06)	3.51 (0.08)
	csuv.s.0.all	0.9 (0.01)	0.73 (0.08)	0.67	0.06	10 (0.21)	1.86 (0.1)	1.1 (0.05)	3.61 (0.07)
	csuv.m.0.mcp	0.9 (0.01)	0.74 (0.1)	0.57	0.17	10.25 (0.23)	1.99 (0.13)	1.18 (0.07)	3.4 (0.11)
	csuv.s.0.mcp	0.9 (0.01)	0.69 (0.09)	0.55	0.14	10.15 (0.22)	1.9 (0.12)	1.14 (0.06)	3.41 (0.08)
setting 3	lasso	0.73 (0.02)	2.16 (0.14)	1.86	0.3	40.51 (0.85)	4.33 (0.21)	2.28 (0.09)	4.56 (0.16)
	elastic net	0.71 (0.01)	2.39 (0.14)	2.14	0.25	40.66 (0.9)	4.37 (0.2)	2.25 (0.08)	4.89 (0.16)
rho = 0.5	relaxed lasso	0.73 (0.02)	1.78 (0.12)	1.13	0.65	41.41 (0.91)	4.52 (0.21)	2.51 (0.1)	3.48 (0.16)
s = 3	mcp	0.66 (0.02)	2.26 (0.13)	1.41	0.85	42.76 (0.98)	5.54 (0.23)	2.97 (0.1)	3.56 (0.2)
p = 8	scad	0.7 (0.02)	2.22 (0.14)	1.67	0.55	42.47 (0.92)	5.51 (0.22)	2.94 (0.1)	4.12 (0.17)
	vsd	0.54 (0.01)	2.24 (0.06)	0.01	2.23	48.66 (1.21)	5.94 (0.14)	3.53 (0.08)	0.78 (0.06)
	bic	0.69 (0.02)	1.9 (0.12)	1.03	0.87	42.24 (0.95)	4.87 (0.21)	2.72 (0.1)	3.16 (0.16)
	ebic	0.69 (0.02)	1.9 (0.12)	1.03	0.87	42.24 (0.95)	4.87 (0.21)	2.72 (0.1)	3.16 (0.16)
	cv	**0.73** (0.02)	2.16 (0.14)	1.86	**0.3**	**40.51** (0.85)	**4.33** (0.21)	**2.28** (0.09)	4.56 (0.16)
	csuv.m.0	**0.73** (0.02)	**1.45** (0.09)	**0.43**	1.02	42.38 (0.91)	4.47 (0.21)	2.69 (0.1)	2.41 (0.11)
	csuv.s.0	**0.73** (0.02)	1.53 (0.11)	0.63	0.9	42.16 (0.9)	4.46 (0.22)	2.6 (0.1)	2.73 (0.1)
	csuv.m.5	0.73 (0.02)	1.45 (0.09)	0.43	1.02	42.38 (0.91)	4.47 (0.21)	2.69 (0.1)	2.41 (0.11)
	csuv.s.5	0.73 (0.02)	1.53 (0.11)	0.63	0.9	42.16 (0.9)	4.46 (0.22)	2.6 (0.1)	2.73 (0.1)
	csuv.m.0.all	0.74 (0.02)	1.41 (0.1)	0.44	0.97	42.15 (0.89)	4.39 (0.21)	2.65 (0.1)	2.47 (0.11)
	csuv.s.0.all	0.74 (0.02)	1.51 (0.1)	0.65	0.86	41.99 (0.87)	4.4 (0.2)	2.6 (0.1)	2.79 (0.1)
	csuv.m.0.mcp	0.69 (0.02)	1.59 (0.09)	0.36	1.23	43.2 (0.96)	4.77 (0.21)	2.85 (0.1)	2.13 (0.11)
	csuv.s.0.mcp	0.7 (0.02)	1.6 (0.09)	0.45	1.15	42.74 (0.95)	4.77 (0.22)	2.8 (0.1)	2.3 (0.1)

Table A.1 Model 1: performance of CSUV and methods it compares with. Variable selection performance in terms of F-measure (f), total error (FP+FN), false positives (FP) and false negatives (FN), prediction error in terms of mse (pred.err) and estimation error in terms of l1 and l2 distance (l1.diff and l2.diff) and are shown. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

parameters	methods	f	FP+FN	FP	FN	pred.err	l1.diff	l2.diff	size
setting 1	lasso	0.43 (0.01)	15.7 (0.89)	15.7	0	1.28 (0.02)	1.76 (0.07)	0.53 (0.01)	20.7 (0.89)
	elastic net	0.35 (0.01)	21.2 (0.91)	21.2	0	1.36 (0.02)	2.22 (0.07)	0.6 (0.01)	26.2 (0.91)
rho = 0	relaxed lasso	0.88 (0.02)	2.1 (0.47)	2.05	0.05	1.14 (0.02)	0.81 (0.07)	0.35 (0.02)	7 (0.47)
s = 5	mcp	0.85 (0.02)	2.3 (0.32)	2.3	0	1.1 (0.02)	0.68 (0.04)	0.3 (0.01)	7.3 (0.32)
p = 100	scad	0.66 (0.02)	6.16 (0.4)	6.16	0	1.09 (0.02)	0.75 (0.03)	0.3 (0.01)	11.16 (0.4)
	vsd	0.99 (0)	0.09 (0.03)	0.02	0.07	1.07 (0.02)	0.44 (0.02)	0.24 (0.01)	4.95 (0.03)
	bic	0.77 (0.02)	3.6 (0.33)	3.6	0	1.09 (0.02)	0.7 (0.03)	0.3 (0.01)	8.6 (0.33)
	ebic	0.85 (0.02)	2.31 (0.31)	2.31	**0**	1.09 (0.02)	0.67 (0.04)	0.3 (0.01)	7.31 (0.31)
	cv	0.76 (0.02)	4.28 (0.46)	4.28	**0**	1.09 (0.02)	0.7 (0.03)	0.29 (0.01)	9.28 (0.46)
	csuv.m.0	**0.98** (0)	**0.2** (0.04)	**0.18**	0.02	**1.06** (0.02)	**0.47** (0.02)	**0.25** (0.01)	5.16 (0.04)
	csuv.s.0	0.93 (0.01)	0.81 (0.09)	0.79	0.02	1.1 (0.02)	0.62 (0.03)	0.32 (0.01)	5.77 (0.09)
	csuv.m.5	0.99 (0)	0.15 (0.04)	0.12	0.03	1.06 (0.02)	0.45 (0.02)	0.25 (0.01)	5.09 (0.04)
	csuv.s.5	0.86 (0.01)	1.63 (0.1)	1.62	0.01	1.14 (0.02)	0.81 (0.03)	0.38 (0.01)	6.61 (0.1)
	csuv.m.0.all	0.98 (0)	0.21 (0.04)	0.19	0.02	1.06 (0.02)	0.47 (0.02)	0.25 (0.01)	5.17 (0.04)
	csuv.s.0.all	0.92 (0.01)	0.92 (0.09)	0.9	0.02	1.11 (0.02)	0.65 (0.03)	0.33 (0.01)	5.88 (0.09)
	csuv.m.0.mcp	0.99 (0)	0.09 (0.03)	0.04	0.05	1.06 (0.02)	0.43 (0.02)	0.24 (0.01)	4.99 (0.03)
	csuv.s.0.mcp	0.97 (0)	0.29 (0.05)	0.28	0.01	1.07 (0.02)	0.49 (0.02)	0.26 (0.01)	5.27 (0.05)
setting 2	lasso	0.33 (0.01)	24.65 (1.43)	24.64	0.01	1.43 (0.03)	2.41 (0.1)	0.65 (0.02)	29.63 (1.44)
	elastic net	0.26 (0.01)	32.75 (1.44)	32.74	0.01	1.59 (0.03)	3.05 (0.1)	0.75 (0.02)	37.73 (1.44)
rho = 0	relaxed lasso	0.78 (0.02)	4.25 (0.81)	4.12	0.13	1.28 (0.03)	1.3 (0.09)	0.49 (0.02)	8.99 (0.81)
s = 5	mcp	0.75 (0.02)	4.24 (0.39)	4.23	0.01	1.13 (0.02)	0.87 (0.04)	0.36 (0.01)	9.22 (0.39)
p = 300	scad	0.5 (0.02)	12.21 (0.72)	12.2	0.01	1.14 (0.02)	1.1 (0.04)	0.37 (0.01)	17.19 (0.72)
	vsd	0.94 (0.01)	0.5 (0.09)	0	0.5	1.28 (0.05)	0.76 (0.06)	0.43 (0.03)	4.5 (0.09)
	bic	0.55 (0.02)	12.38 (1.43)	12.37	0.01	1.19 (0.02)	1.32 (0.11)	0.41 (0.02)	17.36 (1.43)
	ebic	0.74 (0.02)	5.84 (1.22)	5.83	**0.01**	1.15 (0.02)	0.98 (0.09)	0.37 (0.01)	10.82 (1.22)
	cv	0.61 (0.02)	8.92 (0.75)	8.91	**0.01**	1.13 (0.02)	1.01 (0.05)	0.36 (0.01)	13.9 (0.75)
	csuv.m.0	**0.97** (0.01)	**0.28** (0.06)	**0.13**	0.15	**1.11** (0.02)	**0.58** (0.03)	**0.31** (0.02)	4.98 (0.06)
	csuv.s.0	0.86 (0.01)	1.71 (0.12)	1.65	0.06	1.19 (0.02)	0.95 (0.04)	0.44 (0.01)	6.59 (0.13)
	csuv.m.5	0.97 (0.01)	0.3 (0.05)	0.11	0.19	1.12 (0.02)	0.6 (0.04)	0.33 (0.02)	4.92 (0.05)
	csuv.s.5	0.78 (0.01)	3.02 (0.15)	3	0.02	1.25 (0.02)	1.24 (0.04)	0.51 (0.01)	7.98 (0.15)
	csuv.m.0.all	0.97 (0.01)	0.31 (0.06)	0.15	0.16	1.12 (0.02)	0.59 (0.04)	0.32 (0.02)	4.99 (0.05)
	csuv.s.0.all	0.85 (0.01)	1.87 (0.14)	1.81	0.06	1.19 (0.02)	1 (0.04)	0.46 (0.01)	6.75 (0.14)
	csuv.m.0.mcp	0.97 (0.01)	0.24 (0.05)	0.01	0.23	1.13 (0.03)	0.58 (0.03)	0.33 (0.02)	4.78 (0.05)
	csuv.s.0.mcp	0.92 (0.01)	0.83 (0.07)	0.72	0.11	1.15 (0.02)	0.74 (0.03)	0.38 (0.01)	5.61 (0.09)
setting 3	lasso	0.46 (0.01)	24.88 (0.93)	24.88	0	1.55 (0.03)	3.23 (0.1)	0.73 (0.01)	34.88 (0.93)
	elastic net	0.42 (0.01)	29.6 (0.89)	29.6	0	1.65 (0.03)	3.74 (0.1)	0.79 (0.01)	39.6 (0.89)
rho = 0	relaxed lasso	0.8 (0.01)	5.53 (0.46)	5.49	0.04	1.36 (0.03)	1.95 (0.08)	0.58 (0.02)	15.45 (0.46)
s = 10	mcp	0.88 (0.01)	3.1 (0.31)	3.1	0	1.19 (0.02)	1.23 (0.05)	0.42 (0.01)	13.1 (0.31)
p = 100	scad	0.7 (0.01)	8.98 (0.41)	8.98	0	1.2 (0.02)	1.4 (0.05)	0.43 (0.01)	18.98 (0.41)
	vsd	0.98 (0.01)	0.4 (0.09)	0.1	0.3	1.25 (0.05)	1.06 (0.06)	0.43 (0.03)	9.8 (0.08)
	bic	0.83 (0.01)	5.28 (0.72)	5.28	0	1.22 (0.03)	1.41 (0.1)	0.44 (0.02)	15.28 (0.72)
	ebic	0.87 (0.01)	3.27 (0.33)	3.27	**0**	**1.19** (0.02)	1.24 (0.05)	**0.42** (0.01)	13.27 (0.33)
	cv	0.63 (0.02)	14.87 (1.16)	14.87	**0**	1.38 (0.03)	2.25 (0.14)	0.58 (0.02)	24.87 (1.16)
	csuv.m.0	**0.97** (0)	**0.49** (0.08)	**0.27**	0.22	1.23 (0.04)	**1.07** (0.05)	0.43 (0.02)	10.05 (0.09)
	csuv.s.0	0.87 (0.01)	3.19 (0.16)	3.16	0.03	1.31 (0.03)	1.64 (0.05)	0.54 (0.01)	13.13 (0.16)
	csuv.m.5	0.98 (0)	0.46 (0.08)	0.25	0.21	1.22 (0.04)	1.06 (0.05)	0.42 (0.02)	10.04 (0.08)
	csuv.s.5	0.83 (0)	4.29 (0.15)	4.27	0.02	1.35 (0.03)	1.86 (0.05)	0.58 (0.01)	14.25 (0.15)
	csuv.m.0.all	0.98 (0)	0.47 (0.07)	0.3	0.17	1.21 (0.03)	1.05 (0.04)	0.42 (0.02)	10.13 (0.08)
	csuv.s.0.all	0.86 (0.01)	3.47 (0.16)	3.44	0.03	1.32 (0.03)	1.7 (0.05)	0.55 (0.01)	13.41 (0.17)
	csuv.m.0.mcp	0.98 (0)	0.36 (0.07)	0.03	0.33	1.25 (0.04)	1.05 (0.05)	0.43 (0.02)	9.7 (0.07)
	csuv.s.0.mcp	0.94 (0)	1.26 (0.08)	1.19	0.07	1.24 (0.04)	1.25 (0.04)	0.47 (0.02)	11.12 (0.08)

Table A.2 Model 2: performance of CSUV and methods it compares with. Variable selection performance in terms of F-measure (f), total error (FP+FN), false positives (FP) and false negatives (FN), prediction error in terms of mse (pred.err) and estimation error in terms of l1 and l2 distance (l1.diff and l2.diff) and are shown. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

parameters	methods	f	FP+FN	FP	FN	pred.err	l1.diff	l2.diff	size
setting 4	lasso	0.38 (0.01)	36.64 (1.56)	36.6	0.04	1.88 (0.05)	4.49 (0.14)	0.94 (0.02)	46.56 (1.56)
	elastic net	0.31 (0.01)	47.38 (1.54)	47.34	0.04	2.15 (0.05)	5.62 (0.14)	1.08 (0.02)	57.3 (1.55)
rho = 0	relaxed lasso	0.66 (0.01)	11.84 (0.9)	11.61	0.23	1.7 (0.05)	3.2 (0.13)	0.81 (0.02)	21.38 (0.93)
s = 10	mcp	0.81 (0.01)	5.19 (0.35)	5.16	0.03	1.24 (0.03)	1.46 (0.06)	0.48 (0.02)	15.13 (0.35)
p = 300	scad	0.57 (0.01)	15.88 (0.53)	15.86	0.02	1.23 (0.03)	1.74 (0.06)	0.48 (0.02)	25.84 (0.53)
	vsd	0.88 (0.02)	2.08 (0.28)	0.03	2.05	2.36 (0.24)	2.33 (0.22)	0.9 (0.08)	7.98 (0.29)
	bic	0.6 (0.02)	19.21 (2)	19.19	0.02	1.39 (0.05)	2.44 (0.2)	0.58 (0.03)	29.17 (2.01)
	ebic	0.78 (0.02)	8.05 (1.42)	8.02	**0.03**	**1.28** (0.03)	**1.75** (0.15)	**0.51** (0.02)	17.99 (1.42)
	cv	0.41 (0.01)	33.78 (1.67)	33.74	0.04	1.8 (0.05)	4.18 (0.16)	0.88 (0.02)	43.7 (1.67)
	csuv.m.0	**0.91** (0.01)	**1.51** (0.15)	**0.1**	1.41	1.77 (0.08)	1.84 (0.1)	0.8 (0.04)	8.69 (0.13)
	csuv.s.0	0.76 (0.01)	6.53 (0.25)	6.29	0.24	1.5 (0.04)	2.35 (0.07)	0.7 (0.02)	16.05 (0.27)
	csuv.m.5	0.91 (0.01)	1.59 (0.15)	0.08	1.51	1.82 (0.08)	1.89 (0.11)	0.82 (0.04)	8.57 (0.14)
	csuv.s.5	0.72 (0.01)	7.75 (0.23)	7.51	0.24	1.55 (0.04)	2.57 (0.07)	0.73 (0.02)	17.27 (0.26)
	csuv.m.0.all	0.92 (0.01)	1.48 (0.15)	0.14	1.34	1.75 (0.08)	1.81 (0.11)	0.79 (0.04)	8.8 (0.12)
	csuv.s.0.all	0.74 (0.01)	7.26 (0.27)	7.03	0.23	1.53 (0.04)	2.47 (0.07)	0.72 (0.02)	16.8 (0.29)
	csuv.m.0.mcp	0.78 (0.01)	3.37 (0.18)	0.01	3.36	2.86 (0.13)	3.21 (0.14)	1.31 (0.04)	6.65 (0.18)
	csuv.s.0.mcp	0.91 (0.01)	1.65 (0.14)	0.45	1.2	1.73 (0.09)	1.84 (0.11)	0.77 (0.04)	9.25 (0.14)
setting 5	lasso	0.39 (0.01)	16.4 (0.67)	16.19	0.21	1.28 (0.02)	3.51 (0.11)	1.04 (0.03)	20.98 (0.68)
	elastic net	0.33 (0.01)	21.14 (0.69)	20.99	0.15	1.31 (0.02)	4.21 (0.11)	1.15 (0.02)	25.84 (0.69)
rho = 0.9	relaxed lasso	0.49 (0.01)	10.58 (0.65)	10	0.58	1.31 (0.02)	3.31 (0.13)	1.07 (0.03)	14.42 (0.7)
s = 5	mcp	0.67 (0.02)	4.26 (0.31)	3.17	1.09	1.29 (0.03)	2.48 (0.17)	1.09 (0.06)	7.08 (0.24)
p = 100	scad	0.65 (0.02)	4.91 (0.28)	4.06	0.85	1.24 (0.03)	2.01 (0.14)	0.91 (0.06)	8.21 (0.22)
	vsd	0.76 (0.02)	2.1 (0.16)	0.5	1.6	1.38 (0.03)	1.93 (0.13)	1.01 (0.05)	3.9 (0.08)
	bic	0.69 (0.02)	4.15 (0.32)	3.26	0.89	1.26 (0.03)	2.22 (0.17)	0.96 (0.06)	7.37 (0.24)
	ebic	0.71 (0.02)	3.67 (0.27)	2.77	0.9	**1.24** (0.03)	2.07 (0.15)	0.94 (0.06)	6.87 (0.2)
	cv	0.47 (0.02)	13.01 (0.75)	12.64	**0.37**	1.27 (0.02)	3.14 (0.14)	1.01 (0.04)	17.27 (0.79)
	csuv.m.0	**0.77** (0.02)	**2.25** (0.17)	**0.97**	1.28	1.33 (0.03)	**1.9** (0.11)	**0.93** (0.05)	4.69 (0.11)
	csuv.s.0	0.68 (0.01)	4.21 (0.18)	3.5	0.71	1.26 (0.02)	2.31 (0.1)	0.95 (0.04)	7.79 (0.18)
	csuv.m.5	0.77 (0.02)	2.16 (0.16)	0.8	1.36	1.35 (0.03)	1.88 (0.11)	0.95 (0.04)	4.44 (0.11)
	csuv.s.5	0.66 (0.01)	4.45 (0.17)	3.77	0.68	1.26 (0.02)	2.37 (0.1)	0.95 (0.03)	8.09 (0.17)
	csuv.m.0.all	0.76 (0.02)	2.43 (0.17)	1.23	1.2	1.31 (0.03)	1.96 (0.11)	0.94 (0.04)	5.03 (0.14)
	csuv.s.0.all	0.65 (0.01)	4.94 (0.21)	4.28	0.66	1.26 (0.03)	2.48 (0.1)	0.97 (0.03)	8.62 (0.22)
	csuv.m.0.mcp	0.69 (0.02)	2.53 (0.14)	0.36	2.17	1.87 (0.07)	2.37 (0.12)	1.24 (0.05)	3.19 (0.09)
	csuv.s.0.mcp	0.72 (0.02)	2.88 (0.18)	1.55	1.33	1.34 (0.03)	2.24 (0.14)	1.03 (0.05)	5.22 (0.08)
setting 6	lasso	0.3 (0.01)	24.79 (1.08)	24.6	0.19	1.42 (0.03)	3.81 (0.15)	1.04 (0.03)	29.41 (1.09)
	elastic net	0.23 (0.01)	34.73 (1.22)	34.6	0.13	1.51 (0.03)	4.91 (0.14)	1.21 (0.02)	39.47 (1.21)
rho = 0.9	relaxed lasso	0.51 (0.02)	10.84 (0.88)	10.37	0.47	1.35 (0.03)	3.13 (0.15)	0.99 (0.03)	14.9 (0.9)
s = 5	mcp	0.59 (0.02)	5.52 (0.33)	4.24	1.28	1.33 (0.03)	2.66 (0.16)	1.16 (0.06)	7.96 (0.24)
p = 300	scad	0.57 (0.02)	7.54 (0.42)	6.94	0.6	1.21 (0.02)	1.78 (0.12)	0.79 (0.05)	11.34 (0.36)
	vsd	0.75 (0.02)	2.47 (0.2)	0.58	1.89	1.87 (0.1)	2.28 (0.16)	1.12 (0.07)	3.69 (0.19)
	bic	0.59 (0.02)	8.1 (1.1)	7.33	0.77	1.27 (0.03)	2.3 (0.19)	0.92 (0.05)	11.56 (1.12)
	ebic	0.63 (0.02)	5.69 (0.73)	4.65	1.04	1.29 (0.03)	2.32 (0.17)	1 (0.06)	8.61 (0.72)
	cv	0.44 (0.02)	16.75 (1.32)	16.32	0.43	1.33 (0.03)	2.89 (0.19)	0.93 (0.04)	20.89 (1.34)
	csuv.m.0	**0.79** (0.02)	**2.01** (0.16)	**0.91**	1.1	1.4 (0.04)	**1.81** (0.12)	0.92 (0.05)	4.81 (0.1)
	csuv.s.0	0.64 (0.01)	5.53 (0.24)	5.19	**0.34**	**1.24** (0.02)	2.31 (0.1)	**0.87** (0.03)	9.85 (0.21)
	csuv.m.5	0.8 (0.02)	1.87 (0.15)	0.76	1.11	1.43 (0.04)	1.75 (0.11)	0.9 (0.05)	4.65 (0.1)
	csuv.s.5	0.61 (0.01)	6.14 (0.19)	5.84	0.3	1.25 (0.02)	2.43 (0.09)	0.88 (0.03)	10.54 (0.16)
	csuv.m.0.all	0.8 (0.02)	2 (0.16)	1	1	1.38 (0.04)	1.74 (0.11)	0.87 (0.05)	5 (0.11)
	csuv.s.0.all	0.61 (0.01)	6.31 (0.27)	5.98	0.33	1.26 (0.03)	2.49 (0.11)	0.89 (0.03)	10.65 (0.25)
	csuv.m.0.mcp	0.68 (0.02)	2.56 (0.15)	0.39	2.17	2.11 (0.07)	2.46 (0.12)	1.25 (0.05)	3.22 (0.11)
	csuv.s.0.mcp	0.73 (0.01)	2.98 (0.17)	2.05	0.93	1.3 (0.03)	2.13 (0.13)	0.97 (0.05)	6.12 (0.07)

Table A.3 Model 2: performance of CSUV and methods it compares with (continue). Variable selection performance in terms of F-measure (f), total error (FP+FN), false positives (FP) and false negatives (FN), prediction error in terms of mse (pred.err) and estimation error in terms of l1 and l2 distance (l1.diff and l2.diff) and are shown. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

parameters	methods	f	FP+FN	FP	FN	pred.err	l1.diff	l2.diff	size
setting 7	lasso	0.47 (0.01)	22.4 (0.7)	21.74	0.66	1.49 (0.03)	6.3 (0.17)	1.49 (0.03)	31.08 (0.73)
	elastic net	0.42 (0.01)	27.11 (0.73)	26.68	0.43	1.54 (0.03)	7.11 (0.17)	1.57 (0.03)	36.25 (0.73)
rho = 0.9	relaxed lasso	0.54 (0.01)	17.19 (0.83)	16.23	0.96	1.53 (0.03)	6.31 (0.22)	1.55 (0.03)	25.27 (0.91)
s = 10	mcp	0.64 (0.01)	7.8 (0.32)	4.56	3.24	1.66 (0.05)	6.15 (0.23)	1.98 (0.06)	11.32 (0.27)
p = 100	scad	0.66 (0.01)	8.03 (0.35)	5.54	2.49	1.48 (0.04)	4.97 (0.21)	1.71 (0.06)	13.05 (0.3)
	vsd	0.7 (0.02)	5.13 (0.24)	1.36	3.77	1.86 (0.08)	5.3 (0.22)	1.89 (0.06)	7.59 (0.15)
	bic	0.67 (0.01)	7.44 (0.33)	4.84	2.6	1.55 (0.04)	5.35 (0.22)	1.78 (0.05)	12.24 (0.29)
	ebic	0.68 (0.01)	6.73 (0.29)	3.94	2.79	1.55 (0.04)	5.21 (0.21)	1.77 (0.06)	11.15 (0.23)
	cv	0.47 (0.01)	22.4 (0.7)	21.74	**0.66**	1.49 (0.03)	6.3 (0.17)	1.49 (0.03)	31.08 (0.73)
	csuv.m.0	**0.76** (0.01)	**4.57** (0.2)	**1.84**	2.73	1.66 (0.04)	**4.35** (0.14)	1.53 (0.04)	9.11 (0.15)
	csuv.s.0	0.67 (0.01)	8.65 (0.24)	7.19	1.46	**1.45** (0.03)	4.87 (0.13)	**1.44** (0.03)	15.73 (0.23)
	csuv.m.5	0.76 (0.01)	4.54 (0.19)	1.74	2.8	1.69 (0.04)	4.37 (0.14)	1.54 (0.04)	8.94 (0.15)
	csuv.s.5	0.66 (0.01)	9 (0.23)	7.54	1.46	1.45 (0.03)	4.96 (0.12)	1.45 (0.03)	16.08 (0.22)
	csuv.m.0.all	0.76 (0.01)	4.76 (0.22)	2.31	2.45	1.6 (0.04)	4.25 (0.15)	1.46 (0.04)	9.86 (0.16)
	csuv.s.0.all	0.64 (0.01)	9.97 (0.25)	8.62	1.35	1.45 (0.03)	5.14 (0.11)	1.46 (0.03)	17.27 (0.27)
	csuv.m.0.mcp	0.59 (0.01)	5.95 (0.18)	0.47	5.48	3.22 (0.11)	6.16 (0.18)	2.19 (0.05)	4.99 (0.12)
	csuv.s.0.mcp	0.72 (0.01)	5.48 (0.21)	2.39	3.09	1.74 (0.04)	5.04 (0.17)	1.74 (0.04)	9.3 (0.09)
setting 8	lasso	0.39 (0.01)	32.18 (1.08)	31.63	0.55	1.72 (0.04)	6.9 (0.19)	1.5 (0.03)	41.08 (1.06)
	elastic net	0.32 (0.01)	41.56 (0.99)	41.13	0.43	1.83 (0.04)	8.36 (0.16)	1.68 (0.03)	50.7 (0.97)
rho = 0.9	relaxed lasso	0.51 (0.01)	18.39 (0.75)	17.43	0.96	1.64 (0.04)	6.25 (0.18)	1.5 (0.03)	26.47 (0.79)
s = 10	mcp	0.53 (0.02)	11.72 (0.57)	7.88	3.84	2.09 (0.07)	7.45 (0.36)	2.2 (0.08)	14.04 (0.3)
p = 300	scad	0.54 (0.01)	14.05 (0.51)	11.82	2.23	1.71 (0.06)	5.27 (0.26)	1.7 (0.07)	19.59 (0.35)
	vsd	0.6 (0.02)	6.71 (0.33)	1.82	4.89	3.24 (0.23)	6.5 (0.31)	2.21 (0.09)	6.93 (0.25)
	bic	0.55 (0.02)	13.84 (1.12)	11.25	2.59	1.79 (0.06)	5.95 (0.31)	1.81 (0.07)	18.66 (1.15)
	ebic	0.57 (0.02)	10.93 (0.5)	7.81	3.12	1.86 (0.05)	6.22 (0.3)	1.94 (0.07)	14.69 (0.36)
	cv	0.39 (0.01)	32.18 (1.08)	31.63	**0.55**	1.72 (0.04)	6.9 (0.19)	**1.5** (0.03)	41.08 (1.06)
	csuv.m.0	**0.69** (0.01)	**5.8** (0.23)	**2.27**	3.53	2.37 (0.08)	**5.28** (0.19)	1.82 (0.05)	8.74 (0.16)
	csuv.s.0	0.55 (0.01)	14.1 (0.32)	12.73	1.37	**1.65** (0.04)	5.87 (0.16)	1.52 (0.03)	21.36 (0.34)
	csuv.m.5	0.69 (0.01)	5.68 (0.22)	1.96	3.72	2.57 (0.1)	5.25 (0.17)	1.83 (0.05)	8.24 (0.16)
	csuv.s.5	0.55 (0.01)	14.24 (0.33)	12.87	1.37	1.64 (0.04)	5.92 (0.16)	1.54 (0.04)	21.5 (0.34)
	csuv.m.0.all	0.69 (0.01)	5.99 (0.24)	2.68	3.31	2.26 (0.08)	5.17 (0.18)	1.76 (0.05)	9.37 (0.17)
	csuv.s.0.all	0.53 (0.01)	15.93 (0.35)	14.69	1.24	1.64 (0.04)	6.12 (0.16)	1.53 (0.03)	23.45 (0.36)
	csuv.m.0.mcp	0.38 (0.01)	7.95 (0.16)	0.32	7.63	7.18 (0.21)	7.43 (0.16)	2.53 (0.04)	2.69 (0.13)
	csuv.s.0.mcp	0.59 (0.01)	8.07 (0.26)	3.9	4.17	2.53 (0.08)	6.56 (0.21)	2.08 (0.05)	9.73 (0.11)

Table A.4 Model 2: performance of CSUV and methods it compares with (continue). Variable selection performance in terms of F-measure (f), total error (FP+FN), false positives (FP) and false negatives (FN), prediction error in terms of mse (pred.err) and estimation error in terms of l1 and l2 distance (l1.diff and l2.diff) and are shown. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

parameters	methods	f	FP+FN	FP	FN	pred.err	l1.diff	l2.diff	size
setting 1	lasso	0.38 (0.01)	17.85 (0.75)	17.84	0.01	1.31 (0.02)	2.35 (0.07)	0.68 (0.01)	22.83 (0.75)
	elastic net	0.32 (0.01)	23.17 (0.81)	23.17	0	1.38 (0.02)	2.91 (0.08)	0.77 (0.01)	28.17 (0.81)
block.cor = 0.5	relaxed lasso	0.62 (0.02)	7.16 (0.63)	6.79	0.37	1.3 (0.03)	1.76 (0.07)	0.63 (0.02)	11.42 (0.69)
s = 5	mcp	0.85 (0.01)	2.15 (0.26)	2.07	0.08	1.12 (0.03)	0.77 (0.07)	0.34 (0.02)	6.99 (0.26)
p = 100	scad	0.63 (0.01)	6.61 (0.37)	6.58	0.03	1.13 (0.03)	0.9 (0.06)	0.36 (0.02)	11.55 (0.38)
	vsd	0.92 (0.01)	0.7 (0.08)	0.05	0.65	1.21 (0.03)	0.87 (0.06)	0.5 (0.03)	4.4 (0.08)
	bic	0.81 (0.01)	2.79 (0.3)	2.73	0.06	1.13 (0.03)	0.82 (0.08)	0.35 (0.02)	7.67 (0.31)
	ebic	0.85 (0.01)	2.15 (0.26)	2.07	0.08	**1.12** (0.03)	**0.77** (0.07)	**0.34** (0.02)	6.99 (0.26)
	cv	0.69 (0.02)	5.62 (0.52)	5.56	**0.06**	1.13 (0.03)	0.9 (0.07)	0.36 (0.02)	10.5 (0.53)
	csuv.m.0	**0.92** (0.01)	**0.76** (0.08)	**0.23**	0.53	1.2 (0.03)	0.87 (0.06)	0.48 (0.03)	4.7 (0.09)
	csuv.s.0	0.85 (0.01)	1.72 (0.13)	1.42	0.3	1.21 (0.03)	1.06 (0.05)	0.51 (0.02)	6.12 (0.15)
	csuv.m.5	0.92 (0.01)	0.75 (0.08)	0.17	0.58	1.2 (0.03)	0.88 (0.05)	0.5 (0.03)	4.59 (0.08)
	csuv.s.5	0.81 (0.01)	2.35 (0.12)	2.08	0.27	1.22 (0.03)	1.17 (0.05)	0.53 (0.02)	6.81 (0.16)
	csuv.m.0.all	0.91 (0.01)	0.83 (0.09)	0.31	0.52	1.2 (0.03)	0.89 (0.06)	0.49 (0.03)	4.79 (0.1)
	csuv.s.0.all	0.83 (0.01)	1.98 (0.14)	1.7	0.28	1.21 (0.03)	1.1 (0.05)	0.51 (0.02)	6.42 (0.16)
	csuv.m.0.mcp	0.92 (0.01)	0.72 (0.08)	0.07	0.65	1.21 (0.03)	0.89 (0.05)	0.51 (0.03)	4.42 (0.07)
	csuv.s.0.mcp	0.9 (0.01)	0.99 (0.08)	0.5	0.49	1.2 (0.03)	0.92 (0.05)	0.5 (0.03)	5.01 (0.12)
setting 2	lasso	0.29 (0.01)	26.37 (1.1)	26.32	0.05	1.49 (0.03)	3.19 (0.1)	0.85 (0.02)	31.27 (1.1)
	elastic net	0.23 (0.01)	35.16 (1.2)	35.1	0.06	1.62 (0.03)	3.94 (0.11)	0.98 (0.02)	40.04 (1.2)
block.cor = 0.5	relaxed lasso	0.49 (0.01)	12.49 (0.9)	12.23	0.26	1.48 (0.03)	2.68 (0.12)	0.8 (0.03)	16.97 (0.94)
s = 5	mcp	0.79 (0.01)	2.99 (0.25)	2.9	0.09	1.15 (0.02)	0.93 (0.06)	0.41 (0.02)	7.81 (0.24)
p = 300	scad	0.54 (0.01)	9.86 (0.51)	9.83	0.03	1.15 (0.02)	1.13 (0.05)	0.41 (0.02)	14.8 (0.5)
	vsd	0.9 (0.01)	1.06 (0.14)	0.05	1.01	1.56 (0.09)	1.26 (0.12)	0.67 (0.06)	4.04 (0.14)
	bic	0.65 (0.02)	8.15 (1.17)	8.08	0.07	1.18 (0.03)	1.3 (0.12)	0.45 (0.02)	13.01 (1.18)
	ebic	0.79 (0.02)	3.75 (0.85)	3.66	0.09	**1.16** (0.02)	1.01 (0.1)	**0.42** (0.02)	8.57 (0.85)
	cv	0.5 (0.02)	14.67 (1.34)	14.61	**0.06**	1.28 (0.03)	1.85 (0.15)	0.57 (0.03)	19.55 (1.34)
	csuv.m.0	**0.92** (0.01)	**0.72** (0.1)	**0.18**	0.54	1.3 (0.05)	**0.96** (0.08)	0.52 (0.04)	4.64 (0.07)
	csuv.s.0	0.73 (0.01)	3.8 (0.21)	3.67	0.13	1.28 (0.03)	1.55 (0.07)	0.6 (0.02)	8.54 (0.2)
	csuv.m.5	0.92 (0.01)	0.71 (0.1)	0.16	0.55	1.3 (0.05)	0.97 (0.08)	0.53 (0.04)	4.61 (0.08)
	csuv.s.5	0.67 (0.01)	5.08 (0.17)	4.96	0.12	1.32 (0.02)	1.8 (0.06)	0.65 (0.02)	9.84 (0.18)
	csuv.m.0.all	0.92 (0.01)	0.72 (0.11)	0.18	0.54	1.29 (0.05)	0.96 (0.08)	0.52 (0.04)	4.64 (0.07)
	csuv.s.0.all	0.72 (0.01)	4.13 (0.23)	4.01	0.12	1.28 (0.02)	1.6 (0.07)	0.6 (0.02)	8.89 (0.22)
	csuv.m.0.mcp	0.9 (0.01)	0.84 (0.09)	0.04	0.8	1.4 (0.05)	1.09 (0.07)	0.62 (0.04)	4.24 (0.09)
	csuv.s.0.mcp	0.87 (0.01)	1.38 (0.1)	1.12	0.26	1.23 (0.04)	1.04 (0.07)	0.5 (0.03)	5.86 (0.08)
setting 3	lasso	0.48 (0.01)	22.55 (0.82)	22.53	0.02	1.52 (0.03)	3.77 (0.1)	0.88 (0.01)	32.51 (0.83)
	elastic net	0.44 (0.01)	26.43 (0.83)	26.41	0.02	1.58 (0.03)	4.22 (0.1)	0.94 (0.01)	36.39 (0.83)
block.cor = 0.5	relaxed lasso	0.66 (0.01)	11.12 (0.69)	10.68	0.44	1.49 (0.02)	3.17 (0.1)	0.87 (0.02)	20.24 (0.75)
s = 10	mcp	0.89 (0.01)	2.62 (0.21)	2.44	0.18	1.22 (0.02)	1.5 (0.07)	0.54 (0.02)	12.26 (0.23)
p = 100	scad	0.76 (0.01)	6.64 (0.3)	6.5	0.14	1.21 (0.02)	1.55 (0.06)	0.52 (0.02)	16.36 (0.3)
	vsd	0.94 (0.01)	1.17 (0.11)	0.15	1.02	1.39 (0.03)	1.77 (0.08)	0.73 (0.03)	9.13 (0.1)
	bic	0.87 (0.01)	3.28 (0.27)	3.12	0.16	1.22 (0.02)	1.52 (0.07)	0.54 (0.02)	12.96 (0.27)
	ebic	0.89 (0.01)	2.63 (0.21)	2.45	0.18	**1.22** (0.02)	**1.5** (0.07)	**0.54** (0.02)	12.27 (0.23)
	cv	0.5 (0.01)	21.91 (0.89)	21.86	**0.05**	1.52 (0.03)	3.72 (0.11)	0.88 (0.02)	31.81 (0.9)
	csuv.m.0	**0.93** (0.01)	**1.39** (0.11)	**0.54**	0.85	1.36 (0.03)	1.8 (0.07)	0.73 (0.03)	9.69 (0.11)
	csuv.s.0	0.82 (0.01)	4.47 (0.18)	4.11	0.36	1.36 (0.03)	2.25 (0.07)	0.73 (0.02)	13.75 (0.19)
	csuv.m.5	0.93 (0.01)	1.42 (0.12)	0.47	0.95	1.39 (0.03)	1.86 (0.07)	0.76 (0.03)	9.52 (0.1)
	csuv.s.5	0.8 (0.01)	4.86 (0.15)	4.54	0.32	1.37 (0.03)	2.29 (0.07)	0.73 (0.02)	14.22 (0.16)
	csuv.m.0.all	0.93 (0.01)	1.46 (0.12)	0.66	0.8	1.35 (0.03)	1.81 (0.07)	0.73 (0.03)	9.86 (0.11)
	csuv.s.0.all	0.8 (0.01)	5.02 (0.19)	4.69	0.33	1.37 (0.03)	2.32 (0.07)	0.74 (0.02)	14.36 (0.2)
	csuv.m.0.mcp	0.89 (0.01)	1.87 (0.12)	0.05	1.82	1.63 (0.04)	2.34 (0.09)	0.97 (0.03)	8.23 (0.11)
	csuv.s.0.mcp	0.93 (0.01)	1.43 (0.1)	0.59	0.84	1.35 (0.03)	1.8 (0.07)	0.73 (0.03)	9.75 (0.1)

Table A.5 Model 3: performance of CSUV and methods it compares with. Variable selection performance in terms of F-measure (f), total error (FP+FN), false positives (FP) and false negatives (FN), prediction error in terms of mse (pred.err) and estimation error in terms of l1 and l2 distance (l1.diff and l2.diff) and are shown. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

parameters	methods	f	FP+FN	FP	FN	pred.err	l1.diff	l2.diff	size
setting 4	lasso	0.37 (0.01)	35.15 (1.15)	35.03	0.12	1.77 (0.04)	5.29 (0.13)	1.11 (0.02)	44.91 (1.15)
	elastic net	0.31 (0.01)	45.8 (1.39)	45.65	0.15	1.97 (0.05)	6.59 (0.17)	1.27 (0.02)	55.5 (1.39)
block.cor = 0.5	relaxed lasso	0.51 (0.01)	20.03 (0.83)	19.75	0.28	1.73 (0.04)	4.62 (0.15)	1.04 (0.03)	29.47 (0.86)
s = 10	mcp	0.84 (0.01)	3.71 (0.27)	3.31	0.4	1.39 (0.06)	1.96 (0.14)	0.67 (0.04)	12.91 (0.21)
p = 300	scad	0.68 (0.01)	9.88 (0.44)	9.75	0.13	1.3 (0.04)	1.96 (0.11)	0.62 (0.03)	19.62 (0.41)
	vsd	0.91 (0.01)	1.83 (0.2)	0.21	1.62	1.88 (0.17)	2.31 (0.16)	0.92 (0.06)	8.59 (0.18)
	bic	0.73 (0.02)	10.55 (1.62)	10.37	0.18	1.37 (0.04)	2.35 (0.19)	0.68 (0.04)	20.19 (1.62)
	ebic	**0.85** (0.01)	4.28 (0.81)	3.97	0.31	**1.33** (0.04)	**1.9** (0.13)	**0.64** (0.03)	13.66 (0.81)
	cv	0.37 (0.01)	35.15 (1.15)	35.03	**0.12**	1.77 (0.04)	5.29 (0.13)	1.11 (0.02)	44.91 (1.15)
	csuv.m.0	**0.85** (0.01)	**2.77** (0.17)	**0.88**	1.89	1.92 (0.07)	2.73 (0.12)	1.08 (0.04)	8.99 (0.13)
	csuv.s.0	0.61 (0.01)	12.58 (0.35)	12.06	0.52	1.64 (0.04)	3.82 (0.11)	1 (0.02)	21.54 (0.37)
	csuv.m.5	0.85 (0.01)	2.78 (0.17)	0.73	2.05	2.01 (0.08)	2.79 (0.13)	1.11 (0.04)	8.68 (0.13)
	csuv.s.5	0.6 (0.01)	13 (0.35)	12.45	0.55	1.66 (0.04)	3.89 (0.12)	1 (0.03)	21.9 (0.37)
	csuv.m.0.all	0.85 (0.01)	2.85 (0.18)	1.12	1.73	1.87 (0.07)	2.65 (0.12)	1.04 (0.04)	9.39 (0.14)
	csuv.s.0.all	0.58 (0.01)	14.46 (0.41)	13.95	0.51	1.68 (0.04)	4.11 (0.12)	1.02 (0.03)	23.44 (0.44)
	csuv.m.0.mcp	0.6 (0.02)	5.53 (0.16)	0.04	5.49	4.62 (0.2)	5.23 (0.13)	1.9 (0.04)	4.55 (0.16)
	csuv.s.0.mcp	0.82 (0.01)	3.47 (0.21)	1.37	2.1	2.07 (0.08)	3.14 (0.15)	1.18 (0.04)	9.27 (0.07)
setting 5	lasso	0.38 (0.01)	14.34 (0.77)	13.22	1.12	1.27 (0.02)	4.03 (0.14)	1.28 (0.02)	17.1 (0.85)
	elastic net	0.31 (0.01)	19.43 (0.84)	18.45	0.98	1.3 (0.02)	4.77 (0.15)	1.38 (0.02)	22.47 (0.91)
block.cor = 0.9	relaxed lasso	0.48 (0.01)	8.15 (0.5)	6.46	1.69	1.26 (0.02)	3.42 (0.11)	1.26 (0.03)	9.77 (0.6)
s = 5	mcp	0.58 (0.02)	4.88 (0.22)	3.03	1.85	1.28 (0.02)	2.99 (0.15)	1.34 (0.05)	6.18 (0.26)
p = 100	scad	0.6 (0.02)	4.7 (0.27)	3.04	1.66	1.25 (0.02)	2.7 (0.15)	1.26 (0.05)	6.38 (0.32)
	vsd	0.65 (0.02)	2.86 (0.14)	0.46	2.4	1.25 (0.03)	2.41 (0.13)	1.25 (0.04)	3.06 (0.03)
	bic	0.61 (0.02)	4.35 (0.21)	2.7	1.65	1.26 (0.03)	2.74 (0.15)	1.26 (0.05)	6.05 (0.29)
	ebic	0.62 (0.02)	4.13 (0.2)	2.39	1.74	1.26 (0.03)	2.66 (0.14)	1.24 (0.05)	5.65 (0.28)
	cv	0.47 (0.02)	10.44 (0.76)	9.07	**1.37**	1.26 (0.02)	3.53 (0.16)	1.28 (0.03)	12.7 (0.86)
	csuv.m.0	**0.66** (0.01)	**2.99** (0.12)	**0.85**	2.14	1.25 (0.03)	**2.41** (0.09)	**1.19** (0.03)	3.71 (0.09)
	csuv.s.0	0.58 (0.01)	4.72 (0.16)	2.89	1.83	**1.22** (0.02)	2.95 (0.09)	1.22 (0.03)	6.06 (0.14)
	csuv.m.5	0.66 (0.01)	2.93 (0.12)	0.74	2.19	1.25 (0.03)	2.37 (0.09)	1.19 (0.03)	3.55 (0.09)
	csuv.s.5	0.57 (0.01)	4.88 (0.14)	3.03	1.85	1.23 (0.02)	2.98 (0.09)	1.23 (0.03)	6.18 (0.13)
	csuv.m.0.all	0.63 (0.01)	3.42 (0.16)	1.28	2.14	1.25 (0.03)	2.6 (0.1)	1.22 (0.03)	4.14 (0.11)
	csuv.s.0.all	0.53 (0.01)	5.71 (0.17)	3.91	1.8	1.23 (0.02)	3.16 (0.09)	1.25 (0.03)	7.11 (0.16)
	csuv.m.0.mcp	0.53 (0.02)	3.51 (0.12)	0.18	3.33	2.42 (0.1)	3.1 (0.11)	1.56 (0.04)	1.85 (0.09)
	csuv.s.0.mcp	0.61 (0.02)	3.45 (0.18)	0.99	2.46	1.39 (0.04)	2.91 (0.15)	1.39 (0.05)	3.53 (0.08)
setting 6	lasso	0.28 (0.01)	21.2 (1.05)	19.8	1.4	1.4 (0.02)	5.37 (0.15)	1.6 (0.02)	23.4 (1.1)
	elastic net	0.21 (0.01)	30.54 (1.35)	29.32	1.22	1.41 (0.02)	6.18 (0.2)	1.66 (0.02)	33.1 (1.41)
block.cor = 0.9	relaxed lasso	0.34 (0.01)	14.1 (0.77)	12.28	1.82	1.41 (0.03)	5.18 (0.17)	1.65 (0.03)	15.46 (0.83)
s = 5	mcp	0.42 (0.02)	7.21 (0.26)	4.52	2.69	1.45 (0.03)	4.8 (0.16)	1.85 (0.04)	6.83 (0.25)
p = 300	scad	0.4 (0.02)	8.83 (0.35)	6.53	2.3	1.42 (0.03)	4.42 (0.19)	1.74 (0.05)	9.23 (0.32)
	vsd	0.45 (0.02)	4.8 (0.14)	1.14	3.66	2 (0.06)	4.4 (0.11)	1.94 (0.03)	2.48 (0.12)
	bic	0.42 (0.02)	9.21 (0.91)	6.9	2.31	1.44 (0.03)	4.76 (0.21)	1.77 (0.05)	9.59 (0.95)
	ebic	0.44 (0.02)	6.91 (0.27)	4.39	2.52	1.43 (0.03)	4.59 (0.17)	1.79 (0.05)	6.87 (0.26)
	cv	0.29 (0.01)	20.18 (1.01)	18.7	**1.48**	1.39 (0.02)	5.31 (0.15)	**1.62** (0.02)	22.22 (1.06)
	csuv.m.0	**0.51** (0.02)	**3.97** (0.12)	**0.81**	3.16	2.06 (0.06)	**3.78** (0.1)	1.72 (0.03)	2.65 (0.12)
	csuv.s.0	0.44 (0.01)	7.26 (0.19)	5.08	2.18	**1.38** (0.03)	4.45 (0.11)	1.65 (0.03)	7.9 (0.13)
	csuv.m.5	0.51 (0.02)	3.86 (0.12)	0.65	3.21	2.1 (0.06)	3.71 (0.1)	1.71 (0.03)	2.44 (0.11)
	csuv.s.5	0.44 (0.01)	7.46 (0.19)	5.32	2.14	1.38 (0.03)	4.45 (0.1)	1.64 (0.03)	8.18 (0.13)
	csuv.m.0.all	0.51 (0.02)	4.02 (0.13)	1.02	3	1.94 (0.06)	3.8 (0.1)	1.7 (0.03)	3.02 (0.13)
	csuv.s.0.all	0.42 (0.01)	8.34 (0.2)	6.34	2	1.37 (0.03)	4.51 (0.1)	1.62 (0.03)	9.34 (0.17)
	csuv.m.0.mcp	0.42 (0.01)	4.26 (0.09)	0.14	4.12	2.89 (0.06)	3.93 (0.07)	1.84 (0.02)	1.02 (0.09)
	csuv.s.0.mcp	0.47 (0.02)	5.21 (0.17)	2.35	2.86	1.65 (0.05)	4.39 (0.11)	1.78 (0.03)	4.49 (0.08)

Table A.6 Model 3: performance of CSUV and methods it compares with (continue). Variable selection performance in terms of F-measure (f), total error (FP+FN), false positives (FP) and false negatives (FN), prediction error in terms of mse (pred.err) and estimation error in terms of l1 and l2 distance (l1.diff and l2.diff) and are shown. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

parameters	methods	f	FP+FN	FP	FN	pred.err	l1.diff	l2.diff	size
setting 7	lasso	0.5 (0.01)	18.54 (0.66)	17.19	1.35	1.42 (0.02)	6.55 (0.16)	1.65 (0.03)	25.84 (0.69)
	elastic net	0.44 (0.01)	22.85 (0.63)	21.67	1.18	1.43 (0.02)	7.18 (0.15)	1.7 (0.02)	30.49 (0.67)
block.cor = 0.9	relaxed lasso	0.56 (0.01)	13.74 (0.64)	11.91	1.83	1.42 (0.03)	6.37 (0.18)	1.68 (0.03)	20.08 (0.72)
s = 10	mcp	0.53 (0.01)	8.51 (0.29)	3.35	5.16	1.73 (0.04)	7.19 (0.25)	2.34 (0.06)	8.19 (0.13)
p = 100	scad	0.59 (0.01)	7.76 (0.28)	3.23	4.53	1.61 (0.04)	6.23 (0.21)	2.17 (0.06)	8.7 (0.16)
	vsd	0.6 (0.01)	6.52 (0.23)	1.53	4.99	1.8 (0.06)	6.56 (0.22)	2.3 (0.06)	6.54 (0.1)
	bic	0.6 (0.01)	7.5 (0.28)	3.04	4.46	1.59 (0.04)	6.11 (0.21)	2.11 (0.06)	8.58 (0.21)
	ebic	0.6 (0.01)	7.27 (0.27)	2.77	4.5	1.6 (0.04)	6.12 (0.22)	2.12 (0.06)	8.27 (0.14)
	cv	0.5 (0.01)	18.54 (0.66)	17.19	**1.35**	1.42 (0.02)	6.55 (0.16)	**1.65** (0.03)	25.84 (0.69)
	csuv.m.0	**0.69** (0.01)	**5.88** (0.23)	**2.47**	3.41	1.66 (0.05)	**5.24** (0.16)	1.77 (0.04)	9.06 (0.18)
	csuv.s.0	0.62 (0.01)	9.49 (0.27)	7.22	2.27	**1.38** (0.02)	5.69 (0.14)	**1.65** (0.03)	14.95 (0.22)
	csuv.m.5	0.68 (0.01)	5.92 (0.23)	2.38	3.54	1.71 (0.06)	5.34 (0.16)	1.8 (0.04)	8.84 (0.18)
	csuv.s.5	0.62 (0.01)	9.76 (0.26)	7.54	2.22	1.38 (0.02)	5.67 (0.13)	1.64 (0.03)	15.32 (0.21)
	csuv.m.0.all	0.7 (0.01)	5.99 (0.25)	2.86	3.13	1.58 (0.04)	5.21 (0.17)	1.73 (0.04)	9.73 (0.18)
	csuv.s.0.all	0.6 (0.01)	10.58 (0.28)	8.54	2.04	1.38 (0.02)	5.79 (0.13)	1.64 (0.03)	16.5 (0.25)
	csuv.m.0.mcp	0.4 (0.01)	7.64 (0.13)	0.31	7.33	5.38 (0.28)	7.79 (0.14)	2.71 (0.04)	2.98 (0.12)
	csuv.s.0.mcp	0.6 (0.01)	7.01 (0.21)	2.25	4.76	1.98 (0.06)	6.57 (0.19)	2.22 (0.05)	7.49 (0.08)
setting 8	lasso	0.33 (0.01)	34.07 (0.92)	32.27	1.8	1.59 (0.03)	10.03 (0.23)	2.1 (0.03)	40.47 (0.9)
	elastic net	0.29 (0.01)	42.66 (0.87)	41.11	1.55	1.62 (0.03)	10.86 (0.21)	2.15 (0.03)	49.56 (0.84)
block.cor = 0.9	relaxed lasso	0.35 (0.01)	28.37 (0.78)	25.88	2.49	1.63 (0.03)	10.14 (0.22)	2.19 (0.04)	33.39 (0.88)
s = 10	mcp	0.38 (0.02)	12.94 (0.35)	6.81	6.13	2.2 (0.06)	11.21 (0.31)	3.09 (0.06)	10.68 (0.18)
p = 300	scad	0.43 (0.02)	12.2 (0.39)	6.8	5.4	2.01 (0.05)	9.94 (0.32)	2.89 (0.07)	11.4 (0.21)
	vsd	0.44 (0.02)	10 (0.31)	3.78	6.22	2.73 (0.17)	9.8 (0.29)	2.97 (0.06)	7.56 (0.14)
	bic	0.42 (0.02)	14 (1)	8.7	5.3	2.05 (0.05)	10.53 (0.34)	2.92 (0.07)	13.4 (1.09)
	ebic	0.44 (0.02)	11.7 (0.35)	6.29	5.41	2.03 (0.05)	10.05 (0.31)	2.9 (0.06)	10.88 (0.18)
	cv	0.33 (0.01)	34.07 (0.92)	32.27	**1.8**	**1.59** (0.03)	10.03 (0.23)	**2.1** (0.03)	40.47 (0.9)
	csuv.m.0	**0.52** (0.01)	**7.84** (0.22)	**2.15**	5.69	3.96 (0.16)	**7.74** (0.21)	2.52 (0.05)	6.46 (0.16)
	csuv.s.0	0.42 (0.01)	20.14 (0.32)	17.25	2.89	1.65 (0.03)	9.4 (0.2)	2.21 (0.04)	24.36 (0.25)
	csuv.m.5	0.52 (0.01)	7.69 (0.21)	1.87	5.82	4.29 (0.16)	7.61 (0.2)	2.5 (0.05)	6.05 (0.15)
	csuv.s.5	0.41 (0.01)	20.32 (0.32)	17.36	2.96	1.67 (0.03)	9.45 (0.2)	2.21 (0.04)	24.4 (0.23)
	csuv.m.0.all	0.53 (0.01)	7.91 (0.21)	2.46	5.45	3.55 (0.15)	7.77 (0.19)	2.53 (0.05)	7.01 (0.16)
	csuv.s.0.all	0.41 (0.01)	21.06 (0.34)	18.27	2.79	1.65 (0.03)	9.47 (0.2)	2.19 (0.04)	25.48 (0.26)
	csuv.m.0.mcp	0.23 (0.01)	9.67 (0.09)	0.22	9.45	10.76 (0.22)	8.96 (0.1)	2.97 (0.03)	0.77 (0.09)
	csuv.s.0.mcp	0.36 (0.02)	11.97 (0.28)	5.33	6.64	3.41 (0.12)	10.68 (0.25)	3.08 (0.05)	8.69 (0.05)

Table A.7 Model 3: performance of CSUV and methods it compares with (continue). Variable selection performance in terms of F-measure (f), total error (FP+FN), false positives (FP) and false negatives (FN), prediction error in terms of mse (pred.err) and estimation error in terms of l1 and l2 distance (l1.diff and l2.diff) and are shown. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

parameters	methods	f	FP+FN	FP	FN	pred.err	l1.diff	l2.diff	size
setting 1	lasso	0.37 (0.01)	18.19 (0.89)	17.98	0.21	1.85 (0.09)	3.02 (0.13)	0.9 (0.03)	22.77 (0.89)
	elastic net	0.32 (0.01)	23.2 (0.97)	23.01	0.19	1.98 (0.08)	3.57 (0.13)	0.98 (0.03)	27.82 (0.98)
num.factor = 2	relaxed lasso	0.5 (0.02)	11.16 (0.77)	10.68	0.48	1.89 (0.08)	2.77 (0.12)	0.91 (0.03)	15.2 (0.81)
s = 5	mcp	0.75 (0.02)	3.26 (0.28)	2.57	0.69	1.71 (0.09)	1.67 (0.11)	0.76 (0.04)	6.88 (0.26)
p = 100	scad	0.65 (0.02)	5.65 (0.37)	5.17	0.48	1.67 (0.09)	1.62 (0.1)	0.73 (0.04)	9.69 (0.37)
	vsd	0.82 (0.02)	1.71 (0.15)	0.19	1.52	2.22 (0.15)	1.9 (0.13)	0.98 (0.06)	3.67 (0.13)
	bic	0.72 (0.02)	4.01 (0.32)	3.41	0.6	1.69 (0.1)	1.67 (0.11)	0.74 (0.04)	7.81 (0.31)
	ebic	0.75 (0.02)	3.18 (0.28)	2.52	0.66	**1.7** (0.1)	**1.62** (0.11)	**0.74** (0.04)	6.86 (0.27)
	cv	0.47 (0.02)	14.49 (1.1)	14.18	**0.31**	1.8 (0.09)	2.66 (0.15)	0.85 (0.03)	18.87 (1.12)
	csuv.m.0	**0.81** (0.02)	**1.81** (0.15)	**0.82**	0.99	1.95 (0.13)	1.73 (0.11)	0.85 (0.05)	4.83 (0.13)
	csuv.s.0	0.69 (0.01)	4.03 (0.23)	3.34	0.69	1.84 (0.09)	2.05 (0.1)	0.85 (0.04)	7.65 (0.23)
	csuv.m.5	0.82 (0.02)	1.7 (0.14)	0.67	1.03	1.95 (0.13)	1.71 (0.11)	0.85 (0.05)	4.64 (0.12)
	csuv.s.5	0.67 (0.01)	4.43 (0.21)	3.73	0.7	1.84 (0.09)	2.12 (0.1)	0.86 (0.04)	8.03 (0.21)
	csuv.m.0.all	0.81 (0.02)	1.93 (0.18)	0.98	0.95	1.87 (0.11)	1.71 (0.11)	0.83 (0.05)	5.03 (0.15)
	csuv.s.0.all	0.66 (0.01)	4.67 (0.24)	4	0.67	1.84 (0.09)	2.14 (0.1)	0.86 (0.04)	8.33 (0.26)
	csuv.m.0.mcp	0.78 (0.02)	1.73 (0.13)	0.15	1.58	2.32 (0.14)	2.03 (0.11)	1.04 (0.05)	3.57 (0.1)
	csuv.s.0.mcp	0.78 (0.02)	2.19 (0.17)	1.08	1.11	2.03 (0.11)	1.91 (0.11)	0.92 (0.05)	4.97 (0.1)
setting 2	lasso	0.3 (0.01)	26.02 (1.59)	25.54	0.48	2.19 (0.08)	3.84 (0.15)	1.09 (0.03)	30.06 (1.64)
	elastic net	0.24 (0.01)	32.73 (1.45)	32.32	0.41	2.41 (0.09)	4.46 (0.14)	1.19 (0.03)	36.91 (1.49)
num.factor = 2	relaxed lasso	0.43 (0.01)	14.49 (1.11)	13.72	0.77	2.17 (0.09)	3.44 (0.13)	1.07 (0.03)	17.95 (1.18)
s = 5	mcp	0.68 (0.02)	4.14 (0.28)	3.19	0.95	2.02 (0.12)	1.94 (0.13)	0.89 (0.05)	7.24 (0.26)
p = 300	scad	0.53 (0.01)	8.71 (0.45)	8.12	0.59	1.83 (0.1)	1.87 (0.11)	0.8 (0.04)	12.53 (0.45)
	vsd	0.68 (0.02)	2.52 (0.15)	0.19	2.33	3.16 (0.22)	2.61 (0.14)	1.32 (0.06)	2.86 (0.13)
	bic	0.57 (0.02)	10.8 (1.67)	10.12	0.68	1.91 (0.11)	2.35 (0.2)	0.86 (0.05)	14.44 (1.7)
	ebic	0.66 (0.02)	5.83 (1.15)	4.94	0.89	**2.03** (0.13)	**2.08** (0.16)	**0.88** (0.05)	9.05 (1.17)
	cv	0.4 (0.02)	20.34 (1.67)	19.83	**0.51**	2.12 (0.09)	3.26 (0.17)	1.01 (0.04)	24.32 (1.7)
	csuv.m.0	**0.75** (0.02)	**2.09** (0.13)	**0.52**	1.57	2.28 (0.12)	2.11 (0.1)	1.07 (0.05)	3.95 (0.14)
	csuv.s.0	0.62 (0.01)	5.37 (0.27)	4.44	0.93	2.05 (0.09)	2.49 (0.1)	0.99 (0.04)	8.51 (0.32)
	csuv.m.5	0.74 (0.02)	2.13 (0.14)	0.44	1.69	2.42 (0.13)	2.18 (0.11)	1.11 (0.05)	3.75 (0.13)
	csuv.s.5	0.58 (0.01)	6.04 (0.25)	5.06	0.98	2.11 (0.09)	2.63 (0.09)	1.02 (0.04)	9.08 (0.31)
	csuv.m.0.all	0.75 (0.02)	2.15 (0.13)	0.64	1.51	2.22 (0.11)	2.1 (0.1)	1.05 (0.05)	4.13 (0.15)
	csuv.s.0.all	0.59 (0.01)	6.2 (0.32)	5.31	0.89	2.07 (0.09)	2.58 (0.1)	1 (0.04)	9.42 (0.37)
	csuv.m.0.mcp	0.66 (0.02)	2.47 (0.12)	0.04	2.43	2.92 (0.15)	2.61 (0.1)	1.33 (0.04)	2.61 (0.12)
	csuv.s.0.mcp	0.71 (0.02)	2.63 (0.15)	1.03	1.6	2.57 (0.15)	2.41 (0.12)	1.17 (0.05)	4.43 (0.13)
setting 3	lasso	0.45 (0.01)	25.15 (0.93)	24.76	0.39	2.6 (0.11)	5.16 (0.15)	1.23 (0.03)	34.37 (0.95)
	elastic net	0.41 (0.01)	29.57 (0.97)	29.24	0.33	2.73 (0.11)	5.84 (0.15)	1.3 (0.02)	38.91 (0.99)
num.factor = 2	relaxed lasso	0.56 (0.01)	16.19 (0.9)	15.26	0.93	2.74 (0.12)	4.82 (0.14)	1.25 (0.03)	24.33 (1)
s = 10	mcp	0.79 (0.01)	4.32 (0.24)	2.68	1.64	2.62 (0.16)	3.08 (0.16)	1.14 (0.05)	11.04 (0.22)
p = 100	scad	0.74 (0.01)	6.33 (0.27)	5.11	1.22	2.49 (0.14)	3.02 (0.13)	1.11 (0.05)	13.89 (0.29)
	vsd	0.71 (0.02)	4.26 (0.24)	0.19	4.07	4.24 (0.29)	4.36 (0.19)	1.67 (0.06)	6.12 (0.24)
	bic	0.79 (0.01)	4.77 (0.26)	3.47	1.3	2.47 (0.14)	2.96 (0.15)	1.09 (0.05)	12.17 (0.25)
	ebic	**0.8** (0.01)	4.32 (0.23)	2.85	1.47	**2.56** (0.14)	**2.98** (0.15)	**1.11** (0.05)	11.38 (0.25)
	cv	0.47 (0.01)	24 (1.03)	23.48	**0.52**	2.67 (0.11)	5.1 (0.15)	1.24 (0.03)	32.96 (1.09)
	csuv.m.0	0.79 (0.01)	**3.71** (0.16)	**1.05**	2.66	3.29 (0.17)	3.66 (0.14)	1.4 (0.04)	8.39 (0.2)
	csuv.s.0	0.71 (0.01)	7.01 (0.29)	5.28	1.73	2.82 (0.14)	3.85 (0.12)	1.27 (0.04)	13.55 (0.34)
	csuv.m.5	0.79 (0.01)	3.73 (0.17)	0.94	2.79	3.39 (0.18)	3.71 (0.14)	1.42 (0.05)	8.15 (0.19)
	csuv.s.5	0.7 (0.01)	7.14 (0.27)	5.45	1.69	2.87 (0.14)	3.86 (0.12)	1.26 (0.04)	13.76 (0.32)
	csuv.m.0.all	0.8 (0.01)	3.74 (0.17)	1.25	2.49	3.19 (0.18)	3.58 (0.13)	1.37 (0.04)	8.76 (0.21)
	csuv.s.0.all	0.69 (0.01)	7.88 (0.33)	6.3	1.58	2.84 (0.14)	3.96 (0.13)	1.26 (0.04)	14.72 (0.37)
	csuv.m.0.mcp	0.67 (0.01)	4.84 (0.15)	0.11	4.73	5.18 (0.32)	5.18 (0.16)	1.95 (0.05)	5.38 (0.16)
	csuv.s.0.mcp	0.75 (0.01)	4.26 (0.19)	0.93	3.33	4.23 (0.29)	4.28 (0.17)	1.61 (0.05)	7.6 (0.17)

Table A.8 Model 4: performance of CSUV and methods it compares with. Variable selection performance in terms of F-measure (f), total error (FP+FN), false positives (FP) and false negatives (FN), prediction error in terms of mse (pred.err) and estimation error in terms of l1 and l2 distance (l1.diff and l2.diff) and are shown. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

parameters	methods	f	FP+FN	FP	FN	pred.err	l1.diff	l2.diff	size
setting 4	lasso	0.32 (0.01)	40.26 (1.61)	39.05	1.21	3.71 (0.18)	7.52 (0.2)	1.64 (0.03)	47.84 (1.67)
	elastic net	0.28 (0.01)	47.74 (1.62)	46.54	1.2	4.11 (0.19)	8.45 (0.19)	1.74 (0.03)	55.34 (1.68)
num.factor = 2	relaxed lasso	0.43 (0.01)	23.75 (1.41)	21.6	2.15	3.87 (0.22)	6.88 (0.21)	1.67 (0.04)	29.45 (1.59)
s = 10	mcp	0.67 (0.01)	7.02 (0.35)	4.06	2.96	4 (0.26)	4.64 (0.2)	1.63 (0.06)	11.1 (0.36)
p = 300	scad	0.58 (0.01)	11.98 (0.47)	9.91	2.07	3.48 (0.22)	4.36 (0.18)	1.49 (0.06)	17.84 (0.49)
	vsd	0.5 (0.02)	7.11 (0.24)	0.26	6.85	6.99 (0.38)	6.89 (0.23)	2.41 (0.07)	3.41 (0.23)
	bic	0.54 (0.02)	22.46 (2.35)	20.64	1.82	3.43 (0.2)	5.61 (0.3)	1.52 (0.05)	28.82 (2.45)
	ebic	**0.66** (0.02)	9.52 (1.27)	7	2.52	**3.63** (0.22)	**4.53** (0.21)	**1.53** (0.05)	14.48 (1.35)
	cv	0.36 (0.01)	36.39 (1.7)	35.02	**1.37**	3.64 (0.17)	7.12 (0.21)	1.63 (0.03)	43.65 (1.81)
	csuv.m.0	0.61 (0.02)	**5.79** (0.17)	**0.63**	5.16	5.54 (0.31)	5.6 (0.16)	2.03 (0.05)	5.47 (0.21)
	csuv.s.0	0.56 (0.01)	10.06 (0.34)	6.51	3.55	4.4 (0.22)	5.77 (0.17)	1.79 (0.05)	12.96 (0.47)
	csuv.m.5	0.6 (0.02)	5.89 (0.17)	0.55	5.34	5.65 (0.31)	5.74 (0.16)	2.07 (0.05)	5.21 (0.2)
	csuv.s.5	0.55 (0.01)	10.51 (0.33)	6.95	3.56	4.49 (0.23)	5.91 (0.16)	1.81 (0.05)	13.39 (0.46)
	csuv.m.0.all	0.62 (0.01)	5.86 (0.18)	0.85	5.01	5.38 (0.3)	5.6 (0.16)	2.02 (0.05)	5.84 (0.22)
	csuv.s.0.all	0.55 (0.01)	11.23 (0.39)	7.89	3.34	4.32 (0.21)	5.88 (0.16)	1.77 (0.05)	14.55 (0.51)
	csuv.m.0.mcp	0.46 (0.02)	6.98 (0.16)	0.05	6.93	7.18 (0.43)	6.98 (0.17)	2.46 (0.05)	3.12 (0.16)
	csuv.s.0.mcp	0.58 (0.02)	6.33 (0.2)	0.9	5.43	5.84 (0.34)	6.06 (0.18)	2.16 (0.05)	5.47 (0.18)
setting 5	lasso	0.39 (0.01)	15.08 (0.68)	14.51	0.57	2.5 (0.09)	4 (0.16)	1.22 (0.03)	18.94 (0.71)
	elastic net	0.33 (0.01)	20.63 (0.96)	20.2	0.43	2.76 (0.1)	4.84 (0.21)	1.32 (0.03)	24.77 (0.97)
num.factor = 10	relaxed lasso	0.42 (0.01)	12.22 (0.73)	11.21	1.01	2.83 (0.13)	4.3 (0.21)	1.32 (0.04)	15.2 (0.8)
s = 5	mcp	0.55 (0.02)	5.41 (0.29)	3.58	1.83	3.36 (0.21)	3.69 (0.2)	1.43 (0.07)	6.75 (0.21)
p = 100	scad	0.51 (0.02)	6.87 (0.33)	5.25	1.62	3.59 (0.25)	3.64 (0.2)	1.42 (0.07)	8.63 (0.27)
	vsd	0.57 (0.03)	3.84 (0.22)	1.02	2.82	3.96 (0.24)	3.65 (0.18)	1.59 (0.07)	3.2 (0.13)
	bic	0.57 (0.02)	5.46 (0.29)	3.83	1.63	3.36 (0.21)	3.66 (0.2)	1.4 (0.07)	7.2 (0.23)
	ebic	0.57 (0.02)	5.24 (0.29)	3.52	1.72	3.27 (0.21)	3.56 (0.2)	1.38 (0.07)	6.8 (0.21)
	cv	0.39 (0.01)	14.96 (0.7)	14.3	**0.66**	**2.53** (0.1)	4 (0.16)	**1.22** (0.04)	18.64 (0.74)
	csuv.m.0	**0.63** (0.02)	**3.61** (0.2)	**1.72**	1.89	2.86 (0.14)	**3.08** (0.14)	1.31 (0.05)	4.83 (0.14)
	csuv.s.0	0.53 (0.01)	6.52 (0.22)	5.24	1.28	2.61 (0.11)	3.42 (0.13)	1.25 (0.04)	8.96 (0.23)
	csuv.m.5	0.62 (0.02)	3.48 (0.19)	1.48	2	2.99 (0.15)	3.12 (0.15)	1.35 (0.05)	4.48 (0.13)
	csuv.s.5	0.52 (0.01)	6.67 (0.21)	5.34	1.33	2.63 (0.11)	3.43 (0.13)	1.26 (0.04)	9.01 (0.21)
	csuv.m.0.all	0.62 (0.02)	3.77 (0.21)	1.93	1.84	2.82 (0.13)	3.11 (0.14)	1.31 (0.05)	5.09 (0.15)
	csuv.s.0.all	0.52 (0.01)	7.29 (0.25)	6.11	1.18	2.61 (0.12)	3.5 (0.13)	1.25 (0.04)	9.93 (0.27)
	csuv.m.0.mcp	0.55 (0.02)	3.45 (0.16)	0.48	2.97	3.78 (0.21)	3.43 (0.14)	1.59 (0.05)	2.51 (0.09)
	csuv.s.0.mcp	0.56 (0.02)	4.49 (0.21)	2.46	2.03	3.14 (0.15)	3.49 (0.16)	1.42 (0.05)	5.43 (0.09)
setting 6	lasso	0.28 (0.01)	22.46 (1.13)	21.37	1.09	3.19 (0.15)	5.23 (0.2)	1.46 (0.03)	25.28 (1.15)
	elastic net	0.23 (0.01)	30.03 (1.38)	29.08	0.95	3.48 (0.15)	5.98 (0.21)	1.55 (0.03)	33.13 (1.39)
num.factor = 10	relaxed lasso	0.32 (0.01)	15.45 (0.87)	13.71	1.74	3.55 (0.19)	5.2 (0.21)	1.56 (0.04)	16.97 (0.92)
s = 5	mcp	0.42 (0.02)	7.03 (0.29)	4.27	2.76	4.64 (0.27)	4.86 (0.21)	1.79 (0.06)	6.51 (0.22)
p = 300	scad	0.37 (0.02)	9.32 (0.3)	7.05	2.27	4.29 (0.29)	4.58 (0.22)	1.71 (0.07)	9.78 (0.25)
	vsd	0.48 (0.02)	4.78 (0.18)	1.06	3.72	5.31 (0.3)	4.62 (0.16)	1.98 (0.06)	2.34 (0.12)
	bic	0.41 (0.02)	10.11 (1.19)	7.7	2.41	4.57 (0.31)	5.2 (0.28)	1.77 (0.07)	10.29 (1.24)
	ebic	0.42 (0.02)	7.11 (0.31)	4.46	2.65	4.69 (0.32)	4.83 (0.23)	1.77 (0.07)	6.81 (0.24)
	cv	0.29 (0.01)	21.97 (1.16)	20.83	**1.14**	**3.24** (0.16)	5.19 (0.21)	**1.47** (0.03)	24.69 (1.19)
	csuv.m.0	**0.5** (0.02)	**4.27** (0.16)	**1.35**	2.92	3.96 (0.2)	**3.85** (0.12)	1.66 (0.04)	3.43 (0.12)
	csuv.s.0	0.39 (0.01)	9.17 (0.28)	7.05	2.12	3.61 (0.18)	4.51 (0.14)	1.56 (0.04)	9.93 (0.29)
	csuv.m.5	0.49 (0.02)	4.23 (0.15)	1.25	2.98	4.03 (0.2)	3.86 (0.12)	1.68 (0.04)	3.27 (0.13)
	csuv.s.5	0.39 (0.01)	9.12 (0.28)	6.97	2.15	3.68 (0.18)	4.57 (0.14)	1.58 (0.04)	9.82 (0.28)
	csuv.m.0.all	0.5 (0.02)	4.42 (0.18)	1.67	2.75	3.82 (0.2)	3.81 (0.13)	1.61 (0.04)	3.92 (0.14)
	csuv.s.0.all	0.37 (0.01)	10.27 (0.32)	8.26	2.01	3.52 (0.16)	4.63 (0.14)	1.55 (0.04)	11.25 (0.34)
	csuv.m.0.mcp	0.43 (0.01)	4.38 (0.12)	0.37	4.01	4.91 (0.26)	4.23 (0.12)	1.93 (0.04)	1.36 (0.08)
	csuv.s.0.mcp	0.41 (0.02)	5.95 (0.17)	2.92	3.03	4.48 (0.24)	4.63 (0.13)	1.8 (0.04)	4.89 (0.1)

Table A.9 Model 4: performance of CSUV and methods it compares with (continue). Variable selection performance in terms of F-measure (f), total error (FP+FN), false positives (FP) and false negatives (FN), prediction error in terms of mse (pred.err) and estimation error in terms of l1 and l2 distance (l1.diff and l2.diff) and are shown. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

parameters	methods	f	FP+FN	FP	FN	pred.err	l1.diff	l2.diff	size
setting 7	lasso	0.44 (0.01)	21.75 (0.68)	19.99	1.76	4.57 (0.2)	7.63 (0.19)	1.84 (0.04)	28.23 (0.77)
	elastic net	0.4 (0.01)	27.39 (0.79)	26.07	1.32	4.83 (0.23)	8.52 (0.24)	1.89 (0.04)	34.75 (0.86)
num.factor = 10	relaxed lasso	0.46 (0.01)	19.07 (0.88)	16.61	2.46	5.34 (0.26)	8.24 (0.27)	2 (0.04)	24.15 (1.05)
s = 10	mcp	0.51 (0.02)	9.26 (0.28)	4.25	5.01	7.58 (0.44)	8.19 (0.24)	2.47 (0.06)	9.24 (0.27)
p = 100	scad	0.51 (0.02)	9.76 (0.32)	5.05	4.71	7.41 (0.43)	7.93 (0.25)	2.43 (0.06)	10.34 (0.22)
	vsd	0.42 (0.02)	8.28 (0.23)	1.28	7	8.89 (0.52)	8.34 (0.23)	2.7 (0.06)	4.28 (0.18)
	bic	0.55 (0.02)	8.96 (0.39)	4.57	4.39	6.89 (0.41)	7.68 (0.25)	2.32 (0.06)	10.18 (0.38)
	ebic	0.55 (0.02)	8.62 (0.3)	3.98	4.64	7 (0.41)	7.7 (0.25)	2.36 (0.06)	9.34 (0.25)
	cv	0.44 (0.01)	21.75 (0.68)	19.99	**1.76**	**4.57** (0.2)	7.63 (0.19)	**1.84** (0.04)	28.23 (0.77)
	csuv.m.0	**0.58** (0.01)	**7.1** (0.2)	**2.14**	4.96	6.13 (0.36)	**6.78** (0.17)	2.19 (0.04)	7.18 (0.2)
	csuv.s.0	0.53 (0.01)	10.88 (0.25)	7.16	3.72	5.34 (0.24)	7.08 (0.18)	2.05 (0.04)	13.44 (0.32)
	csuv.m.5	0.57 (0.01)	7.11 (0.2)	1.98	5.13	6.45 (0.38)	6.84 (0.17)	2.22 (0.04)	6.85 (0.21)
	csuv.s.5	0.53 (0.01)	10.87 (0.25)	7.15	3.72	5.25 (0.22)	7.05 (0.18)	2.04 (0.04)	13.43 (0.32)
	csuv.m.0.all	0.58 (0.01)	7.41 (0.21)	2.66	4.75	5.94 (0.32)	6.78 (0.18)	2.16 (0.05)	7.91 (0.22)
	csuv.s.0.all	0.52 (0.01)	11.72 (0.24)	8.26	3.46	5.27 (0.22)	7.18 (0.17)	2.03 (0.04)	14.8 (0.33)
	csuv.m.0.mcp	0.39 (0.01)	8.05 (0.16)	0.53	7.52	8.89 (0.47)	8.21 (0.17)	2.73 (0.05)	3.01 (0.12)
	csuv.s.0.mcp	0.53 (0.01)	7.86 (0.23)	2.4	5.46	7.07 (0.38)	7.58 (0.2)	2.38 (0.05)	6.94 (0.13)
setting 8	lasso	0.31 (0.01)	32.71 (1.19)	29.58	3.13	6.43 (0.26)	10.07 (0.22)	2.25 (0.03)	36.45 (1.33)
	elastic net	0.28 (0.01)	38.5 (1.24)	35.73	2.77	6.7 (0.26)	10.7 (0.19)	2.29 (0.03)	42.96 (1.35)
num.factor = 10	relaxed lasso	0.32 (0.01)	25.3 (1.17)	21.11	4.19	7.22 (0.31)	10.41 (0.25)	2.39 (0.04)	26.92 (1.43)
s = 10	mcp	0.35 (0.01)	12.31 (0.29)	5.46	6.85	9.21 (0.39)	10.25 (0.24)	2.91 (0.05)	8.61 (0.23)
p = 300	scad	0.34 (0.01)	13.98 (0.35)	7.49	6.49	9.88 (0.46)	9.88 (0.25)	2.89 (0.06)	11 (0.31)
	vsd	0.25 (0.01)	10.75 (0.15)	1.72	9.03	12.25 (0.59)	10.71 (0.17)	3.36 (0.05)	2.69 (0.13)
	bic	0.36 (0.02)	16.89 (1.3)	11.03	5.86	9.08 (0.44)	10.25 (0.28)	2.76 (0.06)	15.17 (1.52)
	ebic	0.37 (0.01)	12.05 (0.31)	5.52	6.53	9.34 (0.44)	9.78 (0.23)	2.82 (0.05)	8.99 (0.26)
	cv	0.31 (0.01)	32.71 (1.19)	29.58	**3.13**	**6.43** (0.26)	10.07 (0.22)	**2.25** (0.03)	36.45 (1.33)
	csuv.m.0	**0.41** (0.01)	**9** (0.2)	**2.15**	6.85	8.16 (0.35)	**8.41** (0.18)	2.65 (0.05)	5.3 (0.2)
	csuv.s.0	0.37 (0.01)	15.23 (0.26)	9.75	5.48	7.55 (0.3)	9.2 (0.16)	2.47 (0.04)	14.27 (0.35)
	csuv.m.5	0.42 (0.01)	8.87 (0.2)	1.88	6.99	8.32 (0.36)	8.38 (0.19)	2.68 (0.05)	4.89 (0.19)
	csuv.s.5	0.37 (0.01)	15.25 (0.26)	9.77	5.48	7.41 (0.28)	9.2 (0.16)	2.47 (0.04)	14.29 (0.34)
	csuv.m.0.all	0.42 (0.01)	9.23 (0.22)	2.56	6.67	8.15 (0.34)	8.46 (0.18)	2.62 (0.04)	5.89 (0.2)
	csuv.s.0.all	0.36 (0.01)	16.64 (0.29)	11.47	5.17	7.39 (0.29)	9.41 (0.16)	2.45 (0.04)	16.3 (0.38)
	csuv.m.0.mcp	0.24 (0.01)	9.66 (0.13)	0.65	9.01	11.13 (0.5)	9.61 (0.16)	3.17 (0.05)	1.64 (0.09)
	csuv.s.0.mcp	0.34 (0.01)	10.65 (0.22)	3.27	7.38	9.61 (0.45)	9.75 (0.19)	2.92 (0.05)	5.89 (0.1)

Table A.10 Model 4: performance of CSUV and methods it compares with (continue). Variable selection performance in terms of F-measure (f), total error (FP+FN), false positives (FP) and false negatives (FN), prediction error in terms of mse (pred.err) and estimation error in terms of l1 and l2 distance (l1.diff and l2.diff) and are shown. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

parameters	methods	f	FP+FN	FP	FN	pred.err	l1.diff	l2.diff	size
setting 1	lasso	0.58 (0.02)	9.6 (0.72)	9.6	0	1.18 (0.02)	1.19 (0.05)	0.42 (0.01)	14.6 (0.72)
	elastic net	0.49 (0.02)	12.7 (0.76)	12.7	0	1.23 (0.02)	1.42 (0.06)	0.46 (0.01)	17.7 (0.76)
rho = 0.5	relaxed lasso	0.95 (0.01)	0.84 (0.32)	0.84	0	1.08 (0.02)	0.67 (0.04)	0.32 (0.01)	5.84 (0.32)
s = 5	mcp	1 (0)	0 (0)	0	0	1.06 (0.02)	0.6 (0.03)	0.33 (0.01)	5 (0)
p = 100	scad	1 (0)	0.01 (0.01)	0.01	0	1.21 (0.03)	1.02 (0.05)	0.57 (0.03)	5.01 (0.01)
	vsd	1 (0)	0.03 (0.02)	0.03	0	1.04 (0.02)	0.54 (0.02)	0.29 (0.01)	5.03 (0.02)
	bic	0.92 (0.02)	1.41 (0.33)	1.41	0	1.1 (0.02)	0.73 (0.04)	0.36 (0.02)	6.41 (0.33)
	ebic	1 (0)	0.01 (0.01)	0.01	0	1.07 (0.02)	0.62 (0.03)	0.33 (0.02)	5.01 (0.01)
	cv	1 (0)	**0** (0)	**0**	0	1.06 (0.02)	0.6 (0.03)	0.33 (0.01)	5 (0)
	csuv.m.0	1 (0)	**0** (0)	**0**	0	**1.04** (0.02)	**0.52** (0.02)	**0.28** (0.01)	5 (0)
	csuv.s.0	1 (0)	**0** (0)	**0**	0	**1.04** (0.02)	**0.52** (0.02)	**0.28** (0.01)	5 (0)
	csuv.m.5	1 (0)	0 (0)	0	0	1.04 (0.02)	0.52 (0.02)	0.28 (0.01)	5 (0)
	csuv.s.5	1 (0)	0 (0)	0	0	1.04 (0.02)	0.52 (0.02)	0.28 (0.01)	5 (0)
	csuv.m.0.all	1 (0)	0.01 (0.01)	0.01	0	1.04 (0.02)	0.53 (0.02)	0.28 (0.01)	5.01 (0.01)
	csuv.s.0.all	0.99 (0)	0.06 (0.02)	0.06	0	1.05 (0.02)	0.55 (0.02)	0.29 (0.01)	5.06 (0.02)
	csuv.m.0.mcp	1 (0)	0 (0)	0	0	1.04 (0.02)	0.52 (0.02)	0.28 (0.01)	5 (0)
	csuv.s.0.mcp	1 (0)	0 (0)	0	0	1.04 (0.02)	0.52 (0.02)	0.28 (0.01)	5 (0)
setting 2	lasso	0.49 (0.02)	14.89 (1.18)	14.89	0	1.23 (0.02)	1.46 (0.07)	0.45 (0.01)	19.89 (1.18)
	elastic net	0.39 (0.01)	18.8 (1.03)	18.8	0	1.32 (0.02)	1.74 (0.06)	0.52 (0.01)	23.8 (1.03)
rho = 0.5	relaxed lasso	0.95 (0.01)	0.99 (0.4)	0.99	0	1.08 (0.02)	0.65 (0.05)	0.31 (0.01)	5.99 (0.4)
s = 5	mcp	1 (0)	0.03 (0.02)	0.03	0	1.06 (0.02)	0.57 (0.03)	0.3 (0.01)	5.03 (0.02)
p = 300	scad	0.99 (0)	0.07 (0.03)	0.07	0	1.18 (0.02)	0.94 (0.05)	0.52 (0.03)	5.07 (0.03)
	vsd	1 (0)	0.01 (0.01)	0.01	0	1.05 (0.02)	0.49 (0.02)	0.26 (0.01)	5.01 (0.01)
	bic	0.6 (0.03)	10.86 (1.11)	10.86	0	1.21 (0.02)	1.28 (0.07)	0.43 (0.02)	15.86 (1.11)
	ebic	0.99 (0)	0.08 (0.03)	0.08	0	1.07 (0.02)	0.61 (0.04)	0.32 (0.02)	5.08 (0.03)
	cv	**1** (0)	0.03 (0.02)	0.03	0	1.06 (0.02)	0.57 (0.03)	0.3 (0.01)	5.03 (0.02)
	csuv.m.0	**1** (0)	**0.01** (0.01)	**0.01**	0	**1.05** (0.02)	**0.49** (0.02)	**0.26** (0.01)	5.01 (0.01)
	csuv.s.0	**1** (0)	0.03 (0.02)	0.03	0	**1.05** (0.02)	0.5 (0.02)	**0.26** (0.01)	5.03 (0.02)
	csuv.m.5	1 (0)	0.01 (0.01)	0.01	0	1.05 (0.02)	0.49 (0.02)	0.26 (0.01)	5.01 (0.01)
	csuv.s.5	1 (0)	0.01 (0.01)	0.01	0	1.05 (0.02)	0.49 (0.02)	0.26 (0.01)	5.01 (0.01)
	csuv.m.0.all	1 (0)	0.03 (0.02)	0.03	0	1.05 (0.02)	0.5 (0.02)	0.26 (0.01)	5.03 (0.02)
	csuv.s.0.all	0.99 (0)	0.14 (0.03)	0.14	0	1.07 (0.02)	0.55 (0.03)	0.28 (0.01)	5.14 (0.03)
	csuv.m.0.mcp	1 (0)	0 (0)	0	0	1.04 (0.02)	0.49 (0.02)	0.26 (0.01)	5 (0)
	csuv.s.0.mcp	1 (0)	0 (0)	0	0	1.04 (0.02)	0.49 (0.02)	0.26 (0.01)	5 (0)
setting 3	lasso	0.65 (0.01)	12.17 (0.65)	12.17	0	1.29 (0.02)	1.97 (0.06)	0.56 (0.01)	22.17 (0.65)
	elastic net	0.6 (0.01)	14.68 (0.71)	14.68	0	1.34 (0.02)	2.2 (0.06)	0.59 (0.01)	24.68 (0.71)
rho = 0.5	relaxed lasso	0.97 (0.01)	0.89 (0.32)	0.85	0.04	1.15 (0.02)	1.27 (0.05)	0.46 (0.01)	10.81 (0.32)
s = 10	mcp	0.93 (0.01)	1.23 (0.09)	0.02	1.21	1.91 (0.05)	3.43 (0.1)	1.34 (0.04)	8.81 (0.09)
p = 100	scad	0.95 (0.01)	0.87 (0.09)	0.12	0.75	2.41 (0.06)	4.35 (0.09)	1.65 (0.03)	9.37 (0.09)
	vsd	0.99 (0)	0.25 (0.05)	0	0.25	1.16 (0.02)	1.29 (0.06)	0.52 (0.02)	9.75 (0.05)
	bic	0.83 (0.02)	5.52 (0.73)	5.44	0.08	1.66 (0.05)	2.72 (0.11)	0.95 (0.05)	15.36 (0.74)
	ebic	0.94 (0.01)	1.24 (0.19)	0.71	0.53	1.95 (0.06)	3.37 (0.12)	1.29 (0.05)	10.18 (0.22)
	cv	0.67 (0.01)	11.01 (0.68)	10.83	0.18	1.37 (0.04)	2.15 (0.09)	0.65 (0.03)	20.65 (0.73)
	csuv.m.0	**0.99** (0)	**0.14** (0.03)	**0.02**	0.12	**1.13** (0.02)	1.17 (0.04)	0.46 (0.02)	9.9 (0.04)
	csuv.s.0	**0.99** (0)	0.21 (0.05)	0.15	**0.06**	**1.13** (0.02)	**1.15** (0.04)	**0.44** (0.02)	10.09 (0.04)
	csuv.m.5	0.99 (0)	0.18 (0.04)	0.01	0.17	1.15 (0.02)	1.2 (0.05)	0.48 (0.02)	9.84 (0.04)
	csuv.s.5	0.99 (0)	0.3 (0.06)	0.23	0.07	1.14 (0.02)	1.18 (0.04)	0.45 (0.02)	10.16 (0.05)
	csuv.m.0.all	0.99 (0)	0.13 (0.03)	0.04	0.09	1.13 (0.02)	1.15 (0.04)	0.45 (0.02)	9.95 (0.04)
	csuv.s.0.all	0.98 (0)	0.46 (0.07)	0.42	0.04	1.14 (0.02)	1.21 (0.04)	0.46 (0.01)	10.38 (0.07)
	csuv.m.0.mcp	0.91 (0.01)	1.62 (0.09)	0	1.62	1.74 (0.05)	2.88 (0.13)	1.17 (0.05)	8.38 (0.09)
	csuv.s.0.mcp	0.89 (0)	1.92 (0.05)	0	1.92	1.91 (0.05)	3.23 (0.09)	1.32 (0.03)	8.08 (0.05)

Table A.11 Model 5: performance of CSUV and methods it compares with. Variable selection performance in terms of F-measure (f), total error (FP+FN), false positives (FP) and false negatives (FN), prediction error in terms of mse (pred.err) and estimation error in terms of l1 and l2 distance (l1.diff and l2.diff) and are shown. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

parameters	methods	f	FP+FN	FP	FN	pred.err	l1.diff	l2.diff	size
setting 4	lasso	0.55 (0.01)	19.23 (1.1)	19.23	0	1.4 (0.02)	2.38 (0.07)	0.62 (0.01)	29.23 (1.1)
	elastic net	0.48 (0.01)	24.02 (1.09)	24.02	0	1.5 (0.03)	2.76 (0.07)	0.67 (0.01)	34.02 (1.09)
rho = 0.5	relaxed lasso	0.96 (0.01)	1.3 (0.44)	1.23	0.07	1.19 (0.02)	1.39 (0.06)	0.5 (0.02)	11.16 (0.44)
s = 10	mcp	0.92 (0)	1.47 (0.09)	0.13	1.34	1.89 (0.04)	3.39 (0.09)	1.33 (0.03)	8.79 (0.08)
p = 300	scad	0.93 (0.01)	1.31 (0.11)	0.51	0.8	2.41 (0.06)	4.29 (0.09)	1.62 (0.03)	9.71 (0.12)
	vsd	0.96 (0)	0.76 (0.07)	0.02	0.74	1.35 (0.03)	1.77 (0.08)	0.74 (0.03)	9.28 (0.07)
	bic	0.61 (0.02)	16.37 (1.2)	16.32	0.05	1.49 (0.04)	2.54 (0.09)	0.72 (0.03)	26.27 (1.22)
	ebic	0.93 (0.01)	1.44 (0.17)	0.76	0.68	2.07 (0.06)	3.54 (0.11)	1.36 (0.04)	10.08 (0.21)
	cv	0.56 (0.02)	18.5 (1.16)	18.38	**0.12**	1.44 (0.03)	2.49 (0.08)	0.67 (0.02)	28.26 (1.2)
	csuv.m.0	**0.99** (0)	**0.27** (0.05)	**0.02**	0.25	1.19 (0.02)	1.3 (0.06)	0.52 (0.02)	9.77 (0.05)
	csuv.s.0	0.98 (0)	0.4 (0.06)	0.26	0.14	**1.17** (0.02)	**1.26** (0.05)	**0.49** (0.02)	10.12 (0.06)
	csuv.m.5	0.98 (0)	0.33 (0.05)	0.02	0.31	1.21 (0.02)	1.34 (0.06)	0.54 (0.02)	9.71 (0.05)
	csuv.s.5	0.97 (0)	0.72 (0.07)	0.64	0.08	1.17 (0.02)	1.3 (0.04)	0.49 (0.02)	10.56 (0.07)
	csuv.m.0.all	0.99 (0)	0.24 (0.05)	0.04	0.2	1.17 (0.02)	1.26 (0.05)	0.5 (0.02)	9.84 (0.05)
	csuv.s.0.all	0.96 (0)	0.8 (0.08)	0.72	0.08	1.19 (0.02)	1.32 (0.05)	0.5 (0.02)	10.64 (0.07)
	csuv.m.0.mcp	0.88 (0.01)	2.1 (0.08)	0	2.1	2.07 (0.08)	3.49 (0.13)	1.4 (0.05)	7.9 (0.08)
	csuv.s.0.mcp	0.89 (0)	2.01 (0.05)	0	2.01	1.99 (0.05)	3.36 (0.1)	1.37 (0.03)	7.99 (0.05)
setting 5	lasso	0.3 (0.01)	25.34 (0.85)	25.34	0	1.46 (0.03)	3.36 (0.09)	0.92 (0.02)	30.34 (0.85)
	elastic net	0.23 (0)	34.05 (0.8)	34.05	0	1.69 (0.04)	4.51 (0.1)	1.1 (0.02)	39.05 (0.8)
rho = -0.5	relaxed lasso	0.62 (0.01)	7.04 (0.48)	7.04	0	1.27 (0.03)	1.76 (0.09)	0.57 (0.02)	12.04 (0.48)
s = 5	mcp	0.98 (0.01)	0.26 (0.06)	0.26	0	1.05 (0.02)	0.55 (0.02)	0.28 (0.01)	5.26 (0.06)
p = 100	scad	0.96 (0.01)	0.55 (0.11)	0.55	0	1.05 (0.02)	0.55 (0.02)	0.28 (0.01)	5.55 (0.11)
	vsd	1 (0)	0.01 (0.01)	0.01	0	1.05 (0.02)	0.53 (0.02)	0.28 (0.01)	5.01 (0.01)
	bic	0.95 (0.01)	0.6 (0.11)	0.6	0	1.05 (0.02)	0.56 (0.02)	0.28 (0.01)	5.6 (0.11)
	ebic	0.96 (0.01)	0.43 (0.09)	0.43	0	**1.05** (0.02)	0.55 (0.02)	**0.28** (0.01)	5.43 (0.09)
	cv	0.97 (0.01)	0.41 (0.09)	0.41	0	**1.05** (0.02)	0.55 (0.02)	**0.28** (0.01)	5.41 (0.09)
	csuv.m.0	**1** (0)	**0.03** (0.02)	**0.03**	0	**1.05** (0.02)	**0.54** (0.02)	**0.28** (0.01)	5.03 (0.02)
	csuv.s.0	0.98 (0)	0.24 (0.05)	0.24	0	1.07 (0.02)	0.61 (0.03)	0.31 (0.01)	5.24 (0.05)
	csuv.m.5	1 (0)	0.03 (0.02)	0.03	0	1.05 (0.02)	0.54 (0.02)	0.28 (0.01)	5.03 (0.02)
	csuv.s.5	0.94 (0.01)	0.68 (0.07)	0.68	0	1.1 (0.02)	0.74 (0.03)	0.36 (0.01)	5.68 (0.07)
	csuv.m.0.all	1 (0)	0.03 (0.02)	0.03	0	1.05 (0.02)	0.54 (0.02)	0.28 (0.01)	5.03 (0.02)
	csuv.s.0.all	0.98 (0)	0.27 (0.05)	0.27	0	1.07 (0.02)	0.62 (0.03)	0.32 (0.01)	5.27 (0.05)
	csuv.m.0.mcp	1 (0)	0.02 (0.01)	0.02	0	1.05 (0.02)	0.53 (0.02)	0.28 (0.01)	5.02 (0.01)
	csuv.s.0.mcp	1 (0)	0.03 (0.02)	0.03	0	1.05 (0.02)	0.54 (0.02)	0.28 (0.01)	5.03 (0.02)
setting 6	lasso	0.21 (0)	40.76 (1.16)	40.76	0	1.86 (0.05)	5.06 (0.14)	1.28 (0.03)	45.76 (1.16)
	elastic net	0.15 (0)	56.91 (1.19)	56.9	0.01	2.72 (0.08)	7.8 (0.19)	1.79 (0.04)	61.89 (1.19)
rho = -0.5	relaxed lasso	0.41 (0.01)	17.7 (1.17)	17.64	0.06	1.74 (0.06)	3.86 (0.2)	0.98 (0.04)	22.58 (1.18)
s = 5	mcp	0.96 (0.01)	0.48 (0.1)	0.48	0	1.05 (0.02)	0.56 (0.03)	0.28 (0.01)	5.48 (0.1)
p = 300	scad	0.92 (0.01)	1.09 (0.16)	1.09	0	1.05 (0.02)	0.56 (0.02)	0.28 (0.01)	6.09 (0.16)
	vsd	1 (0)	0.04 (0.02)	0.04	0	1.05 (0.02)	0.53 (0.02)	0.28 (0.01)	5.04 (0.02)
	bic	0.82 (0.03)	7.95 (1.89)	7.95	0	1.19 (0.05)	1.31 (0.21)	0.41 (0.04)	12.95 (1.89)
	ebic	0.93 (0.01)	0.81 (0.11)	0.81	**0**	**1.05** (0.02)	**0.56** (0.03)	**0.28** (0.01)	5.81 (0.11)
	cv	0.92 (0.01)	1.04 (0.16)	1.04	**0**	**1.05** (0.02)	**0.56** (0.02)	**0.28** (0.01)	6.04 (0.16)
	csuv.m.0	**1** (0)	**0.05** (0.02)	**0.04**	0.01	1.07 (0.02)	**0.56** (0.04)	0.3 (0.02)	5.03 (0.02)
	csuv.s.0	0.96 (0.01)	0.51 (0.08)	0.51	**0**	1.1 (0.02)	0.7 (0.04)	0.35 (0.01)	5.51 (0.08)
	csuv.m.5	0.99 (0)	0.06 (0.02)	0.05	0.01	1.07 (0.02)	0.57 (0.04)	0.3 (0.02)	5.04 (0.02)
	csuv.s.5	0.86 (0.01)	1.73 (0.11)	1.73	0	1.18 (0.02)	1.03 (0.04)	0.46 (0.01)	6.73 (0.11)
	csuv.m.0.all	1 (0)	0.05 (0.02)	0.04	0.01	1.07 (0.02)	0.56 (0.04)	0.3 (0.02)	5.03 (0.02)
	csuv.s.0.all	0.95 (0.01)	0.54 (0.08)	0.54	0	1.1 (0.02)	0.71 (0.04)	0.35 (0.02)	5.54 (0.08)
	csuv.m.0.mcp	0.99 (0)	0.05 (0.03)	0.02	0.03	1.08 (0.03)	0.59 (0.05)	0.31 (0.03)	4.99 (0.03)
	csuv.s.0.mcp	0.99 (0)	0.12 (0.04)	0.12	0	1.06 (0.02)	0.56 (0.03)	0.29 (0.01)	5.12 (0.04)

Table A.12 Model 5: performance of CSUV and methods it compares with (continue). Variable selection performance in terms of F-measure (f), total error (FP+FN), false positives (FP) and false negatives (FN), prediction error in terms of mse (pred.err) and estimation error in terms of l1 and l2 distance (l1.diff and l2.diff) and are shown. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

parameters	methods	f	FP+FN	FP	FN	pred.err	l1.diff	l2.diff	size
setting 7	lasso	0.37 (0)	34.06 (0.68)	33.96	0.1	1.86 (0.04)	5.83 (0.12)	1.24 (0.03)	43.86 (0.7)
	elastic net	0.35 (0)	38.37 (0.66)	38.29	0.08	2.09 (0.05)	6.83 (0.13)	1.4 (0.03)	48.21 (0.68)
rho = -0.5	relaxed lasso	0.48 (0.01)	22.32 (1.04)	21.73	0.59	1.87 (0.04)	5.2 (0.14)	1.18 (0.03)	31.14 (1.15)
s = 10	mcp	0.85 (0.01)	3.04 (0.21)	1.53	1.51	1.56 (0.04)	2.69 (0.12)	1.06 (0.04)	10.02 (0.15)
p = 100	scad	0.76 (0.01)	5.14 (0.26)	3.25	1.89	1.87 (0.05)	3.81 (0.11)	1.45 (0.03)	11.36 (0.23)
	vsd	0.81 (0.01)	3.01 (0.19)	0.16	2.85	1.81 (0.06)	3.24 (0.14)	1.3 (0.05)	7.31 (0.18)
	bic	0.84 (0.01)	3.58 (0.46)	2.21	1.37	1.58 (0.04)	2.78 (0.13)	1.08 (0.04)	10.84 (0.46)
	ebic	**0.85** (0.01)	**3.09** (0.21)	1.59	**1.5**	**1.58** (0.04)	**2.74** (0.12)	**1.07** (0.05)	10.09 (0.16)
	cv	0.77 (0.01)	5.01 (0.27)	3.09	1.92	1.83 (0.05)	3.68 (0.12)	1.41 (0.04)	11.17 (0.24)
	csuv.m.0	0.76 (0.01)	3.92 (0.13)	**0.22**	3.7	2.08 (0.05)	3.95 (0.11)	1.57 (0.04)	6.52 (0.13)
	csuv.s.0	0.8 (0.01)	3.83 (0.18)	1.56	2.27	1.85 (0.06)	3.44 (0.13)	1.3 (0.04)	9.29 (0.19)
	csuv.m.5	0.75 (0.01)	3.99 (0.13)	0.17	3.82	2.11 (0.05)	4.03 (0.11)	1.6 (0.03)	6.35 (0.13)
	csuv.s.5	0.79 (0.01)	4.15 (0.18)	1.89	2.26	1.92 (0.06)	3.59 (0.13)	1.33 (0.04)	9.63 (0.17)
	csuv.m.0.all	0.76 (0.01)	3.87 (0.13)	0.25	3.62	2.06 (0.05)	3.91 (0.11)	1.55 (0.03)	6.63 (0.13)
	csuv.s.0.all	0.8 (0.01)	3.86 (0.17)	1.71	2.15	1.84 (0.06)	3.4 (0.12)	1.28 (0.04)	9.56 (0.18)
	csuv.m.0.mcp	0.72 (0.01)	4.25 (0.12)	0.03	4.22	2.17 (0.05)	4.22 (0.11)	1.67 (0.03)	5.81 (0.12)
	csuv.s.0.mcp	0.79 (0.01)	3.55 (0.16)	0.54	3.01	1.97 (0.06)	3.67 (0.12)	1.44 (0.04)	7.53 (0.14)
setting 8	lasso	0.26 (0)	50.38 (1.39)	48.77	1.61	3.4 (0.11)	10.41 (0.21)	2.23 (0.05)	57.16 (1.54)
	elastic net	0.22 (0)	59.24 (1.43)	57.32	1.92	4.64 (0.17)	13.28 (0.23)	2.75 (0.06)	65.4 (1.63)
rho = -0.5	relaxed lasso	0.35 (0.01)	26.61 (1.59)	22.95	3.66	3.76 (0.14)	9.65 (0.24)	2.33 (0.06)	29.29 (1.96)
s = 10	mcp	0.76 (0.01)	5.25 (0.28)	3.29	1.96	1.68 (0.05)	3.11 (0.13)	1.18 (0.05)	11.33 (0.25)
p = 300	scad	0.63 (0.01)	9.48 (0.37)	7.35	2.13	1.97 (0.05)	4.2 (0.12)	1.5 (0.04)	15.22 (0.33)
	vsd	0.67 (0.01)	4.87 (0.16)	0.02	4.85	2.49 (0.07)	4.98 (0.16)	1.89 (0.05)	5.17 (0.16)
	bic	0.61 (0.02)	20.31 (2.56)	18.77	1.54	2.04 (0.09)	5.1 (0.37)	1.4 (0.06)	27.23 (2.63)
	ebic	**0.75** (0.01)	**5.32** (0.28)	3.37	**1.95**	**1.69** (0.05)	**3.12** (0.13)	**1.18** (0.05)	11.42 (0.26)
	cv	0.67 (0.01)	8.23 (0.42)	6.13	2.1	1.87 (0.05)	3.85 (0.13)	1.4 (0.04)	14.03 (0.37)
	csuv.m.0	0.59 (0.01)	5.77 (0.11)	**0.09**	5.68	2.96 (0.08)	5.94 (0.14)	2.2 (0.04)	4.41 (0.12)
	csuv.s.0	0.64 (0.01)	6.02 (0.17)	1.64	4.38	2.61 (0.09)	5.37 (0.16)	1.89 (0.05)	7.26 (0.23)
	csuv.m.5	0.59 (0.01)	5.77 (0.11)	0.08	5.69	2.99 (0.09)	5.95 (0.15)	2.2 (0.04)	4.39 (0.12)
	csuv.s.5	0.64 (0.01)	6.69 (0.2)	2.65	4.04	2.56 (0.07)	5.41 (0.15)	1.84 (0.04)	8.61 (0.22)
	csuv.m.0.all	0.59 (0.01)	5.77 (0.11)	0.09	5.68	2.96 (0.08)	5.94 (0.14)	2.2 (0.04)	4.41 (0.12)
	csuv.s.0.all	0.64 (0.01)	6.01 (0.17)	1.66	4.35	2.61 (0.08)	5.37 (0.16)	1.89 (0.05)	7.31 (0.23)
	csuv.m.0.mcp	0.56 (0.01)	6.05 (0.1)	0	6.05	3.18 (0.1)	6.32 (0.14)	2.32 (0.04)	3.95 (0.1)
	csuv.s.0.mcp	0.63 (0.01)	5.63 (0.14)	0.58	5.05	2.81 (0.09)	5.61 (0.15)	2.05 (0.05)	5.53 (0.17)

Table A.13 Model 5: performance of CSUV and methods it compares with (continue). Variable selection performance in terms of F-measure (f), total error (FP+FN), false positives (FP) and false negatives (FN), prediction error in terms of mse (pred.err) and estimation error in terms of l1 and l2 distance (l1.diff and l2.diff) and are shown. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

methods	pred.err	size
lasso	26.08 (0.39)	12.39 (0.09)
elastic net	26.12 (0.4)	12.38 (0.1)
relaxed lasso	26.53 (0.42)	11.21 (0.14)
mcp	26.16 (0.39)	11.29 (0.15)
scad	26.1 (0.39)	11.54 (0.12)
vsd	28.54 (0.42)	6.03 (0.16)
bic	26.2 (0.4)	11.05 (0.15)
ebic	26.19 (0.4)	11.07 (0.15)
cv	**26.08** (0.39)	12.36 (0.09)
csuv.m.0	26.64 (0.41)	10.02 (0.17)
csuv.s.0	26.73 (0.41)	9.9 (0.16)
csuv.m.5	26.58 (0.41)	10.07 (0.16)
csuv.s.5	26.63 (0.41)	10.07 (0.15)
csuv.m.0.all	26.72 (0.41)	9.95 (0.17)
csuv.s.0.all	26.76 (0.41)	9.81 (0.16)
csuv.m.0.mcp	26.33 (0.4)	10.52 (0.16)
csuv.s.0.mcp	26.39 (0.4)	10.46 (0.15)

Table A.14 Boston data: performance of CSUV and methods it compares with. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

methods	f	FP+FN	FP	FN	size
lasso	0.46 (0.01)	19.96 (1.45)	16.88	3.08	23.8 (1.48)
elastic net	0.45 (0.01)	22.59 (1.51)	20.44	2.15	28.29 (1.53)
relaxed lasso	0.56 (0.02)	10.15 (0.81)	5.85	4.3	11.55 (0.97)
mcp	0.45 (0.01)	8.97 (0.22)	2.62	6.35	6.27 (0.25)
scad	0.48 (0.02)	13.01 (0.67)	8.35	4.66	13.69 (0.56)
vsd	NaN (NA)	10 (0)	0	10	0 (0)
bic	0.43 (0.01)	15.78 (1.47)	10.5	5.28	15.22 (1.65)
ebic	0.44 (0.01)	11.33 (1.18)	5.09	6.24	8.85 (1.31)
cv	0.45 (0.02)	19.3 (1.45)	15.82	**3.48**	22.34 (1.54)
csuv.m.0	0.45 (0.01)	7.04 (0.12)	**0.03**	7.01	3.02 (0.12)
csuv.s.0	**0.65** (0.01)	**5.49** (0.19)	0.99	4.5	6.49 (0.32)
csuv.m.5	0.43 (0.01)	7.26 (0.12)	0.02	7.24	2.78 (0.12)
csuv.s.5	0.69 (0.01)	5.07 (0.19)	1.08	3.99	7.09 (0.3)
csuv.m.0.all	0.49 (0.01)	6.72 (0.13)	0.02	6.7	3.32 (0.13)
csuv.s.0.all	0.69 (0.01)	5.16 (0.22)	1.3	3.86	7.44 (0.35)
csuv.m.0.mcp	0.28 (0.01)	8.61 (0.09)	0	8.61	1.39 (0.09)
csuv.s.0.mcp	0.44 (0.01)	7.19 (0.09)	0.05	7.14	2.91 (0.09)

Table A.15 Riboflavin data with permutation: performance of CSUV and methods it compares with. The numbers are based on 100 simulations. The last 8 rows are the performance of CSUV with different parameters (e.g. csuv.m.0.mcp corresponds to CSUV with MCP as constituent method and r = 0). A bold number represents the best result among delete-n/2 cross validation, eBIC and CSUV using Lasso, MCP and SCAD while a underlined number represents the worst among them. Standard errors are shown inside the parentheses.

Appendix B

Appendix for Chapter 3

index	name of data series	number of observations
1	Selling prices of animal products (absolute prices) - monthly - old code - data from 1969 to 2006	401
2	Selling prices of crop products (absolute prices) - monthly - old codes - data from 1969 to 2006	408
3	Purchase prices of the means of agricultural production (absolute prices) - monthly - old codes - data from 1969 to 2006	270
4	Price indices of agricultural products, output (2000 = 100) - monthly data	216
5	Poultry - monthly data	567
6	Cows’milk collection and products obtained - monthly data	627
7	Meat production and foreign trade - head - monthly data	514
8	Slaughtering in slaughterhouses - monthly data	675
9	Euro area 19 international trade - monthly data	241
10	EU28 international trade - monthly data	241
11	Construction - monthly data - growth rates (NACE Rev. 2)	289
12	Industry - monthly data - growth rates (NACE Rev. 2)	266
13	Retail trade - monthly data - growth rates (NACE Rev. 2)	242
14	Effective exchange rates indices - monthly data	255
15	Industrial countries’ effective exchange rates - monthly data	315
16	Money market interest rates - monthly data	280
17	HICP at constant taxes - monthly data (index)	208
18	HICP at constant taxes - monthly data (monthly rate of change)	207
19	HICP (2015 = 100) - monthly data (annual rate of change)	209
20	HICP (2015 = 100) - monthly data (index)	291
21	HICP (2015 = 100) - monthly data (monthly rate of change)	231
22	HICP (2015 = 100) - monthly data (12-month average rate of change)	268
23	Construction - monthly data	478
24	Consumers - monthly data	424
25	Industry - monthly data	484
26	Retail sale - monthly data	436
27	Services - monthly data	203
28	Sentiment indicators - monthly data	478
29	Construction - monthly data - index (2015 = 100) (NACE Rev. 2)	290
30	Industry - monthly data - index (2015 = 100) (NACE Rev. 2)	266
31	Retail trade - monthly data - index (2015 = 100) (NACE Rev. 2)	242
32	Harmonised unemployment rates (%) - monthly data	445
33	Harmonised unemployment (1 000) - monthly data	445
34	Production in construction - monthly data	264
35	Production in industry - monthly data	240
36	Turnover in industry, total - monthly data	240
37	Turnover in industry, domestic market - monthly data	240
38	Turnover in industry, non domestic market - monthly data	240
39	Turnover in services - monthly data	204
40	Turnover and volume of sales in wholesale and retail trade - monthly data	216
41	Unemployment by sex and age – monthly data	445

Table B.1 List of real data used in numerical study