HEC MONTRÉAL
École affiliée à l'Université de Montréal

Three Essays on Nonparametric Prediction Intervals and Robust Variable Selection

par
Marie-Hélène Roy

Thèse présentée en vue de l'obtention du grade de Ph. D. en administration
(option Sciences de la décision)

Décembre 2018

HEC MONTRÉAL
École affiliée à l'Université de Montréal

Cette thèse intitulée :

Three Essays on Nonparametric Prediction Intervals and Robust Variable Selection

Présentée par :

Marie-Hélène Roy

a été évaluée par un jury composé des personnes suivantes :

Jean-François Plante
HEC Montréal
Président-rapporteur

Denis Larocque
HEC Montréal
Directeur de recherche

Codirecteur de recherche

Aurélie Labbe
HEC Montréal
Membre du jury

Hemant Ishwaran
University of Miami
Examinateur externe

Daniel Parent
HEC Montréal
Représentant du directeur de HEC Montréal

Résumé

Cette thèse propose de nouvelles méthodes dans deux champs d'expertise de l'apprentissage supervisé: la construction d'intervalles de prédiction et la sélection de variables. Les deux premiers chapitres présentent et étudient de nouvelles approches pour la construction d'intervalles de prédiction basées sur les forêts aléatoires. Dans le premier chapitre, nous considérons le contexte classique. L'impact du choix de critère de coupure utilisé dans la construction des arbres ainsi que la méthode de construction des intervalles de prédictions sont minutieusement étudiés et testés sur des données simulées et réelles. Dans le deuxième chapitre, la recherche concerne le contexte des modèles de mélange fini. Une nouvelle façon d'utiliser les forêts aléatoires pour modéliser une variation de l'algorithme EM est présentée. Cinq différentes approches pour l'obtention d'intervalles de prédiction sont étudiées en détails dans une étude par simulation. Ces méthodes sont basées sur la perspective moderne qui traite les forêts aléatoires comme une façon d'identifier des observations similaires voisines (dans l'espace des covariables) pour obtenir des estimations. Dans le troisième chapitre, une méthode robuste de filtrage de variables pour données de très grande dimension est présentée et testée sur des données simulées propres et contaminées, ainsi que sur de vrais jeux de données. Il s'agit d'une méthode ensembliste appliquée à une technique de sélection de variables robuste. Celle-ci est utilisée itérativement pour atteindre l'objectif d'éliminer graduellement les variables non désirées. La performance des méthodes proposées dans cette thèse surpasse généralement celle des compétiteurs actuels respectifs et peuvent être très utiles dans la pratique.

Mots-clés

Forêt aléatoire, Intervalle de prédiction, Calibration Out-Of-Bag, Modèle de mélange, Algorithme EM, Régression nonparamétrique, Méthodes ensemblistes, Filtrage de variables, Sélection de variables, Données en grande dimension, Forage de données, Analyse multivariée.

Méthodes de recherche: Exploitation de données, Analyse multivariée

Abstract

This thesis proposes new methods in two areas of supervised learning: Building prediction intervals and variable selection. The first two chapters present and investigate new methods based on random forests to build prediction intervals. In the first chapter, the classical setting is considered. The impact of the splitting rule used to build the trees and of the method used to build the prediction interval are thoroughly investigated with simulated and real data sets. In the second chapter, finite mixture regression models are considered. A new way of using random forests for modelling with a pseudo EM-algorithm is presented. Five ways to derive prediction intervals from this model are extensively investigated in a simulation study. These methods are based on the modern view that considers a random forest as a way to identify a set of locally (in the covariate space) similar observations for estimation. In the third chapter, a robust screening method for ultra-high dimensional data is presented and tested with clean and contaminated simulated data and a real data set. It is an ensemble method applied to a robust variable selection technique, used iteratively, to gradually eliminate spurious variables. The proposed methods presented in this thesis generally outperform their current competing methods and can be very useful in practice.

Keywords

Random Forest, Prediction Interval, Splitting rule, Out-Of-Bag Calibration, Mixture Models, EM algorithm, Nonparametric Regression, Ensemble Method, Variable Screening, Robust Variable Selection, High-Dimensional Data, Data Mining, Multivariate Analysis.

Research Methods: Data Mining, Multivariate Analysis

Contents

Résumé **iii**

Abstract **v**

List of Tables **ix**

List of Figures **xi**

Acknowledgements **xv**

General Introduction **1**

1 Prediction Intervals with Random Forests **3**

Abstract 3

1.1 Introduction 4

1.2 Methodology 8

1.2.1 Splitting Rules 9

1.2.2 Methods for Building a Prediction Interval 11

1.2.3 Calibration 12

1.2.4 Implementation 13

1.3 Simulation Study 13

1.3.1 Simulation Design 14

1.3.2 Competing Methods 16

1.3.3 Parameters and calibration for the Simulation Study 16

1.4 Results of the Simulation Study 17

1.5 Performance with Real Datasets 21
1.6 Conclusion 33
References 35

2 Prediction Intervals for Finite Mixture of Regressions Based on Random Forests 37

Abstract 37
2.1 Introduction 38
2.2 Methodology 40
2.2.1 The modified EM algorithm 41
2.2.2 Prediction Intervals 43
2.2.3 Four PI methods based on the Bag of Observations for Prediction (BOP) . . 44
2.2.4 Calibration for the four BOP methods 47
2.3 Simulation Study 48
2.3.1 Simulation Design 48
2.4 Results of the Simulation Study 50
2.4.1 Computational Efficiency 68
2.5 Conclusion 69
References 71

3 High-Dimensional Variable Screening using a Robust Ensemble Method 75

Abstract 75
3.1 Introduction 76
3.2 Methodology 80
3.2.1 Robust VIF Regression 80
3.2.2 The SERVIF Algorithm 82
3.3 Simulation Study 86
3.4 Real Data 91
3.5 Concluding remarks 93
References 93

General Conclusion 97

Appendix A **i**

List of Tables

1.1 Averaged results of the five repetitions of 10-fold cross-validation for the seven real datasets 28

2.1 Complete results of the simulation study aggregated by DGP, sample size and error variance setup 52

3.1 Mean number of covariates in the final subset - non-contaminated scenario 89

3.2 Mean number of covariates in the final subset - contaminated scenario 89

3.3 MSE and the MAE for the SERVIF and its three competitors 92

A Appendix A - Complete results of the simulation study presented in Chapter 1 aggregated by DGP and sample size i

List of Figures

1.1 Distributions of the percentage increase in PI length of each method with respect to best performer for a given run for all 68,400 runs in the simulation study 19

1.2 Distributions of the PI length for DGP 1 22

1.3 Distributions of the PI length for DGP 2 23

1.4 Distributions of the PI length for DGP 3 24

1.5 Distributions of the PI length for DGP 4 25

1.6 Distributions of the PI length for DGP 5 26

1.7 Distributions of the PI length for DGP 6 27

1.8 Distributions of the mean PI length for datasets Boston Housing, BWC and Concrete Slump 30

1.9 Distributions of the mean PI length for datasets Concrete Compression and Airfoil Self Noise 31

1.10 Distributions of the mean PI length for datasets Music Origin Latitude and Music Origin Longitude 32

2.1 Distributions of the percentage increase in PI length with respect to best performer for each run for DGP A, training sample size $n = 100$ 56

2.2 Distributions of the percentage increase in PI length with respect to best performer for each run for DGP B, training sample size $n = 100$ 57

2.3 Distributions of the percentage increase in PI length with respect to best performer for each run for DGP C, training sample size $n = 100$ 58

2.4 Distributions of the percentage increase in PI length with respect to best performer for each run for DGP D, training sample size $n = 100$. 59

2.5 Distributions of the percentage increase in PI length with respect to best performer for each run for DGP A, training sample size $n = 250$. 60
2.6 Distributions of the percentage increase in PI length with respect to best performer for each run for DGP B, training sample size $n = 250$. 61
2.7 Distributions of the percentage increase in PI length with respect to best performer for each run for DGP C, training sample size $n = 250$ 62
2.8 Distributions of the percentage increase in PI length with respect to best performer for each run for DGP D, training sample size $n = 250$ 63
2.9 Distributions of the percentage increase in PI length with respect to best performer for each run for DGP A, training sample size $n = 1000$ 64
2.10 Distributions of the percentage increase in PI length with respect to best performer for each run for DGP B, training sample size $n = 1000$ 65
2.11 Distributions of the percentage increase in PI length with respect to best performer for each run for DGP C, training sample size $n = 1000$ 66
2.12 Distributions of the percentage increase in PI length with respect to best performer for each run for DGP D, training sample size $n = 1000$ 67
2.13 Number of times, on average, a method takes to compute compared to the FMRLASSO 74

3.1 Number of selected covariates for non-contaminated scenarios 90
3.2 Number of selected covariates for contaminated scenarios 91

To all women of the era and other eras who did not have this incredible chance to pursue higher education, and more specifically, to my dear grandmother, Madeleine, who would have made a stellar researcher. And to my dad, Réjean, for his dedication, kindness and unconditional support.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Denis Larocque, for his continuous support, his guidance and kindness and for sharing his enthusiasm and his extensive knowledge. I have felt very lucky throughout my PhD to be under his supervision. I could not have imagined a better advisor and mentor to begin this path in the world of data science and scientific research.

I would also like to thank my other co-authors, Dr Ilmari Ahonen, Professor Jaakko Nevalainen & Professor Debbie Dupuis for contributing to this work and sharing their knowledge. In addition, I would like to thank the members of my PhD committee Professors Hemant Ishwaran, Aurélie Labbe and Jean-François Plante for their time, insightful feedback and suggestions.

I thank all the great people of the Department of Decision Sciences for their help, warmth and encouragement throughout my time in this program. I also thank Andrée-Ann, Hoora and Mohammed Ali for being such kind and supportive friends in and out of school throughout this journey. Your presence through all times has meant a lot to me.

My special recognition goes to my family; my parents, brother and sister, Guillaume and Laurianne and my extended family who have always supported me and encouraged me in pursuing my passion. Their presence by my side has been a precious gift.

To my mother, who has always believed in me and encouraged me to never settle and to have confidence. I am inspired by her hard work and perseverance and her dedication as a mother. All that and more, helped me get to where I am today.

To my father, for his unwavering support, his loving-kindness and generosity. I would not be here, completing this great work and pursuing my passion without his presence and guidance.

I also acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) for this research and for investing in discovery and innovation in Canada.

General Introduction

The classical and most commonly used approach to building prediction intervals is the parametric approach. However, its main drawback is that its validity and performance highly depend on the assumed functional link between the covariates and the response. The first two chapters of this thesis investigate new and computationally efficient methods that improve the performance of prediction intervals with random forests. The first chapter explores predictions intervals in the classical supervised learning setting, while the second chapter investigates predictions intervals for finite mixture regression models.

In the first chapter two aspects are explored: The splitting rule and the method used to build the prediction interval. In addition to the default least squares splitting rule, two alternative splitting criteria are investigated. We also present and evaluate the performance of five flexible methods for constructing prediction intervals. This yields 15 distinct method combinations. To reliably attain the desired confidence level, we include a calibration procedure performed on the out-of-bag information provided by the forest. The 15 method combinations are thoroughly investigated, and compared to four alternative methods through simulation studies and in real data settings. The results show that the proposed methods are very competitive as they outperform commonly used methods.

In the second chapter, we propose an extension of these methods to finite mixture regression models, which are a flexible tool for statistical modeling of data that originate from heterogeneous populations. The classical mixture regression model relies on mixtures of linear models. We propose a nonparametric regression method for finite mixture models that captures nonlinear dependencies links between the response and the covariates. To achieve this, we use the structure of the classical EM algorithm but substitute the traditionally used linear models with random forests. We propose five variations to build prediction intervals and a calibration method that ensures the intervals reach the desired coverage level. The performance of the methods are assessed in an extensive simulation study. The results show that the calibration method is very reliable in attaining the prescribed coverage, even with small samples. The results also show that the new methods produce short prediction intervals compared to existing methods in a variety of situations involving linear and nonlinear dependencies, small and large sample sizes, and even with heteroscedastic data. Moreover, four out of the five methods also show very good computational efficiency.

The third chapter proposes a method that belongs to another area of computational statistics, variable screening for high-dimensional data. We propose a robust variable screening method for large high-dimensional data. The method is based on the very fast and robust variable selection technique by Dupuis and Victoria-Feser [1]. To produce a variable screening method, we combine this technique in an iterative ensemble scheme where we proceed to a safe and gradual elimination of covariates. This strategy insures the survival of the true model covariates through the procedure and into the final subset. The performance of the method is assessed in a simulation study with both clean and contaminated data and includes scenarios with a number of covariates up to 100,000. It is also tested on a real data set. The results show very good and consistent performance on both clean and contaminated data even with a very large number of covariates. Furthermore, the method offers built-in automatic final subset size determination. This constitutes an innovative feature for a variable screening method.

Chapter 1

Prediction Intervals with Random Forests

Marie-Hélène Roy & Denis Larocque

Abstract

The classical and most commonly used approach to building prediction intervals is the parametric approach. However, its main drawback is that its validity and performance highly depend on the assumed functional link between the covariates and the response. This research investigates new and computationally efficient methods that improve the performance of prediction intervals with random forests. Two aspects are explored: The splitting rule and the method used to build the prediction interval. In addition to the default least squares splitting rule, two alternative splitting criteria are investigated. We also present and evaluate the performance of five flexible methods for constructing prediction intervals. This yields 15 distinct method combinations. To reliably attain the desired confidence level, we include a calibration procedure performed on the out-of-bag information provided by the forest. The 15 method combinations are thoroughly investigated, and compared to four alternative methods through simulation studies and in real data settings. The results show that the proposed methods are very competitive. They outperform commonly used methods in both in simulation settings and with real data.

1.1 Introduction

We assume the regression model with a continuous (or a variable treated as such) response variable Y and a p-dimensional vector $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)$ of covariates

$$E(Y|\boldsymbol{X} = \boldsymbol{x}) = g(\boldsymbol{X})$$

where g is an unknown function. The goal of predictive models is to provide a point prediction $\hat{y}_{new}$ for a new observation Y_{new} with covariates $\boldsymbol{X} = \boldsymbol{x}_{new}$. However, the precision of the prediction is also highly relevant and one way to quantify it is to use a prediction interval (PI) $PI(\boldsymbol{x}_{new}) = [a(\boldsymbol{x}_{new}), b(\boldsymbol{x}_{new})]$, whose magnitude provides information about the reliability of the point prediction. To be clear, we focus on building a prediction interval for $Y|\boldsymbol{X} = \boldsymbol{x}_{new}$ rather than a confidence interval for the conditional mean $E(Y|\boldsymbol{X} = \boldsymbol{x}_{new})$. A confidence interval provides information about the likely location of the true population mean. A prediction interval, on the other hand, gives a range of possible values for an unobserved instance. Not only do we need to take into account the uncertainty in the estimation of $E(Y|\boldsymbol{X} = \boldsymbol{x}_{new})$, as confidence intervals do, but also to take into consideration the variance associated with the new Y itself. Hence, a prediction interval is wider than its corresponding confidence interval.

The typical way to construct prediction intervals is the parametric approach, under an assumed linear regression model which can be described as

$$E(Y|\boldsymbol{X} = \boldsymbol{x}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p, \quad \text{and} \quad V(Y|\boldsymbol{X} = \boldsymbol{x}) = \sigma^2.$$

In this case, the parameters can be estimated by least-squares using a sample of n observations $(y_1, \boldsymbol{x}_1), \ldots (y_n, \boldsymbol{x}_n)$, and a PI with $(1-\alpha)100\%$ coverage probability has the form

$$PI_{\text{reg}}(\boldsymbol{x}_{new}) = \left[\hat{y}_{new} \pm t_{1-\alpha/2,(n-p-1)} \sqrt{\hat{V}(y_{new} - \hat{y}_{new})}\right], \tag{1.1}$$

where $\hat{y}_{new} = \hat{\beta}_0 + \hat{\beta}_1 x_{1new} + \ldots + \hat{\beta}_p x_{pnew}$ is the point prediction, $t_{\alpha,d}$ is the α quantile of the t-distribution with d degrees of freedom, $\hat{V}(y_{new} - \hat{y}_{new})$ is an estimation of the variance of the prediction error

$$V(y_{new} - \hat{y}_{new}) = \sigma^2 + \sigma^2 \boldsymbol{x}_{new}^T (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{x}_{new},$$

and $\boldsymbol{X}$ is the $n \times (p+1)$ design matrix

$$\begin{pmatrix} 1 & x_1^T \\ \vdots & \vdots \\ 1 & x_n^T \end{pmatrix}.$$

To build the PI, $\hat{V}(y_{new} - \hat{y}_{new})$ replaces the sole unknown quantity σ^2 by its unbiased estimate, the MSE, given by $\frac{1}{n-p-1}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, where $\hat{y}_i$ is the prediction for the i^{th} observation.

This is the classical and most commonly used approach to building prediction intervals. It is a quick, easy way to provide more information than with only point predictions. However, its main drawback is that its validity and performance highly depends on the assumed functional link between the covariates and the response. This is why using more flexible methods to compute prediction intervals can be useful. The random forest (RF) algorithm [4] is a well known and powerful non-parametric statistical learning method. The traditional regression random forest can be described by a simple algorithm.

For $b = 1, \ldots, B$

1. Draw a boostrap sample of the original data

2. Grow a regression tree (usually a large unpruned tree) using the bootstrap data with the following modification. At each node, rather than choosing the best split among all predictors, randomly sample p_0 $(0 < p_0 \leq p)$ of the p predictors and choose the best split among those variables.

3. Compute the predictions. For a new observation, the prediction from the tree, $\hat{y}_{b,new}$, is the average of the observations that are in the same terminal node as $\boldsymbol{x}_{new}$.

The final RF prediction for $\boldsymbol{x}_{new}$ is the average of the individual tree predictions

$$\hat{y}_{RF,new} = \frac{1}{B}\sum_{b=1}^{B}\hat{y}_{b,new}.$$

A limited amount of methods to build PI with RF algorithms have appeared so far. These methods take advantage of the extra information given by the forests to construct the intervals. One of them is the Quantile Regression Forests (QRF), proposed by Meinshausen [15]. Instead of estimating the conditional mean, the aim of QRF is to estimate the conditional quantiles

$$q_\tau(\boldsymbol{x}_{new}) = Q_\tau(Y|X = \boldsymbol{x}_{new}) = \inf\{y : F_{Y|\boldsymbol{X}}(y|\boldsymbol{x}_{new}) \geq \tau\}, \tag{1.2}$$

for a given $0 < \tau < 1$, where $F_{Y|\boldsymbol{X}}$ is the conditional cumulative distribution function (cdf) of Y. The main idea of Meinshausen [15] is to use the “nearest neighbor forest weights" (name coined by Lin and Jeon [14]) that were first used by Hothorn et al. [7]. These weights come from the fact that a prediction from a RF is a weighted average of the observations, that is

$$\hat{y}_{RF,new} = \sum_{i=1}^{n} \hat{w}_i(\boldsymbol{x}_{new}) y_i.$$

This is easy to see since the RF prediction is the average of the predictions of all tree and each of them is the average of the observations that are in the terminal node where $\boldsymbol{x}_{new}$ ends up. From this point of view, a RF is a way to generate data-driven weights. The QRF approach estimates the quantiles using an estimated cdf obtained with the nearest neighbor forest weights. More precisely,

$$\hat{F}(y|X = \boldsymbol{x}_{new}) = \sum_{i=1}^{n} \hat{w}_i(\boldsymbol{x}_{new}) I(y_i \leq y),$$

where I is the indicator function. The estimated quantiles are then

$$\hat{q}_\tau(\boldsymbol{x}_{new}) = \inf\{y : \hat{F}_{Y|\boldsymbol{X}}(y|\boldsymbol{x}_{new}) \geq \tau\}.$$

Finally, the PI can be extracted directly from the estimated quantiles as

$$PI_{\text{QRF}}(\boldsymbol{x}_{new}) = [\hat{q}_{\alpha/2}(\boldsymbol{x}_{new}), \hat{q}_{1-\alpha/2}(\boldsymbol{x}_{new})].$$

This PI uses the data adaptively in two ways: 1) the RF lets the data find the link between the covariates and the response automatically, and 2) the PI itself is more flexible since the quantiles produce a PI not necessarily symmetric around the point prediction.

Lei et al. [12] proposed another interesting and useful methodology for building PIs without requiring distribution assumptions, using the conformal inference approach introduced by Vovk et al. [23]. Their approach is very general and can be used with any predictive model, including a RF. The basic idea is to fit the model with an augmented data set that includes the new observation $(\boldsymbol{x}_{new}, z)$, then use a test of the hypothesis of $Y_{new} = z$, and finally let z vary to build a PI. More precisely, let $\hat{y}_z$ be the prediction when $(y_1, \boldsymbol{x}_1), \ldots (y_n, \boldsymbol{x}_n), (z, \boldsymbol{x}_{new})$ is used as the augmented training data. Define the absolute residuals of this augmented data as

$$R_{z,i} = |y_i - \hat{y}_i|, \quad i = 1, \ldots, n, \quad \text{and} \quad R_{z,n+1} = |z - \hat{y}_z|.$$

Define

$$\pi(z) = \frac{1}{n+1} \left(\sum_{i=1}^{n} I(R_{z,i} \leq R_{z,n+1}) + 1 \right).$$

Hence, $(n+1)\pi(z)$ is the rank of $R_{z,n+1}$ among the $n+1$ absolute residuals. The conformal PI is given by

$$PI_{\text{conf}}(\boldsymbol{x}_{new}) = \{z : (n+1)\pi(z) \leq \lceil (1-\alpha)(n+1) \rceil \},$$

where $\lceil \cdot \rceil$ is the ceiling function. This method is very computationally intensive since, in principle, the model must be fitted many times for each new observation. This is why Lei et al. [12] also propose two alternative methods for practical use. The first one, the split conformal prediction, is the fastest and works by splitting the data to separate the fitting and ranking steps. The second one, jackknife prediction, lies between the first two in terms of speed.

In a closely related problem, some authors have proposed methods to estimate the variance of a RF prediction, $V(\hat{y}_{new})$. For instance, Sexton and Laake [22] and Wager et al. [24] use different versions of the jackknife and bootstrap while Mentch and Hooker [17] use the U-statistics framework with trees built on subsamples. With such an estimate, say $\hat{V}(\hat{y}_{new})$, we can build a confidence interval (CI) for the conditional mean $E(Y|\boldsymbol{X} = \boldsymbol{x}_{new})$ with

$$CI(\boldsymbol{x}_{new}) = \left[\hat{y}_{new} \pm z_{\alpha/2} \sqrt{\hat{V}(\hat{y}_{new})} \right]. \tag{1.3}$$

But it is also possible to use the variance estimates to derive a PI. One such way will be investigated in this paper.

This research investigates new potential avenues to improve the performance of prediction intervals with random forests. Two aspects are explored. Firstly, we look at alternatives to the default least squares (LS) splitting rule in the tree construction itself. Two alternative splitting criteria will be evaluated for their impact on the performance of the resulting PIs. Secondly, we present and evaluate the performance of five methods for constructing prediction intervals. Hence, the three splitting criteria (the default LS criterion plus the two new criteria) for the tree construction and the five PI methods yield 15 distinct method combinations. The performance of these 15 combinations will be thoroughly investigated through simulation studies and in real data settings.

The paper is organized as follows. Section 2 presents the methodology, the three random forest splitting criteria and the five prediction interval methods. In Section 3, we present the results from an extensive simulation study with six Data Generating Processes (DGPs) where the proposed methods are compared to four competitors. Section 4 presents the performances when the same methods are applied to seven real data sets. Section 6 concludes the paper with a short discussion.

1.2 Methodology

The proposed methods are all using an idea, already employed in Moradian et al. [18] and Moradian et al. [19], very similar to the nearest neighbor forest weights. Assume we have built a forest of B trees with any tree growing algorithm. For a new observation $\boldsymbol{x}_{new}$, we define the "Bag of Observations for Prediction" (BOP) to be the pooled set of training observations that are in the same terminal nodes as $\boldsymbol{x}_{new}$ in the forest. More precisely, let $S_b(\boldsymbol{x}_{new})$ be the training responses that are in the same terminal node as $\boldsymbol{x}_{new}$ for the b^{th} tree. Note that an observation can be present more than once in $S_b(\boldsymbol{x}_{new})$ because of the bootstrapping. The BOP for $\boldsymbol{x}_{new}$ is

$$\mathrm{BOP}(\boldsymbol{x}_{new}) = \bigcup_{b=1}^{B} S_b(\boldsymbol{x}_{new}).$$

For example, suppose that $n = 5$ and $B = 2$, that the terminal nodes where $\boldsymbol{x}_{new}$ falls have the training responses $S_1 = \{y_1, y_1, y_2\}$, and $S_2 = \{y_1, y_2, y_2, y_3\}$. Then $\mathrm{BOP}(\boldsymbol{x}_{new}) = \{y_1, y_1, y_1, y_2, y_2, y_2, y_3\}$, whereas the vector of nearest neighbor forest weights for $\boldsymbol{x}_{new}$ is $(11/24, 10/24, 3/24, 0, 0)$. If all terminals nodes have the same number of observations, then the proportion of points in the BOP is the same as the nearest neighbor forest weights. The idea is to use $\mathrm{BOP}(\boldsymbol{x}_{new})$ to compute any desired summary for this new observation. The advantage of the BOP over a weight vector is that it is possible to compute any quantity directly without needing a "weighted" version of it. Obviously, we could build a BOP with the weights but they would typically involve more observations (24 instead of 7 in the above example) and would require more computation time. With a large number of trees involving terminal nodes with similar terminal node sizes, these two approaches are practically equivalent.

The main methodology proposed in this paper is to

1. Build a forest and get the BOPs for the new observations.

2. Compute the PIs using these BOPs.

3. Calibrate the PIs using the Out-Of-Bag (OOB) information.

The original regression RF is built with the CART method and thus uses the least-squares splitting rule. However, it might be the case that using another splitting rule, more in line with the final goal which is to build a PI, might be preferable. Hence, three ways of performing step 1) will be investigated. Moreover, there are many ways to use a BOP to build a PI in step 2) and five of them will be investigated. This general method is in line with the modern view that sees RF as a weight generating machine, and that the splitting rule should be designed according to the specific problem at hand. For example, to estimate the survival function in the context of dependent censoring, this view was used in Moradian et al. [19] with a specific splitting rule (L_1 splitting, see below) and by computing an estimate valid under dependent censoring (copula-graphic) with the BOP. This general method is now starting to be formalized into general frameworks in the interesting work of Athey et al. [1] and Hothorn and Zeileis [8].

1.2.1 Splitting Rules

The first step of the method is to build a forest. Any algorithm can do the job in principle but the CART paradigm will be used in this paper. Assume that we want to split a node. Let S_L and S_R be the set of observations that are in the left and right nodes after the split, with respective sizes n_L and n_R.

Least-squares splitting rule (LS)

The default splitting rule for a CART regression tree is the least squares (LS) criterion which will be the first one we consider in this study. The best split is the one that minimizes

$$\sum_{S_L}(y_i-\bar{y}_L)^2+\sum_{S_R}(y_i-\bar{y}_R)^2, \tag{1.4}$$

where $\bar{y}_L$ and $\bar{y}_R$ are the averages of the observations in the left and right node, respectively.

The LS criterion is certainly a sensible choice when the goal is to predict a new observation with the L_2 loss function under the assumption of homoscedastic error, but it is not obvious that it is the best one if the final goal is to build a PI. This is why two additional splitting rules will be investigated.

L_1 splitting rule

The first alternative is the L_1 splitting rule. This splitting rule was used in Moradian et al. [18] and Moradian et al. [19] with censored survival data with the ultimate goal of estimating the survival function. The best split is the one that maximizes

$$n_L n_R \int |\hat{F}_L - \hat{F}_R|, \tag{1.5}$$

where $\hat{F}_L$ ($\hat{F}_R$) is the empirical cdf of the left (right) node. The idea is that a splitting rule that uses the whole conditional distribution, and not only the conditional mean, might provide better information to build a PI. Note that in Moradian et al. [18], the Kaplan-Meier estimate is used instead of the empirical cdf because of the censoring.

Shortest prediction interval (SPI) splitting rule

Even though it uses the whole distribution, the L_1 splitting rule is still not aimed directly at the task at hand, building a PI. Hence, the last splitting rule, that we call the shortest prediction interval (SPI) splitting rule, is designed with that task in mind. Let $D = \{z_1, \dots, z_m\}$ be a one variable data set and let $z_{(1)} \leq \dots \leq z_{(m)}$ denote the order statistics. For a given value of $0 < \alpha < 1$, we define $\text{SPI}_\alpha(D)$ to be the shortest interval of the form $[z_{(l)}, z_{(u)}]$, with $l \leq u$, that contains at least $(1-\alpha)100\%$ of the observations. The SPI splitting rule is the one such that the best split minimizes

$$n_L \text{length}(\text{SPI}_\alpha(S_L)) + n_R \text{length}(\text{SPI}_\alpha(S_R)). \tag{1.6}$$

The idea is to seek splits that produce compact nodes, with the hope that the resulting PIs will also be short. Contrarily to the first two splitting rules, this one has a tuning parameter, α. It could be set to the same value as the desired coverage for the final PI. Alternatively, it could be estimated from the data.

1.2.2 Methods for Building a Prediction Interval

The second aspect explored is the method to construct the PI for a given $\boldsymbol{x}_{new}$, that is $PI(\boldsymbol{x}_{new})$. We propose five alternatives. All of them are computed using BOP$(\boldsymbol{x}_{new})$.

Classical (LM)

The first method is the simplest one. We compute the classical PI, in an intercept only linear model (no covariates) using the BOP$(\boldsymbol{x}_{new})$ as the sample. This produces a PI symmetric around the mean of the BOP$(\boldsymbol{x}_{new})$.

Quantile

The second method is based on the quantiles, like the QRF method. Like above, let D be a one variable data set and define $q_\alpha(D)$ to be a α-quantile of D. The PI is $[q_{\alpha/2}(\text{BOP}(\boldsymbol{x}_{new})), q_{1-\alpha/2}(\text{BOP}(\boldsymbol{x}_{new}))]$. This PI is not necessarily symmetric around the mean of the BOP$(\boldsymbol{x}_{new})$.

Shortest prediction interval (SPI)

The third one is derived from the splitting rule with the same name described in the previous section. The PI is SPI$_\alpha(\text{BOP}(\boldsymbol{x}_{new}))$, that is, the shortest interval formed by two observations in BOP$(\boldsymbol{x}_{new})$ that contains at least $(1-\alpha)100\%$ of the observations.

In a sense, the first three PI construction methods are the natural counterparts to the three splitting rules described previously. It is natural to use the classical PI when the forest is built with the least-squares splitting rule, and it is also natural to use the SPI to build the PI if we used the SPI splitting rule. Although less direct, it is also natural to use quantiles to build the PI if the forest is built with the L_1 splitting rule, which aims at estimating the whole cdf.

Highest density region (HDR)

The SPI is a quick and easy way to aim at the shortest possible PI. An alternative way is to use the highest density region (HDR); Hyndman [9]. For completeness, the definition of a HDR from Hyndman [9] is reproduced here.

Definition. Let $f(x)$ be the density function of a random variable X. The $(1-\alpha)100\%$ HDR is the subset $R(f_\alpha)$ of the sample space of X such that $R(f_\alpha) = \{x : f(x) \geq f_\alpha\}$ where f_α is the largest constant such that $Pr(X \in R(f_\alpha)) \geq 1-\alpha$.

The HDR is the smallest region, with the desired coverage $(1-\alpha)100\%$, such that the density of every point inside it is at least as large as the density of every point outside it. For a sample, the HDR can be estimated using density estimators (like the kernel); Hyndman [9] and Samworth and Wand [21].

Let $\widehat{\text{HDR}}_\alpha(D)$ be an estimation of the HDR, for a given α, for a one variable data set D. With this method, the PI is $\widehat{\text{HDR}}_\alpha(\text{BOP}(\boldsymbol{x}_{new}))$.

The HDR is not necessarily a single interval. It can be formed by multiple intervals, in the case of a multimodal distribution. The estimated HDR can also be formed by multiple intervals. This can arise if the true density is multimodal or simply by variability if, for instance, a too small bandwidth is used with a kernel density estimate. In some applications, having a PI formed by more than one interval might be acceptable but sometimes a single interval might be preferred. This is why we also consider the following fifth and last method.

Contiguous highest density region (CHDR)

This method is a simple modification of the previous one. Compute $\widehat{\text{HDR}}_\alpha(\text{BOP}(\boldsymbol{x}_{new}))$. The CHDR PI is $[\min(\widehat{\text{HDR}}_\alpha(\text{BOP}(\boldsymbol{x}_{new}))), \max(\widehat{\text{HDR}}_\alpha(\text{BOP}(\boldsymbol{x}_{new})))]$. If $\widehat{\text{HDR}}_\alpha(\text{BOP}(\boldsymbol{x}_{new}))$ is already a single interval, then the CHDR PI is the same. Otherwise, the CHDR fills the gaps to get a single interval. This might seem suboptimal at first but, with the calibration method described in the next section, it performs quite well.

In principle, we could design a splitting rule around the HDR (or the CHDR). For instance, we could replace the SPI by a HDR. However, the computation cost of a HDR is too high to be used in practice in a tree building algorithm, where it has to be computed numerous times.

1.2.3 Calibration

If we apply the PI building methods described in the previous section using the desired coverage level (for example $1-\alpha = .95$), then the resulting PIs tend to be conservative, that is they are too long with an actual coverage higher than what we aim at. In order to reliably attain the desired $1-\alpha$ level, we include a calibration procedure. This calibration uses the BOPs that are collected from the predictions of the OOB observations for all trees. The OOB BOPs are created similarly to the BOPs for a new observation. The OOB BOP for the i^{th} training observation $\boldsymbol{x}_i$ is

$$\text{BOP}_{\text{oob}}(\boldsymbol{x}_i) = \bigcup_{b=1, i \in \text{OOB}_b}^{B} S_b(\boldsymbol{x}_i),$$

where $S_b(\boldsymbol{x}_i)$ is the set of training responses that are in the same terminal node as $\boldsymbol{x}_i$, and OOB_b is the set of OOB observations, for the b^{th} tree. Note that $\text{BOP}_{\text{oob}}(\boldsymbol{x}_i)$ is built using only the trees where $\boldsymbol{x}_i$ is OOB. This is in contrast with $\text{BOP}(\boldsymbol{x}_{new})$ which uses all trees. Hence we have a BOP for each training observations.

The calibration is done through the parameter $1-\alpha$. Let $1-\alpha$ be the target coverage level and let α_w be the “working" value of α. The idea is to find the value of α_w such that the coverage level computed with the training observations, using the $\text{BOP}_{\text{oob}}(\boldsymbol{x}_i)$, is close enough to the target level. Once found, $1-\alpha_w$ becomes the level used to build the PI for the new observations. To fix ideas, in the following simulations, the target $1-\alpha$ is .95 and α_w is selected such that the OOB coverage level lies in the interval $[.94, .95]$.

1.2.4 Implementation

The R software [20] was used to implement the proposed methods. The RF are built with the function `rfsrc` in the package `randomForestSRC` [11]. The least-squares splitting rule is the default one for a continuous response. Moreover, the function also allows to use custom splitting rules. Hence, the L_1 and SPI splitting rules were coded in C and incorporated in `rfsrc` for this work. The classical PI can easily be obtained with a `predict` on a `lm` fit without covariates. For the HDR and CHDR methods the CRAN package `hdrcde` [10] is used. The functions for the Quantile and SPI approaches were coded in R.

1.3 Simulation Study

To assess the performance of the 15 proposed method combinations (three splitting criteria $\times$ five PI approaches) we carry out a simulation study using six Data Generating Processes (DGPs).

1.3.1 Simulation Design

For the first two DGPs (DGP1 and DGP2), the vector of independent predictors $X = (X_1, ..., X_7)$ is generated such that each X_i is from the standard normal distribution. The response is generated according to a depth 3 tree model, with 8 terminal nodes:

$$\begin{aligned} Y = u_1(I(X_1 < 0, X_2 < 0, X_4 < 0) \\ +u_2(I(X_1 < 0, X_2 < 0, X_4 >= 0) \\ +u_3(I(X_1 < 0, X_2 >= 0, X_5 < 0) \\ +u_4(I(X_1 < 0, X_2 >= 0, X_5 >= 0) \\ +u_5(I(X_1 >= 0, X_3 < 0, X_6 < 0) \\ +u_6(I(X_1 >= 0, X_3 < 0, X_6 >= 0) \\ +u_7(I(X_1 >= 0, X_3 >= 0, X_7 < 0) \\ +u_8(I(X_1 >= 0, X_3 >= 0, X_7 >= 0) + \varepsilon, \end{aligned}$$

where the terminal nodes mean are $\boldsymbol{u} = (5, 10, 15, 20, 25, 30, 35, 40)$. In DGP1, ε is generated from a standard normal distribution and in DGP2 from an exponential distribution with mean 1. The reason to use the exponential distribution is to investigate the performance with an asymmetric error distribution. This might have an impact because some methods build symmetric PIs and other built (possibly) asymmetric PIs based on quantiles.

The other four DGPs are generated using functions from the CRAN `mlbench` package [13]. DGP3, DGP4 and DGP5 are variations of Friedman's benchmark problems.

DGP3 is Friedman problem 1, a regression problem described in Friedman [5] and Breiman [3]. The inputs are ten independent variables uniformly distributed on the interval [0,1]. Five out of these ten predictors are used to generate the response:

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \varepsilon,$$

where ε is $N(0, \sigma)$. The standard deviation of ε is left at the default value of 1.

Data for DGP4 is generated following Friedman problem 2, also found in Friedman [5] and Breiman [3]. For this regression problem, the inputs are four independent variables uniformly distributed over the ranges

$$\begin{aligned} 0 \le x_1 &\le 100 \\ 40\pi \le x_2 &\le 560\pi \\ 0 \le x_3 &\le 1 \\ 1 \le x_4 &\le 11. \end{aligned}$$

The response is generated with

$$y = (x_1^2 + \left(x_2 x_3 - \left(\frac{1}{x_2 x_4} \right)^2 \right)^{0.5} + \varepsilon$$

where ε is $N(0, \sigma)$. It is used with the default value of 125 for the standard deviation of noise. This yields a signal to noise ratio of 3:1.

Data for DGP5 is generated with Friedman problem 3 [5, 3]. For this regression problem, the inputs are also four independent variables uniformly distributed over the same ranges as Friedman problem 2. For this problem the response is generated with

$$y = \tan^{-1} \left(\frac{x_2 x_3 - \left(\frac{1}{x_2 x_4} \right)}{x_1} \right) + \varepsilon$$

where ε is $N(0, \sigma)$. It is used with the default value of 0.1 for the standard deviation of noise. This yields a signal to noise ratio of 3:1.

DGP6 is generated using the Peak Benchmark Problem also from the `mlbench` package. The description goes as follows: Let $r = 3u$ where u is uniform on $[0, 1]$. Let x be uniformly distributed on the d-dimensional sphere of radius r. The response is $y = 25\exp(-.5r^2)$. We have left the default value of $d = 20$ dimensions.

The sample sizes for the training set for each of the six DGPs is equal to 150, 500 and 1000, resulting in 18 scenarios. In each case a test set of size 1000 observations was generated. For all scenarios the results are averaged over 200 simulations.

1.3.2 Competing Methods

We compare the predictive performance of the prediction intervals supplied by our methods to four competitors which were all presented in the Introduction. The first one is the QRF method [15]. The CRAN package, `quantregForest` [16] is used.

The second and third competing methods are two versions of the conformal inference method by Lei et al. [12], also discussed in the Introduction. These are the jackknife version (CI-jack) and the splitted conformal inference version (CI-split). The full conformal inference method was excluded due to its very large computational time. We used them exclusively with random forests as base estimators and also kept the default settings. The `conformalInference` package [6] found on GitHub was used.

The fourth competitor is an adaptation of the approach proposed by Wager et al. [24]. We use its associated GitHub package `randomForestCI`. For a new observation, this method provides the point prediction $\hat{y}_{new}$ along with a jackknife estimation of its variance $\hat{V}(\hat{y}_{new})$. These two quantities can be used directly to build a confidence interval through (1.3). However, we want a prediction interval. The way we proceed is by considering intervals of the form

$$PI(\boldsymbol{x}_{new}) = \left[\hat{y}_{new} \pm z_{\alpha/2}\sqrt{\hat{V}(\hat{y}_{new}) + \hat{\sigma}^2}\right],$$

where the value $\hat{\sigma}^2$ is used to calibrate the PIs. We look for a value such that the cross-validated coverage rate is close to 0.95. More precisely, the MSE computed by 5-fold CV is used as the starting value for $\hat{\sigma}^2$. Then we let this value vary until we find a value such that the coverage rate, computed by 5-fold CV, is within the interval $[.940, .950]$. Then this value is used to build the PIs for the new observations. We call this method IF-jack.

1.3.3 Parameters and calibration for the Simulation Study

The target coverage probability is set to .95 for all scenarios and methods. Calibration, as presented in Section 2, is performed for all the proposed method combinations. For calibration, the acceptable range of $1-\alpha$ on the OOB data is $[.940 - .950]$. Because no option for calibration for QRF is offered in its CRAN package, it is not performed for this method. For CI-jack and CI-split, the calibration process is automatically done using five-fold cross validation in their respective functions. For IF-jack we used the calibration process described above and set the target coverage range to [.940 – .950], the same as for the proposed methods.

For all methods, to build the forests, the number of trees was set to 500, the minimum number of observations in the terminal nodes is 1 and all other parameters were left to the package `randomForestSRC` default settings.

In the HDR and CHDR PI methods, a bandwidth has to be chosen. The `hdrcde` package offers an option in the `hdr` function to choose the optimal bandwidth for each BOP. However, estimating the bandwidth separately for all observations in the test set takes a long time. This is why we used the following method to gain computational efficiency. For a given forest, we find the optimal bandwidth for 10 BOPs. The mean of these bandwidths is then used as the bandwidth for all BOPs for this forest.

1.4 Results of the Simulation Study

The detailed results from the simulation study are presented in Table A in the Appendix. Each of the 18 sections (6 DGPs $\times$ 3 sample sizes) of the table provides the mean coverage of the PIs, the mean length of the PIs and their standard deviation on the test set, over the 200 runs, for a given DGP and sample size. The first and foremost important property of a PI is to maintain the prescribed coverage, which was set to .95 throughout this study. All methods were able to do so. Indeed, all the 342 (19 methods $\times$ 6 DGPs $\times$ 3 sample sizes) reported mean (over the 200 runs) coverage are greater or equal to .94, except one which is at .939. Moreover, all methods are able to stay close to .95 except the QRF method which tends to be conservative. The 15 proposed methods have a built-in calibration procedure. The results show that this procedure works well. The 4 competitors were taken as they are. The two conformal inference methods also have a built-in calibration procedure and it was reliable as well. Since the IF-jack method is designed to build a confidence interval, we had to implement a calibration procedure for it to derive predictions intervals, as explained in the last section, and the results show that it also worked well. Only the QRF method does not have a calibration procedure when used “off-the-shelf", and we see that applying it as it is tends to produce conservative PIs.

Once a method is able to produce reliable PIs maintaining the desired coverage, the length of the PIs becomes the performance measure, the shorter the better. Before comparing in details the mean lengths of the PIs for all DGPs and sample sizes, we first provide a global view of the results. To be able to compare the lengths across the different scenarios, we use the percentage increase in PI length of a method with respect to the best performer for a given run. This way, only the relative lengths are used and the results can be aggregated across all scenarios. There are 68,400 runs in all in this simulation study (19 methods $\times$ 6 DGPs $\times$ 3 sample sizes $\times$ 200 runs). For a given run, we have the mean length (on the test set) of the PIs for each method. Let ML_k be the mean length of method k, $k = 1, 2, \ldots, 19$, for this run. Let $\min_k\{\mathrm{ML}_k\}$ be the smallest mean length for this run. The percentage increase in PI length of method k with respect to best performer for this run is defined to be

$$100\frac{(\mathrm{ML}_k - \min_k\{\mathrm{ML}_k\})}{\min_k\{\mathrm{ML}_k\}}.$$

Hence, the smallest is this value, the better is the method. Figure 1.1, shows the distributions of this measure for all 68,400 runs in the simulation study. All methods are there except QRF because its performance was so poor compared to the others that including it would have distorted the graph. But the results for QRF will be shown in the following figures, scenario by scenario. The first 5 box-plots are the ones of the proposed methods when the RF is built with the LS splitting rule. The next 5 box-plots correspond to the RF built with the L_1 splitting rule. The following 5 are the ones with the RF built with the SPI splitting rule. The last three are the competing methods: CI-jack, CI-split, and IF-jack. The results show that, globally, the proposed methods perform well, and are slightly better than the competitors, except when the PIs are built with the quantiles. The HDR method produced the shortest PIs. But this method can produce a PI formed by more than one interval, while all other methods provide a PI formed by a single interval. If this is an issue, the CHDR, SPI and LS methods work very well too, and provide PIs with a single interval. Among the two conformal inference methods, the CI-jack is preferable to CI-split. Also, the performance of CI-jack is very close to the one of IF-jack. It is interesting to see that, globally, using the simplest approach (fifth box-plot from the start) which is to build the RF with the LS splitting rule and using the classical method to build the PIs performs very well.

We now examine the detailed results for each scenario. We are using directly the mean PI length since they are comparable within a scenario. Figures 1.2 to 1.7 show the distribution of the mean length (ML) for each of the 19 methods and 18 scenarios. Each figure corresponds to a DGP.

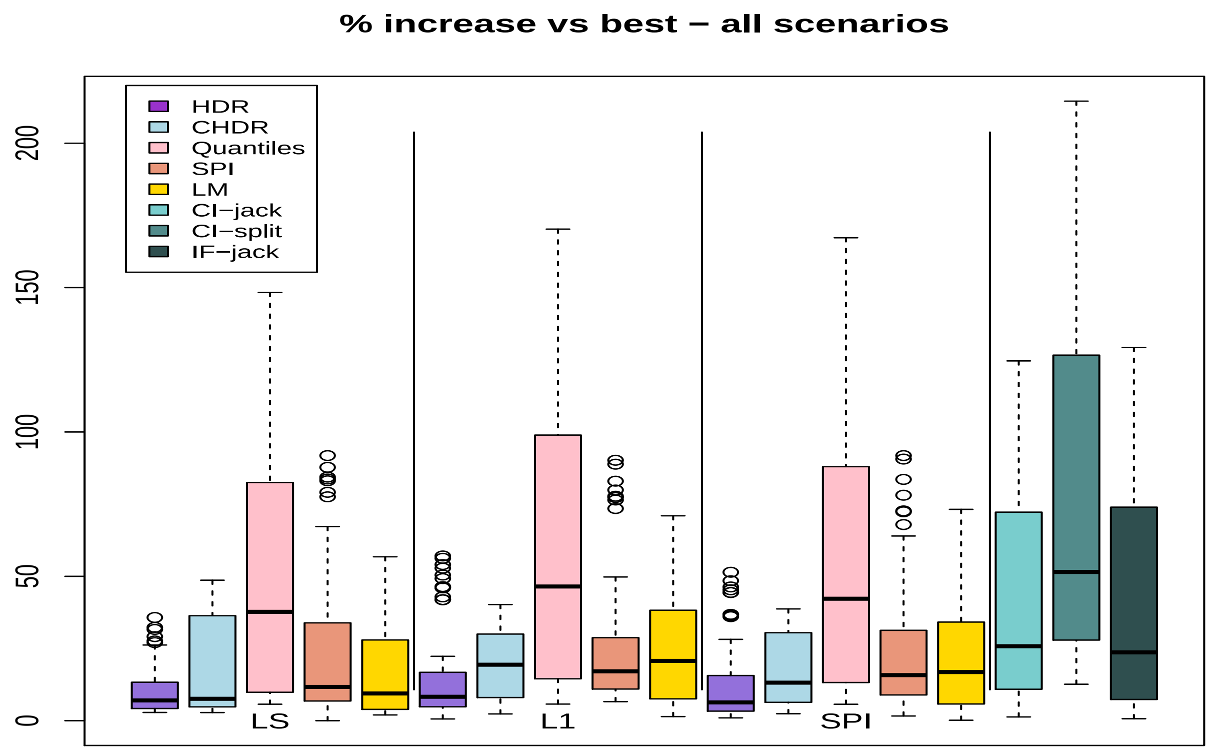


Figure 1.1: Distributions of the percentage increase in PI length of each method with respect to best performer for a given run for all 68,400 runs in the simulation study.

Looking at Figure 1.2, we see that for DGP1, for all sample sizes, 12 out of the 15 proposed methods combinations clearly outperform the four competitors in terms of PI length. When looking at the comparative performance of the three splitting rules, no significant difference is noted. The differences in ML seem to be a result of the choice of PI method used and not the choice of splitting criterion. This conclusion can also be drawn, to different extents, for most of the scenarios. The HDR PI method is distinctly the best approach with DGP1 yielding the shortest, thus more precise PIs. Among the competitors, CI-jack and IF-jack show the best performances, with CI-jack, the most computationally intensive method of the two conformal inference methods in this study, having a slight advantage in terms of median ML. QRF has clearly had the longest MLs. But as we saw this is mainly due to the fact that the PIs of this method are conservative. Also, as could be expected, for all methods and scenarios, the MLs and their variability significantly decrease with the increase in sample size.

In Figure 1.3, the results of DGP2 are very similar to those of DGP1. The only difference between the two are in the ε, it follows a normal distribution in DGP1 and an exponential distribution in DGP2. We wanted to investigate the impact of an asymmetrical error distributions on the symmetrical PI methods, such as LM. We can conclude from the results for DGP2 vs DGP1 that the effect is minimal. LM is only very slightly affected by DGP2's asymmetrical error distribution. Once again, for the proposed methods, we note that it is clearly the choice of PI method rather than splitting criterion that impacts the results.

The performances of the methods with DGP3, in Figure 1.4, are quite different from the first two. The splitting criteria have a noticeable impact on the performance, with LS showing the shortest intervals. In these three scenarios CI-jack and IF-jack show the best overall performances, but most of the proposed methods are not far behind.

For DGP4 and DGP5 (Figures 1.5 and 1.6), generated following Friedman problems 2 and 3, most of the proposed proposed methods achieve shorter MLs than the competing methods on all sample sizes, except for DGP 5 and $n = 1000$, where IF-jack is among the best ones. This time, the PI methods performances are very similar, especially with DGP5. In most scenarios, only the quantiles and SPI have slightly longer MLs. CI-jack and IF-jack yield again the shortest MLs among the competing methods.

In Figure 1.7, that presents the results of the scenarios using DGP6, the discrepancies between all the methods is much more noticeable. This time, the overall best performing methods are the ones using the SPI PI approach followed by CHDR, especially with the LS splitting criterion. The quantile approaches, quantiles PI and QRF, are quite far behind compared to the others MLs, especially as the sample size increases.

When looking globally at the results of the simulation study, we can conclude on three aspects. First, in a majority of scenarios, one or many of the proposed methods yield distinctly better or equal performances in terms of MLs than the competing methods. Second, the choice of splitting criterion, with respect to the three criteria tested in this study, has very little impact on the performance when constructing PIs. Third, it is the choice of the PI construction approach that has the most significant impact on the performance. Furthermore, although HDR is the best overall performing PI method, a simple straightforward approach such as the classical LM provides a very good overall performance. Quantiles is the only PI method that in all tested settings has no advantage in being used over the other proposed PI approaches.

1.5 Performance with Real Datasets

To further explore the performance of the 15 proposed method combinations and their competitors, we test them on 7 real data sets (in fact, 6 data sets but one of them has two response variables). All of them were found on the UCI Machine Learning Repository [2] under the task *regression*. They have sample sizes that range between 103 to 1503 and a number of covariates varying from 5 to 68. The details are presented in Table 1.1.

Five times 10-fold cross-validation was performed for each method on every data set. In each fold, the RF is built with the training data (90% of the data), and the PIs (with a target of $1-\alpha = .95$) are calculated for the validation data (10% of the data). The coverage and mean length (ML) of the PIs were then computed. These results are averaged over the 5 repetitions of 10-fold cross-validation, and are reported in Table 1.1. Figures 8, 9, and 10 present the box-plots of the ML, over the 5 repetitions of 10-fold cross-validation, in graphs similar to those used for the simulation study.

We can see in Table 1.1 that, like for the simulation study, all methods were able to maintain the target coverage. Only in one case did the mean coverage fell below .94, at .936. We can also see

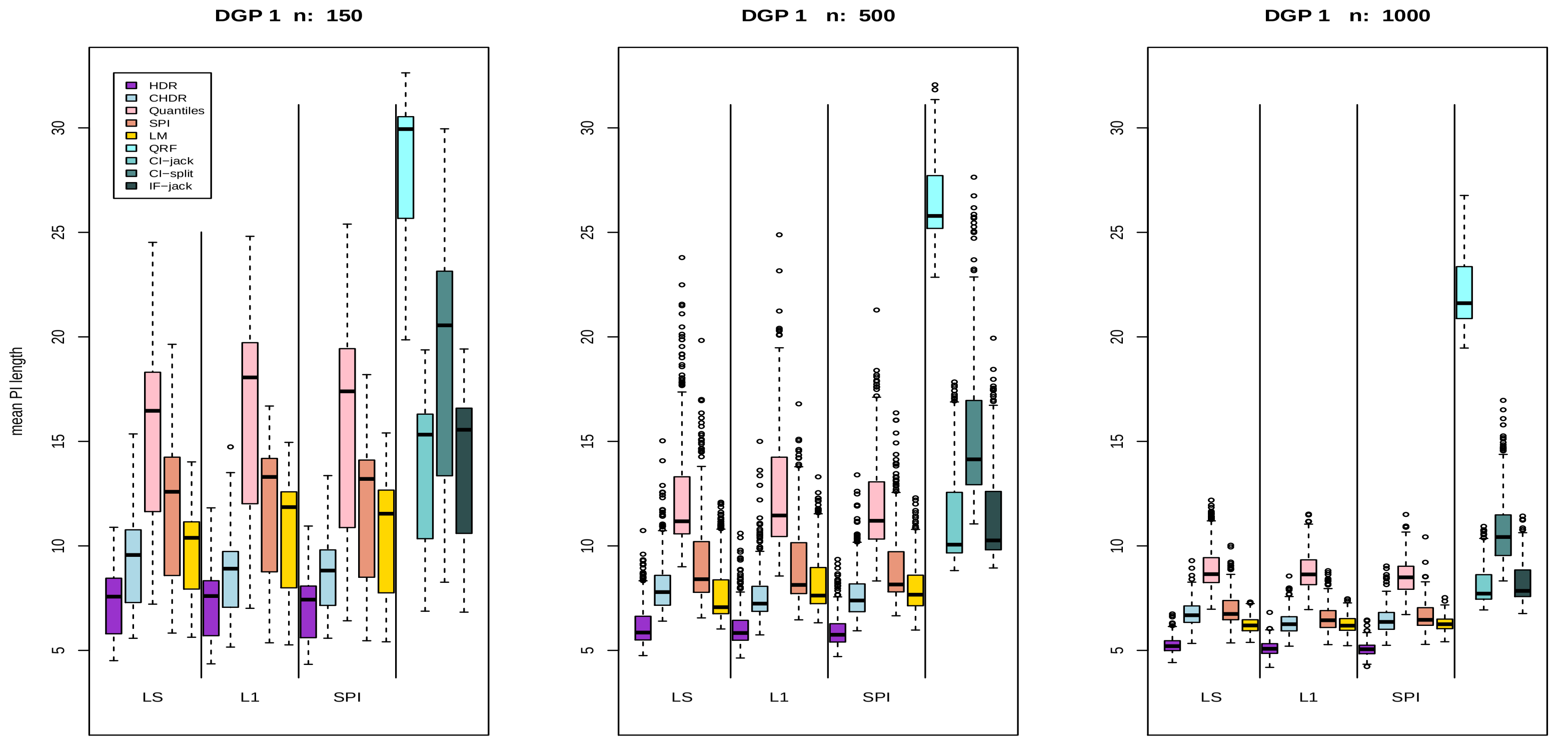


Figure 1.2: Distributions of the PI length for DGP 1.

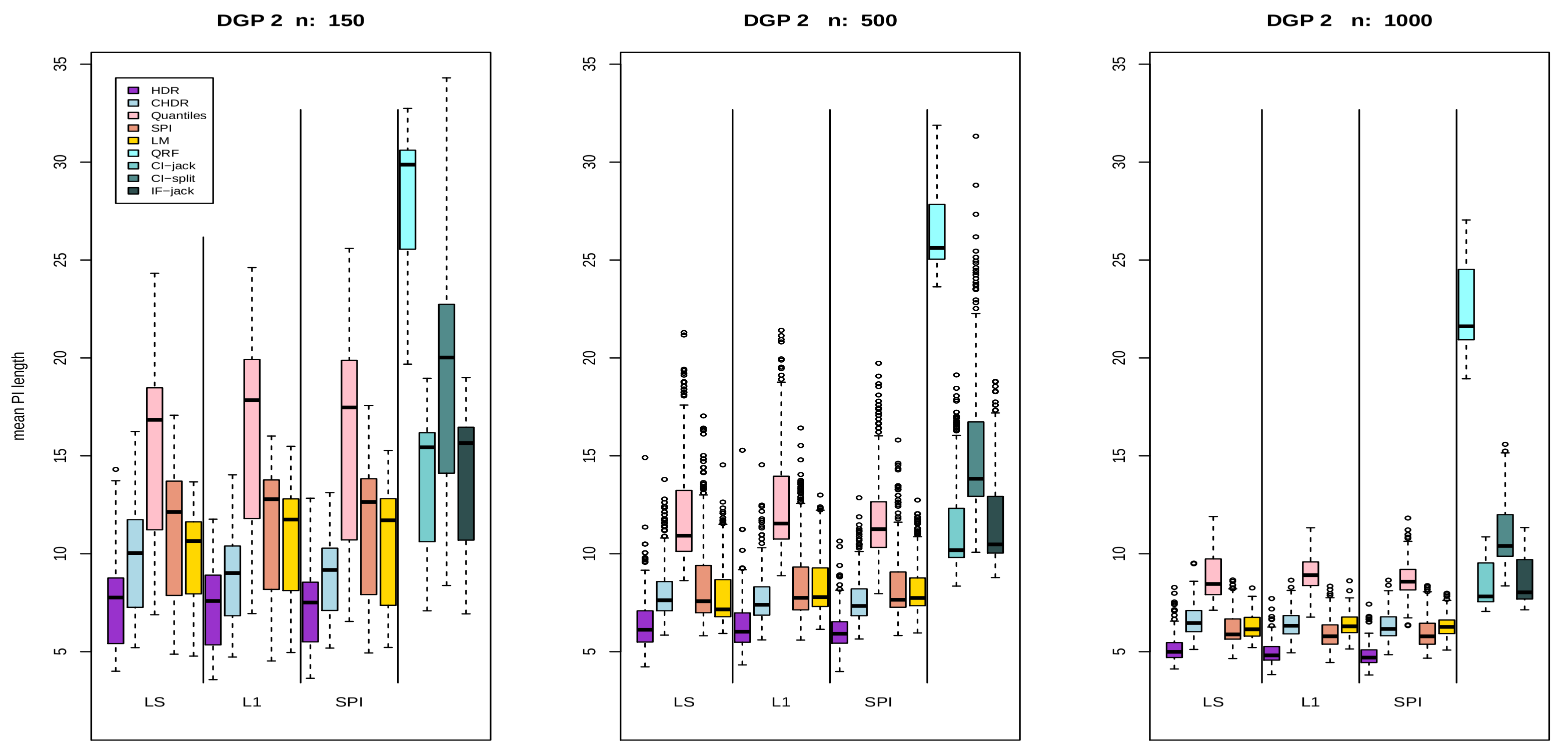


Figure 1.3: Distributions of the PI length for DGP 2.

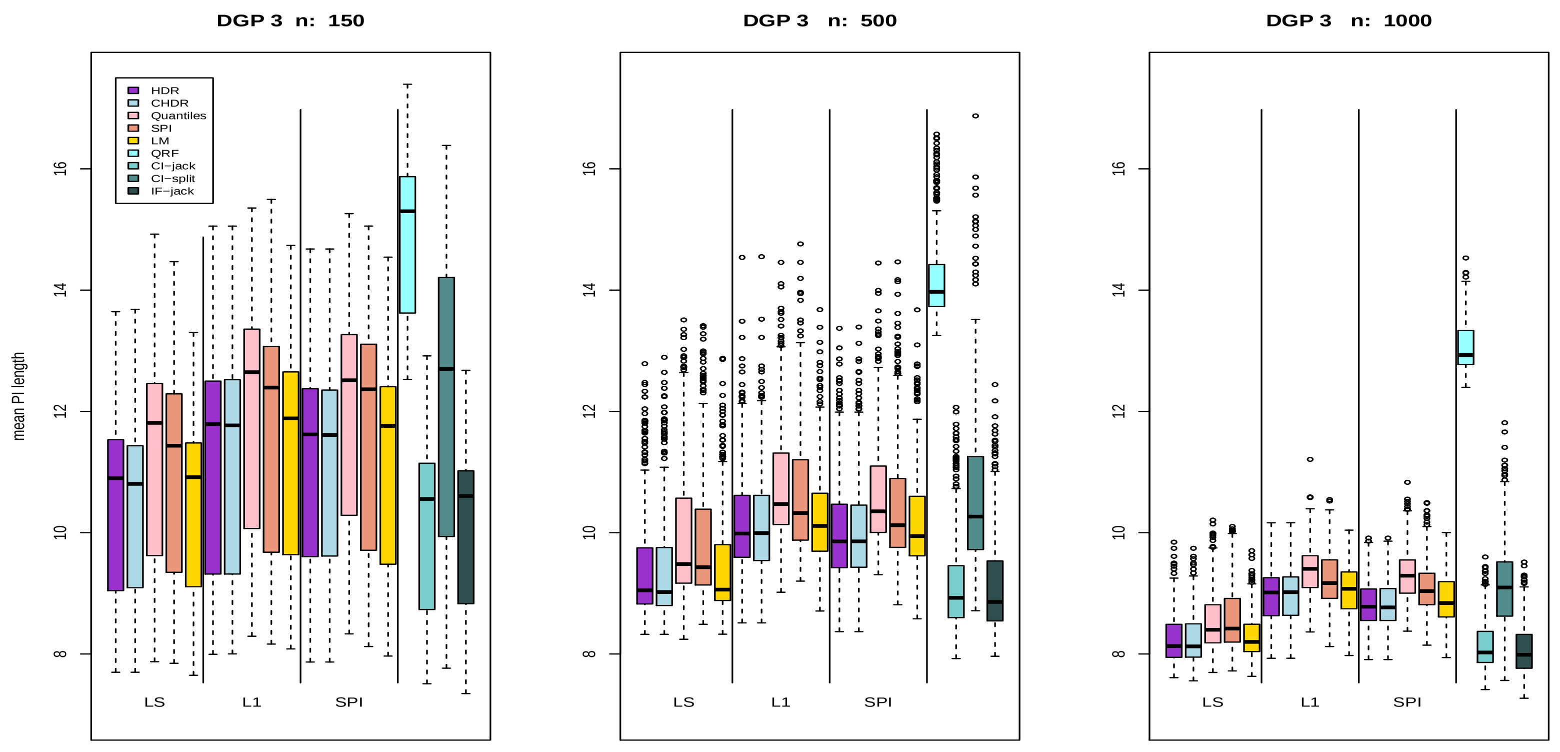


Figure 1.4: Distributions of the PI length for DGP 3.

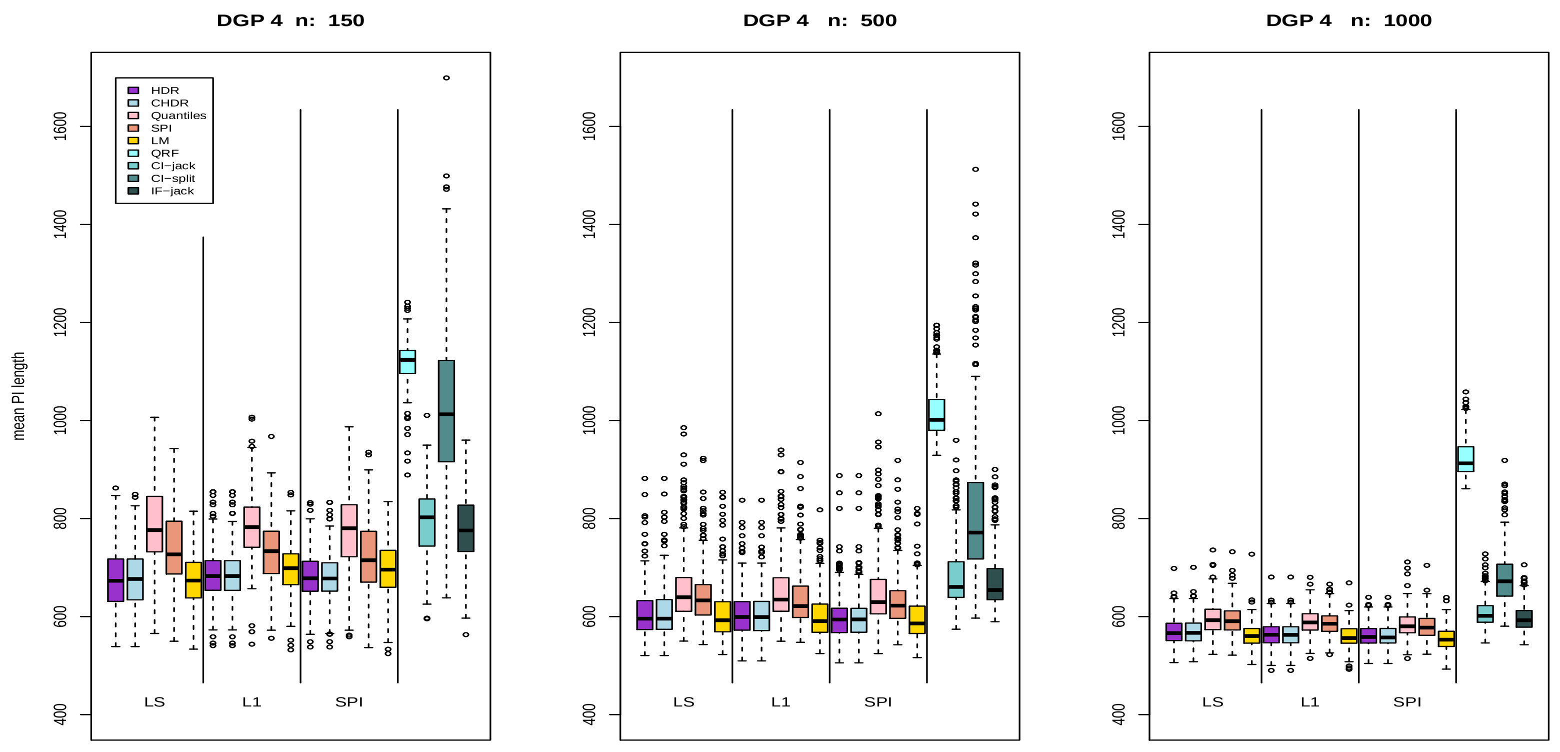


Figure 1.5: Distributions of the PI length for DGP 4.

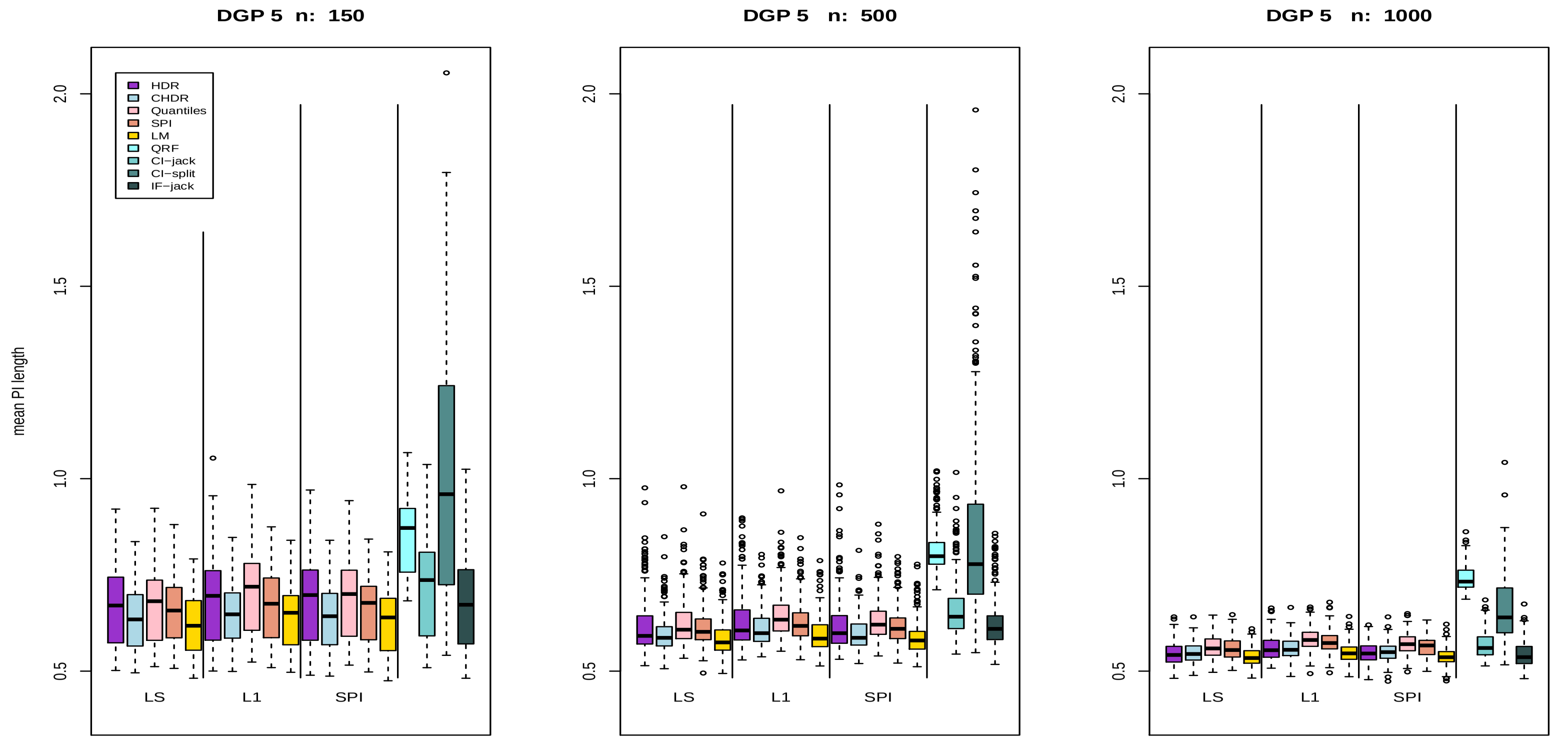


Figure 1.6: Distributions of the PI length for DGP 5.

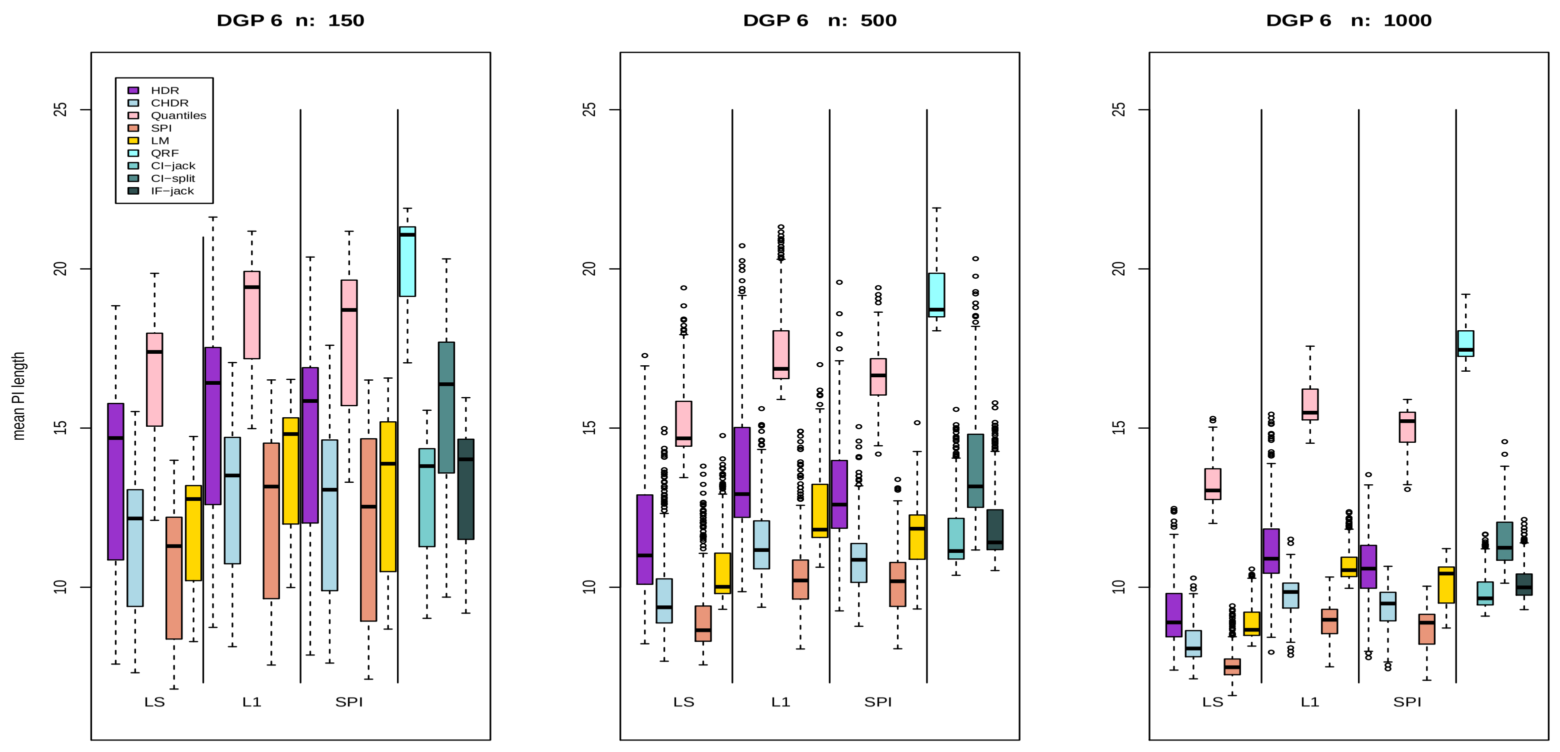


Figure 1.7: Distributions of the PI length for DGP 6.

Table 1.1: Averaged results of the five repetitions of 10-fold cross-validation for the seven real datasets. The column 'cov' shows the mean coverage of the PIs, and 'ml' their mean length. The 15 proposed method combinations are presented in a matrix format where the line (LS, L1, SPI) represent the splitting rule and the column (HDR, CHDR, Quant, SPI, LM) represent the method used to build the interval. The 4 competitors (QRF, CI-jack, CI-split, IF-jack) are presented right under the proposed methods.

			cov	ml	cov	ml	cov	ml	cov	ml	cov	ml
Dataset :	Boston Housing		HDR		CHDR		Quant		SPI		LM	
n :	506	LS	0.950	10.68	0.947	11.09	0.945	10.78	0.947	10.60	0.947	11.15
p :	13	L1	0.951	10.87	0.949	11.34	0.946	11.27	0.946	10.87	0.957	11.85
		SPI	0.947	10.64	0.948	11.00	0.948	10.85	0.946	10.68	0.951	11.36
			QRF		CI-jack		CI-split		IF-jack			
			0.981	15.65	0.952	12.41	0.953	14.73	0.943	11.12		
Dataset :	BWC		HDR		CHDR		Quant		SPI		LM	
n :	198	LS	0.954	0.678	0.950	0.689	0.952	0.735	0.953	0.742	0.951	0.688
p :	29	L1	0.950	0.682	0.952	0.685	0.949	0.736	0.952	0.734	0.952	0.717
		SPI	0.951	0.664	0.951	0.665	0.952	0.705	0.948	0.702	0.950	0.671
			QRF		CI-jack		CI-split		IF-jack			
			0.996	1.981	0.946	0.952	0.953	1.474	0.952	0.953		
Dataset :	Concrete Slump		HDR		CHDR		Quant		SPI		LM	
n :	103	LS	0.961	14.95	0.961	14.82	0.949	18.37	0.942	17.71	0.951	15.60
p :	10	L1	0.964	15.34	0.961	15.11	0.946	17.67	0.948	17.55	0.969	15.94
		SPI	0.963	15.24	0.956	15.10	0.947	19.25	0.936	18.38	0.962	15.76
			QRF		CI-jack		CI-split		IF-jack			
			0.967	22.41	0.951	15.44	0.967	21.64	0.954	15.95		
Dataset :	Concrete Compression		HDR		CHDR		Quant		SPI		LM	
n :	1030	LS	0.944	20.20	0.945	19.34	0.945	20.92	0.949	20.23	0.951	18.87
p :	8	L1	0.947	21.03	0.952	19.95	0.946	22.65	0.947	20.85	0.949	19.48
		SPI	0.943	21.01	0.949	19.89	0.947	22.50	0.943	20.68	0.949	19.43
			QRF		CI-jack		CI-split		IF-jack			
			0.979	34.19	0.953	19.70	0.952	25.98	0.959	20.32		
Dataset :	Airfoil Self Noise		HDR		CHDR		Quant		SPI		LM	
n :	1503	LS	0.947	11.29	0.948	9.19	0.943	10.70	0.942	9.935	0.949	9.134
p :	5	L1	0.949	11.99	0.949	9.67	0.946	11.45	0.943	10.431	0.948	9.857
		SPI	0.946	11.52	0.950	9.26	0.945	10.99	0.943	9.974	0.948	9.271
			QRF		CI-jack		CI-split		IF-jack			
			0.988	20.25	0.949	13.35	0.949	14.55	0.946	13.049		
Dataset :	Music Origin (latitude)		HDR		CHDR		Quant		SPI		LM	
n :	1059	LS	0.947	42.74	0.945	47.01	0.945	45.64	0.946	43.73	0.942	55.59
p :	68	L1	0.948	46.07	0.947	49.34	0.944	46.30	0.945	45.60	0.940	57.51
		SPI	0.945	42.08	0.946	47.16	0.942	44.82	0.946	43.69	0.942	57.56
			QRF		CI-jack		CI-split		IF-jack			
			0.989	60.81	0.951	66.57	0.953	68.36	0.942	62.52		
Dataset :	Music Origin (longitude)		HDR		CHDR		Quant		SPI		LM	
n :	1059	LS	0.949	143.5	0.948	144.6	0.947	132.9	0.944	129.3	0.945	150.5
p :	68	L1	0.946	146.1	0.946	146.8	0.948	135.2	0.948	132.9	0.947	154.0
		SPI	0.944	141.5	0.945	142.2	0.943	129.9	0.944	127.1	0.943	152.3
			QRF		CI-jack		CI-split		IF-jack			
			0.988	177.4	0.951	173.3	0.950	177.5	0.940	165.1		

again that the QRF method is conservative while all other methods produce PIs with a coverage closer to .95. Hence, we can compare the methods through their mean lengths. A first general finding clearly apparent in Figures 8, 9, and 10 is that the MLs of the QRF method are too long and not competitive. This is similar to what we found in the simulation study and can be easily explained by the fact that it is too conservative. Likewise, among the two conformal inference methods, CI-jack provides systematically shorter PIs than CI-split.

Figure 1.8 presents the results for the first 3 data sets. For the Boston Housing data set, the proposed 15 method combinations perform fairly well in comparison to the four competitors. Ten out of the 15 perform better than the best competing method, IF-jack. The results with the LS splitting criterion are slightly better than the ones from the two other splitting criteria. With LS, the best PI construction method is SPI, followed by HDR and quantiles, respectively. For the BWC data set, all the proposed methods show a distinctly better performance than the competing methods. For the Concrete Slump data set, 6 of the proposed methods are better than the best competing methods, which are CI-jack and IF-jack. For the proposed methods, CHDR followed by HDR and LM yield noticeably shorter intervals than the QRF and SPI construction methods. In terms of splitting rule, LS has a small advantage over the two others.

Figure 1.9 presents the results for the next 2 data sets. With the Concrete Compression data set, LM is the best PI construction method, especially with the LS splitting criterion, followed by CHDR. CI-jack and IF-jack show a very good performance with this data set, and provide ML almost as short as the ones of the best proposed methods. This is not the case for with the Airfoil data set, where all 15 proposed methods provide shorter PIs than all competing methods. The CHDR and LM PI construction methods are the best performers.

Figure 1.10 presents the results for the 2 response variables, *latitude* and *longitude*, with the Music Origin data set. The 15 proposed methods are distinctively better than the 4 competitors for both responses. However, we see a different behavior from the methods for these 2 responses. With the *latitude* response variable, HDR and SPI are the best performing PI methods. Meanwhile, LM shows a particularly poor performance but that is still better than the competitors. The *longitude* response yields different performance patterns. The SPI and quantiles construction methods show a very good performance.

We can conclude that several or all the proposed methods showed better performance than

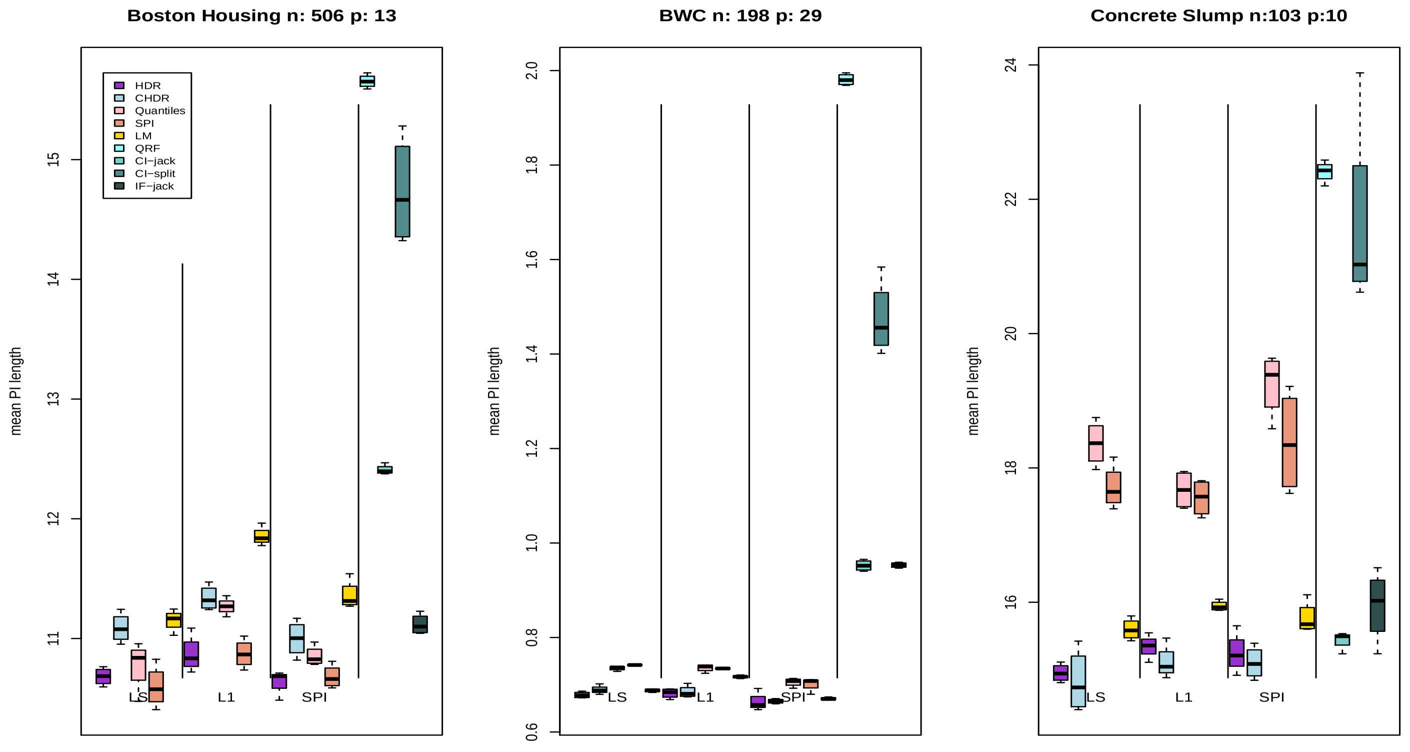
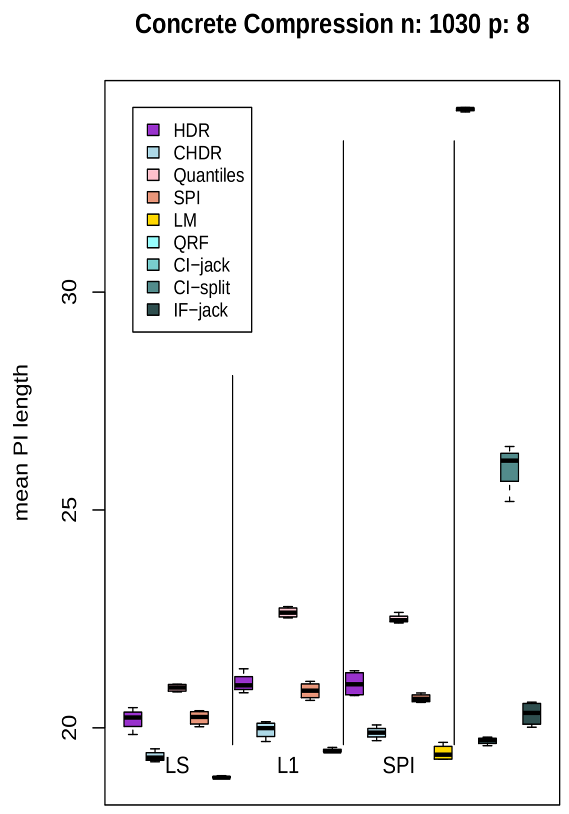


Figure 1.8: Distributions of the mean PI length for datasets Boston Housing, BWC and Concrete Slump.



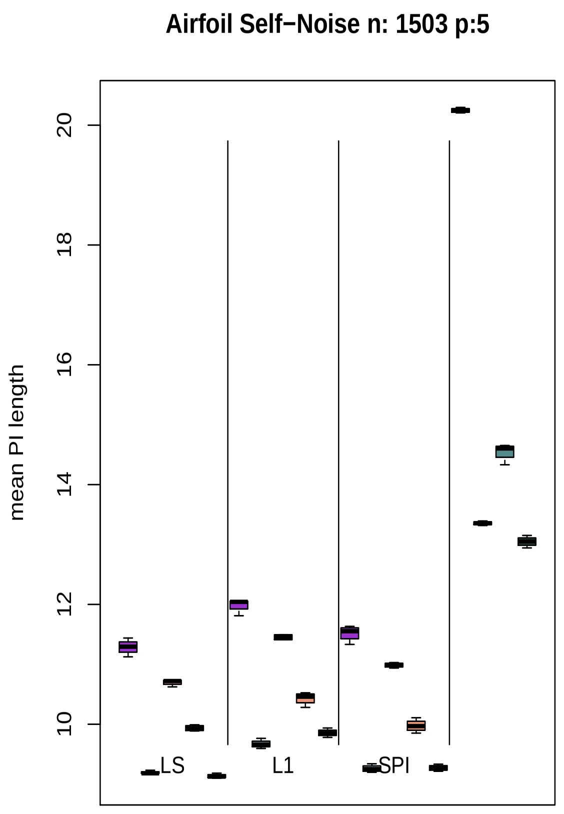
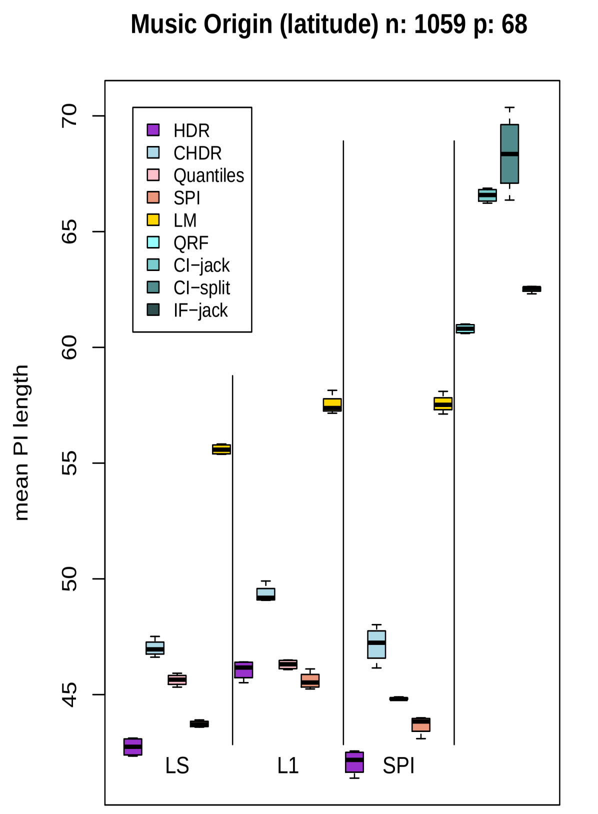


Figure 1.9: Distributions of the mean PI length for datasets Concrete Compression and Airfoil Self Noise.



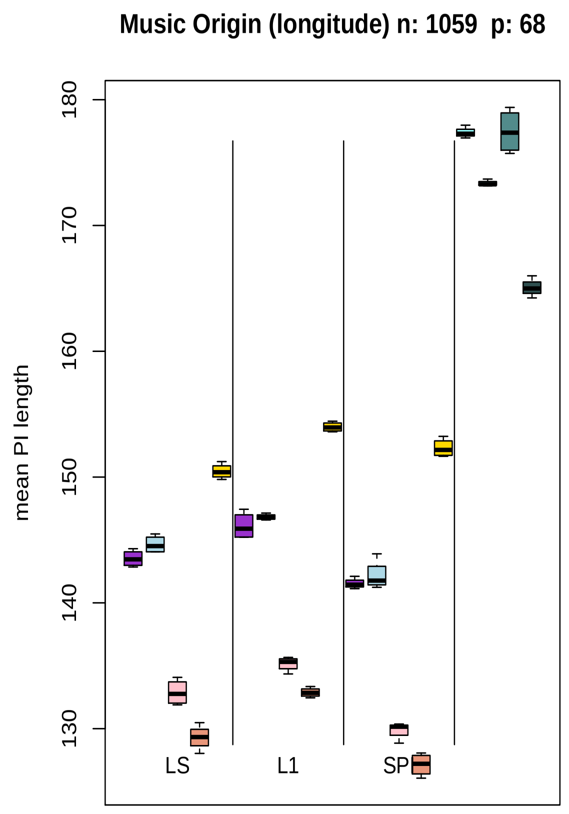


Figure 1.10: Distributions of the mean PI length for datasets Music Origin Latitude and Music Origin Longitude.

the competitors. Among the proposed methods, there is no clear winner, or best performing PI construction method or method combination

1.6 Conclusion

The goal of this research was to investigate new avenues to improve the performance of prediction intervals with random forests. The basic idea is to use the forest to extract local information at a given value of the covariates $\boldsymbol{x}_{new}$ through the Bag of Observations for Prediction (BOP). The BOP is simply the pooled set of observations that are in the same terminal nodes as $\boldsymbol{x}_{new}$ in the forest, a concept very similar to the nearest neighbor forest weights [7, 14]. The general method can be summarized is three steps: 1) Build a forest and get the BOPs for the new observations; 2) Compute the PIs using these BOPs; 3) Calibrate the PIs using the Out-Of-Bag (OOB) information. First, we looked at two alternatives to the default least squares (LS) splitting rule in the tree construction itself (step 1). Second, we presented five methods for constructing prediction intervals once a forest is built (step 2). The resulting 15 method combinations were evaluated in a simulation study and with real data sets.

The proposed methods performed very well. First of all, they were able to maintain the desired coverage level in the simulation study and with the real data sets, showing that the proposed calibration method is reliable. In terms of length, some, or in some cases all, of the proposed methods provided shorter PIs compared to the ones of the competing methods in the simulation study and with real data sets.

In the light of the simulation study and real data sets evaluation, we can conclude that, considering the three splitting criteria investigated, the impact of the choice of this criterion on the performance of the PIs is minimal. The method used for constructing the intervals shows to have a significantly greater impact on the performance. A similar conclusion was obtained in Moradian et al. [19] in the context of estimating the survival function with a survival forest when dependent censoring is present. In that paper, it was shown that correcting for dependent censoring when computing the survival function with the BOP had more positive impact than using a splitting rule that accounts for dependent censoring.

However, there is no clear best performing approach among the five PI methods investigated.

They all provide a solid performance in at least one simulation scenario or real data context. However, the quantiles method was globally the weakest method and we can not recommend it at this time. The simplest method, using the default least-squares criterion to build the forest and then constructing the prediction intervals with the classical linear model method performed very well all around.

With respect to the evaluated competitors, after evaluation in the simulation study and with real data sets, we conclude that the two best performers are the jackknife version of the conformal inference method (CI-jack) by Lei et al. [12] and an adaptation of the infinitesimal jackknife (IF-jack) by Wager et al. [24], that we proposed here. The proposed adaptation was required because the Wager et al. [24] method provides the quantities to build confidence intervals and not prediction intervals. However, the IF-jack method is the most computationally intensive method among those studied here. The present study was limited to random forests. One of the main appeal of the conformal inference approach is that it can be used with any prediction models, and not only with forests.

The QRF method provided longer intervals than all other methods but it was due to the fact that they were conservative, with a coverage higher than required. We used this method off-the-shelf with no additional calibration procedure. Calibrating it would certainly improve its performance. However, the QRF method used a forest built with the least-squares criterion, then estimates the cdf using the nearest neighbor forest weights, and finally computes the PI using the estimated quantiles. In essence, the QRF method is very close to the combination LS-quantiles of the proposed methods, that is, build the forest with the least-squares criterion, and use the quantiles of the BOP to build the PI. This is why we expect that the QRF method, once calibrated, would provide a similar performance to the LS-quantiles combination. But we saw that the quantiles construction method was the weakest method among the five proposed and studied here. Intuitively, we would expect some benefits from using the quantiles when the error distribution is asymmetric. DGP2 in the simulation study was such a case with an exponential error. But even then, the quantiles construction method did not perform well.

Interesting avenues for future research that are currently under investigation include the extension of the proposed methods to more complex settings, including survival data and mixture models.

References

[1] Athey, S., Tibshirani, J., and Wager, S. (2017). Generalized random forests. arxiv preprint. *arXiv preprint arXiv:1610.01271*.

[2] Bache, K. and Lichman, M. (2013). Uci machine learning repository.

[3] Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

[4] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[5] Friedman, J. H. (1991). Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67.

[6] G'Sell M., Lei J., R. A. T. R. and L., W. (2017). *e Tools for conformal inference in regression*. R package conformalInference version 1.0.0.

[7] Hothorn, T., Lausen, B., Benner, A., and Radespiel-Tröger, M. (2004). Bagging survival trees. *Statistics in medicine*, 23(1):77–91.

[8] Hothorn, T. and Zeileis, A. (2017). Transformation forests. *arXiv preprint arXiv:1701.02110*.

[9] Hyndman, R. J. (1996). Computing and graphing highest density regions. *The American Statistician*, 50(2):120–126.

[10] Hyndman, R. J. (2015). *Highest density regions and conditional density estimation*. R package version 3.1.

[11] Ishwaran, H. and Kogalur, U. B. (2017). *Random Forests for Survival, Regression, and Classification (RF-SRC)*. R package version 2.5.1.

[12] Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J., and Wasserman, L. (2017). Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, (just-accepted).

[13] Leisch, F. and Dimitriadou, E. (2010). *mlbench: Machine Learning Benchmark Problems*. R package version 2.1-1.

[14] Lin, Y. and Jeon, Y. (2006). Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590.

[15] Meinshausen, N. (2006). Quantile regression forests. *Journal of Machine Learning Research*, 7(Jun):983–999.

[16] Meinshausen, N. (2016). Quantregforest: quantile regression forests. *R package version 1.3-5*.

[17] Mentch, L. and Hooker, G. (2016). Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *The Journal of Machine Learning Research*, 17(1):841–881.

[18] Moradian, H., Larocque, D., and Bellavance, F. (2017a). L_1 splitting rules in survival forests. *Lifetime data analysis*, 23(4):671–691.

[19] Moradian, H., Larocque, D., and Bellavance, F. (2017b). Survival forests for data with dependent censoring. *Statistical Methods in Medical Research*, page 0962280217727314.

[20] R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[21] Samworth, R. J. and Wand, M. P. (2010). Asymptotics and optimal bandwidth selection for highest density region estimation. *The Annals of Statistics*, 38(3):1767–1792.

[22] Sexton, J. and Laake, P. (2009). Standard errors for bagged and random forest estimators. *Computational Statistics & Data Analysis*, 53(3):801–811.

[23] Vovk, V., Gammerman, A., and Shafer, G. (2005). *Algorithmic learning in a random world*. Springer Science & Business Media.

[24] Wager, S., Hastie, T., and Efron, B. (2014). Confidence intervals for random forests: the jackknife and the infinitesimal jackknife. *Journal of Machine Learning Research*, 15(1):1625–1651.

Chapter 2

Prediction Intervals for Finite Mixture of Regressions Based on Random Forests

Marie-Hélène Roy, Denis Larocque, Ilmari Ahonen, & Jaakko Nevalainen

Abstract

Finite mixture regression models (FMR) are a flexible tool for statistical modeling of data that originate from heterogeneous populations. The classical FMR methods rely on mixtures of linear models. We propose a nonparametric regression method for finite mixture models that captures nonlinear dependencies links between the response and the covariates. To achieve this, we use the structure of the classical EM algorithm but substitute the traditionally used linear models with random forests. We propose five variations to build prediction intervals and a calibration method that ensures the intervals reach the desired coverage level. The performance of the methods are assessed in an extensive simulation study. The results show that the calibration method is very reliable in attaining the prescribed coverage, even with small samples. The results also show that the new methods produce short prediction intervals compared to existing methods in a variety of situations involving linear and nonlinear dependencies, small and large sample sizes, and even with heteroscedastic data. Moreover, four out of the five methods also show very good computational efficiency.

2.1 Introduction

Finite mixture regression models (FMR) are a flexible tool for statistical modeling when there is heterogeneity in the population or group-structure in the data. Consider a continuous response variable y and a p-dimensional vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$ of covariates. When the population is heterogenous, the relationship between y and $\boldsymbol{x}$ may differ across different groups of observations. FMR models offer a convenient way of modeling such heterogenous relationships. In the classic FMR model, for a population that includes K components (subpopulations or groups), the conditional density function of y given $\boldsymbol{x}$ is

$$f(y|\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \phi(y; \beta_{0k} + \boldsymbol{x}^T \beta_k, \sigma_k^2), \tag{2.1}$$

where $\phi(y;\mu,\sigma^2)$ is the density of a normal distribution with mean μ and variance σ^2. For each $k = 1, \ldots, K$, (β_{0k}, β_k^T) are the regression parameters associated with the k-th subpopulation, and σ_k^2 is the variance of the distribution of this subpopulation. The mixing probabilities π_k, such that $\pi_k > 0$, $\sum_{k=1}^{K} \pi_k = 1$, assign a proportion or contribution to the k-th subpopulation. In most methods, the number of components K, also called the order of the model, is assumed known *a priori*. In this case, the set of all parameters is given by $(\pi_k, \beta_{0k}, \beta_k, \sigma_k), k = 1, \ldots, K$. The expectation-maximization algorithm (EM) [4] is an iterative technique that is traditionally used to find the maximum likelihood estimate (MLE). It is by far the most commonly used method to estimate the parameters of a FMR model. McLachlan and Peel [17] provides a comprehensive treatment of FMR models.

Estimating the parameters and clustering the data are common goals with FMR, but predicting the response for new subjects can also be of interest. However, using a single point prediction in the context of a FMR model is not always useful nor meaningful. Indeed, a single point prediction could be a weighted average of the estimated subpopulation means, but this prediction may end up very far away from any individual subpopulation, for instance in the case of a well separated bimodal distribution. Alternatively, assigning the subject to a given subpopulation and using its estimated mean as the point prediction does not make sense either.

This is why using a prediction interval (PI) is a meaningful way of providing information about the likely location of the response for a new subject. To be clear, we will use the term prediction interval in this paper, even though the reported set of likely values may be formed by more than one subinterval, which is expected in the case of a well separated multi-modal distribution.

Once the FMR model (2.1) is fit, building a PI for a new observation can easily be done, for instance using the highest density region method [11]. However, the performance, that is the coverage probability and length, of the PIs built with the basic FMR model relies on how well it fits the data.

One way to improve the basic FMR model, especially when many covariates are present, is to use shrinkage methods. For instance, Khalili and Chen [12], Galimberti et al. [5], Khalili et al. [13], Städler et al. [23] propose lasso-type approaches for FMR models. Another way to improve basic FMR model is to relax the rigid parametric assumptions and use nonparametric or semiparametric methods. In recent years, several authors have suggested such approaches for FMR models. Huang and Yao [10] propose a semiparametric method where the regression functions are estimated in a linear fashion but the mixing proportions are nonparametric, using smoothing functions of a covariate. Huang et al. [9] develop a nonparametric approach where the estimation procedure is done using kernel regression. They also propose a modified EM algorithm. Xiang [24] also introduces a semiparametric approach similar to Huang et al. [9] but where the mixing proportions and variances are constants, while the component regression functions are also smooth functions of a covariate. The author also presents two related modified EM algorithms. This author as well as Huang et al. [9] present a methodology and results solely in the univariate context. Both publications state that the methodology can be extended to a multivariate context, but that the curse of dimensionality [6] limits its usefulness. Huang et al. [8] propose another semiparametic FMR estimation procedure, this time in the context of functional data. It uses in combination, kernel regression, functional principal component analysis and the EM algorithm.

Ahonen et al. [1] is the first work to propose and investigate a flexible way to build PIs with FMR models. They introduce the flexible finite mixture regression (FMRFLEX), a semiparametric method that combines random forests [3] and FMR variable selection techniques with the classical FMR model. Basically, this method expands the set of covariates by adding dummy variables extracted from the terminal nodes of trees in random forests fit using the original covariates $\boldsymbol{x}$ and the outcome y. These additional variables are used to model deviations from linearity. Since this adds many new variables, a penalized FMR regression is then fit on the combination of the original covariates and the additional dummy variables to select the most important ones and perform shrinkage. Once the model is fit, PIs can be build with the highest density region method [11]. They compare their FMRFLEX method with one-class models, the classical finite mixture model and the fmrlasso of Städler et al. [23] in extensive simulation studies. They show that their proposed method achieves equal performance with the other FMR methods when the true model is linear and superior performance in nonlinear cases. They also obtain a superior performance with a real data set where important relationships and/or interactions between variables were likely nonlinear.

Although FMRFLEX performs well in the settings considered in Ahonen et al. [1], the computing time can become an issue in some circumstances. Moreover, other promising ways to build PIs with a random forest (RF) were recently proposed in Roy and Larocque [20] in the general supervised learning setting. Inspired by both papers, the goal of this paper is to propose and investigate other ways to build PIs with RF for FMR. We propose a way to build a mixture of RFs that can model nonlinear multivariate relationships, and five ways to build PIs with it. We then study these methods with a simulation study that includes challenging settings with small samples and heteroscedastic data.

In section 2, we present the methods. The results from the simulation study are presented in section 3. Section 4 concludes and provide avenues for further research.

2.2 Methodology

In this section, we provide a detailed description of the proposed methods. Their goal is to provide a PI for the value of y for a new observation $\boldsymbol{x}_{new}$.

Random Forests

Random forests, that are now part of the essential toolbox of any data analyst, is a very powerful prediction method that can be used for both classification, regression and many other problems including survival data. One of its main appeal is its ability to effectively capture nonlinear dependencies and interactions between covariates. We will focus on the regression context. Random forests are formed by an ensemble of B regression trees. Each tree is grown with a bootstrap sample of the observations and with randomly selected covariates. More precisely, at each node of a tree, rather than choosing the best split among all covariates, it chooses among a randomly selected subset of covariates. The basic RF method takes the average predictions from the terminal nodes of the B trees to predict new data.

Finite Mixture Models using Random Forests

To model nonlinear dependencies between the response and the covariates, we relax the linear assumption in (2.1) and consider the model

$$f(y|\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \phi(y; h_k(\boldsymbol{x}), \sigma_k^2), \tag{2.2}$$

where $h_k(\cdot) : \mathscr{R}^p \to \mathscr{R}$ is an arbitrary unknown function of the covariates, and where the other quantities are defined as in (2.1). We use random forests to approximate the functions $h_k(\cdot)$, for all $k = 1, \dots, K$. To achieve this, we modify the traditional EM algorithm used to fit a FMR model by replacing linear models in the M step by K random forests. This leads to a nonparametric estimation of the FMR. Here is a description of the modified EM algorithm, followed by more details about some key steps.

2.2.1 The modified EM algorithm

Initialization

In what follows, i is the index of the observation, k of the component (class) in the mixture, and j is the index of the iteration number. We denote by p_{ik} the probability that observation i belongs to class k.

Find initial classification probabilities $p_{ik}^{(0)}$ using a linear FMR model. We use the package *flexmix* [14] for this purpose. Find initial mixture proportions $\pi_k^{(0)} = \frac{1}{n} \sum_{i=1}^{n} p_{ik}^{(0)}$. Set the number of trees in a forest to B. Set iteration index j to 0.

The random forest EM algorithm for a mixture iterates between the following E and M steps, starting with the M-step.

E-step

Increment j by 1. Update the classification probabilities p_{ik}.

$$p_{ik}^{(j)} = \frac{\pi_k^{(j)}\phi(y_i;\hat{h}_k(\boldsymbol{x_i}),\sigma_k^{2(j)})}{\sum_{k=1}^{K}\pi_k^{(j)}\phi(y_i;\hat{h}_k(\boldsymbol{x_i}),\sigma_k^{2(j)})}, i=1,\dots,n; k=1,\dots,K.$$

M-step

for each $k = 1,\dots,K$ {

1. Perform a weighted least squares linear regression on response vector y using covariates $\boldsymbol{x}$ and weights set equal to the classification probabilities $p_{ik}^{(j)}$, to obtain an estimated linear model $\hat{l}_k(\boldsymbol{x})$. Obtain the linear predictions $\hat{y}_{ik} = \hat{l}_k(\boldsymbol{x_i})$ and residuals $r_{ik} = y_{ik} - \hat{y}_{ik}$. This is done to remove the linear part. The residuals will be used to build the RF.

2. Train a random forest using the residuals r_{ij} as the response and the covariates $\boldsymbol{x}$.

2.1 a) Set sampling probabilities to $p_{ik}^{(j)}$. Generate B weighted bootstrap samples from the data. For each sample draw, with replacement, $\left\lceil n\pi_k^{(j)} \right\rceil$ observations out of n.

b) As in the classic RF algorithm a tree is build for each of these samples, building a complete random forest.

c) The RF predictions are obtained from averaging predictions from all B trees. Let $\hat{g}_k(\boldsymbol{x})$ denote this forest.

d) For any $\boldsymbol{x}$, the final prediction is $\hat{h}_k(\boldsymbol{x}) = \hat{g}_k(\boldsymbol{x}) + \hat{l}_k(\boldsymbol{x})$.

2.2 a) Obtain predictions for the out-of-bag observations for each tree.

b) Gather out-of-bag predictions from the B out-of-bag samples. Compute the average for each observation that belongs to O_k, the ensemble of the observations that were present in a out-of-bag sample at least once. There are n'_k averaged out-of-bag predictions $\hat{g}_{ik}^{oob}$, for $i \in O_k$.

c) From the n'_k out-of-bag predictions, update σ_k^2.

$$\sigma_k^{2(j+1)} = \frac{1}{n'_k}\sum_{i\in O_k} p_{ik}^{(j)}(y_i - (\hat{y}_{ik} + \hat{g}_{ik}^{oob}))^2.$$

}

3. Update π_k,

$$\pi_k^{(j+1)} = \frac{1}{n}\sum_{i=1}^{n} p_{ik}^{(j)}.$$

The algorithm stops when the observed log-likelihood $\sum_{i=1}^{n} \log(\hat{f}(y_i|\boldsymbol{x}_i))$ stabilizes, where $\hat{f}(y|\boldsymbol{x})$ is the density (2.2) evaluated with the current values of π_k, σ_k^2, and $\hat{h}_k(\boldsymbol{x})$.

It is well-known that a RF will not capture a linear link as fast as a linear model. This is why, in step 1 of the M-step, a weighted linear model is fit for each class, with observations weights equal to the classification probabilities, in order to detect such a link, if present. The residuals from these linear fits are used as responses to build the RFs, one per class, in step 2.1. The idea is to let the RF detect the remaining signal. But we cannot use the classical RF algorithm and simply take ordinary bootstrap samples because we have to account for the classification probabilities of the observations. This is why, for each class, a weighted bootstrap sample with $\left\lceil n\pi_k^{(j)} \right\rceil$ observations is generated with the sampling probabilities set to $p_{ik}^{(j)}$. Then, classical RFs are built with these samples, allowing to extract the out-of-bag samples, which will be used to estimate the variance parameters σ_k^2 in step 2.2.

Note that the prior removing of the linear effect in step 1 of the M-step is optional. In fact, it is only performed for the first PI building method presented below because preliminary results showed that it improved the results. Because of the way they are designed, it should not be used for the other four PI building methods. If the prior removing of the linear effect is not required, we can just set $\hat{l}_k(\boldsymbol{x}) = 0$, then $\hat{y}_{ik} = 0$ and the residuals are the original observations $r_{ik} = y_{ik}$. This way forests are built with the original observations and the algorithm remains the same.

Once the finite mixture model using RF is fit, we can use it to compute PIs. Five different approaches will be presented. The first one is a fully parametric approach, adapted from Ahonen et al. [1]. The four other approaches are inspired by Roy and Larocque [20]. Two of them are nonparametric and two semiparametric. The next subsection describes these methods.

2.2.2 Prediction Intervals

Some of the methods use the concept of highest density region (HDR). For completeness, the definition of a HDR from [11] is reproduced here.

Definition. Let $f(x)$ be the density function of a random variable X. Then the $(1-\alpha)100\%$ HDR is the subset $R(f_\alpha)$ of the sample space of X such that $R(f_\alpha) = \{x : f(x) \geq f_\alpha\}$ where f_α is the largest constant such that $Pr(X \in R(f_\alpha)) \geq 1-\alpha$.

The HDR is the smallest region, with the desired coverage $(1-\alpha)100\%$, such that the density of every point inside it is at least as large as the density of every point outside it. Thus the region $R(f_\alpha)$ forms a PI with confidence level $1-\alpha$. In practice, the density $f(x)$ is unknown. If a parametric model is assumed, then the parameters can be estimated and the HDR can be obtained from the estimated density either exactly or by simulation. Alternatively, a nonparametric approach can be used with a density estimator like the kernel [11, 21]. The HDR can be obtained also by simulation from the estimated density.

PI method 1: HDR1 (Adaptation of the Ahonen et al. [1] method)

This method is the one that is the most closely related to the Ahonen et al. [1] approach. Once the model is fit with the random forest EM algorithm for a mixture, we have estimated values $\hat{\pi}_k$ and $\hat{\sigma}_k^2$, for $k = 1,\ldots,K$. To build a PI for a new observation $\boldsymbol{x}_{new}$, we also have the predicted values for each class, $\hat{h}_k(\boldsymbol{x}_{new})$. Hence, we could simply plug-in these values in (2.2) and compute the PI with the HDR method. However, it was found in Ahonen et al. [1] that this provides PIs that tend to be liberal because they are not taking into account the uncertainty related to the estimation of the parameters. This is why Ahonen et al. [1] proposed a bootstrap adjustment to estimate this variability, and thus to calibrate the PIs. This bootstrap adjustment is explained in section 3.2 of Ahonen et al. [1]. We are using the same approach, only the way we get the estimated parameters of the mixture distributions is different. We identify this PI method as HDR1.

2.2.3 Four PI methods based on the Bag of Observations for Prediction (BOP)

These four PI methods also use the random forest EM algorithm for a mixture to build the model. However, as mentioned previously, the prior removing of the linear part in step 1 of the M-step is not performed. This amounts to setting $\hat{l}_k(\boldsymbol{x}) = 0$. These PI use the concept of “Bag of Observations for Prediction" (BOP), introduced under this name in [20] but already used in Moradian et al.

[18] and Moradian et al. [19]. This concept is very similar to the nearest neighbor forest weights of Hothorn et al. [7] and Lin and Jeon [16]. The main idea is to see a RF as a way to generate a set of local observations that will be used to compute the prediction. In general, assume we have a RF and we want to get a prediction (of any kind) for a new observation $\boldsymbol{x}_{new}$. The Bag of Observations for Prediction for this new observation, $BOP(\boldsymbol{x}_{new})$, is the combined set of training observations that fall in the same terminal nodes as $\boldsymbol{x}_{new}$ in the trees of the forest. Note that the same observation can appear multiple times in the BOP, as it can be in the same terminal node as $\boldsymbol{x}_{new}$ for many trees and also since it can appear more than once even in a single tree if it was resampled more than once by the bootstrap. Once we have $BOP(\boldsymbol{x}_{new})$, we can compute any desired summary using this set of observations.

In the current setting, we have K forests, one per class. Hence, we have K BOP for each observation. Denote by $BOP_k(\boldsymbol{x}_{new})$ the BOP for class k, $k = 1, \ldots, K$. For three of the next four PI methods, we need also to combine these BOPs to get a single BOP per observation. We can not simply pool $BOP_1(\boldsymbol{x}_{new}), \ldots, BOP_K(\boldsymbol{x}_{new})$ together because it would not necessarily reflect the right mixture proportions. Instead, we take subsamples of the $BOP_k(\boldsymbol{x}_{new})$ in such a way that, when pooled, the resulting BOP will have the largest amount of observations while having the right proportion of observations from each forest. We call this BOP, the adjusted BOP and denote it by $BOP_a(\boldsymbol{x}_{new})$. Hence, $BOP_a(\boldsymbol{x}_{new})$ contains a proportion of $\hat{\pi}_k$ observations from $BOP_k(\boldsymbol{x}_{new})$, for $k = 1, \ldots, K$.

One key feature of this general approach is that it lends itself to an automatic calibration method for the PI, based on the OOB information. This will be explained later. Moreover, an important characteristic of the next four methods is that they rely on local estimations of the variability. Hence, they can handle heteroscedasticity, that is, error variances that can be a function of the covariates. This aspect will be investigated in the simulation study.

PI method 2: NP1 (Nonparametric - BOP with Unrestricted Number of Intervals)

This method was introduced in [20]. Let $\widehat{\text{HDR}}_{\alpha}(D)$ be an estimation of the HDR, for a given α, for a data set D. With this method, the PI is $\widehat{\text{HDR}}_{\alpha}(BOP_a(\boldsymbol{x}_{new}))$. Hence it is the HDR computed with the adjusted BOP.

The estimated HDR can be formed by multiple intervals, and we expect it to be the case in the context of mixture models. If we assume that there are K components, then the number of intervals should be at most K in theory. However, in this version no restriction on the number of intervals is imposed. If a too small bandwidth is used with a kernel density estimate, then it is possible that the estimated HDR contains more than K intervals. This may or may not be important and depends on the application. The next version is a slight modification of this one, allowing restrictions on the number of intervals.

PI method 3: NPK (Nonparametric - BOP with at most K Segments)

This approach is the same as the previous one but adds an interval combination step at the end to obtain at most K intervals. Compute $\widehat{\text{HDR}}_\alpha(BOP_a(\boldsymbol{x}_{new}))$, the PI from the previous (NP1) method. If the number of intervals is less or equal than K, then it is the final PI. If not, try every possible (there is a finite number of possibilities) way to fill the gaps between the intervals to get K intervals, and select the PI with the shortest total length. This in a simple extension of the method used in [20]. In that paper, it was required to bring down the number of intervals to one, and the only way to do it was to fill the gaps between all intervals. To fix ideas, assume $\widehat{\text{HDR}}_\alpha(BOP_a(\boldsymbol{x}_{new}))$ is formed by three intervals $[5,10]$, $[12,13]$, $[15,22]$, and that we want two intervals. There are two ways to fill the gaps to get two intervals. The first one is to consider the two intervals $[5,13]$, $[15,22]$, and the second one is $[5,10]$, $[12,22]$. Since the total length of the PI $[5,13]$, $[15,22]$ is shorter, it would be the final PI.

PI method 4: SP1 (Semiparametric - BOP parameters from each class)

For this semiparametric method (SP1), we use the individual class BOPs, $BOP_1(\boldsymbol{x}_{new}),\ldots,BOP_K(\boldsymbol{x}_{new})$, to estimate the mixture parameters in (2.2). First of all, the estimated mixing proportions $\hat{\pi}_k$ are readily available from the fit model. The mean and variances parameters are obtained directly from the BOPs. For the mean parameters, $\hat{h}_k(\boldsymbol{x}_{new})$ is the average of the observations in $BOP_k(\boldsymbol{x}_{new})$. For the variance parameters, $\hat{\sigma}_k^2$ is the variance of the observations in $BOP_k(\boldsymbol{x}_{new})$. With these estimated parameters, the conditional mixture distribution (2.2) for $\boldsymbol{x}_{new}$ is fully known. The PI is the HDR of this distribution. It is important to note that we are using local estimations of the variance and not the global ones coming from the fit model. As mentioned earlier, this allows the method to handle heteroscedasticity. This is in contrast with PI method 1 (HDR1), which uses the global estimated variances, and only the $\hat{h}_k(\boldsymbol{x}_{new})$ vary with $\boldsymbol{x}_{new}$.

PI method 5: SP2 (Semiparametric - BOP with mixture)

The previous method takes the global mixing proportions, and estimates the mean and variances of each component separately using their respective BOP. The SP2 method uses the adjusted BOP, $BOP_a(\boldsymbol{x}_{new})$, and estimates the whole mixture distribution from it. In this study, we use the *mixtools* package [2] to achieve this. The initial parameters given to the EM algorithm in *mixtools* are the ones from SP1. The PI is the HDR of this estimated distribution. Hence, not only the variance parameters can vary with $\boldsymbol{x}_{new}$, but the mixing proportions too since they are re-estimated as well.

2.2.4 Calibration for the four BOP methods

If we apply the PI building methods based on BOPs described above using the desired coverage level, then the resulting PIs will sometimes be too conservative and sometimes too liberal. In order to reliably attain the desired $1-\alpha$ level, we must use a calibration procedure. The one we use in this paper is essentially the one proposed in Roy and Larocque [20]. A shorter description is given here and we point out a modification we made to it. This calibration uses the BOPs that are collected from the predictions of the OOB observations for all trees. Basically, for a given training sample observation, we can obtain the individual class BOPs and adjusted BOP in the same way as for a new observation, except that we only use the trees where this observation is OOB. Once we have these BOPs, we can construct the PIs for the training data in the same way as for the new observations. These OOB PIs are then used for the calibration. The calibration is done through the parameter $1-\alpha$. Let $1-\alpha$ be the target coverage level and let α_w be the “working" value of α. The idea is to find the value of α_w such that the coverage level computed with the training observations is close enough to the target level. Once found, $1-\alpha_w$ becomes the level used to build the PI for the new observations. In the simulation study, the target $1-\alpha$ is .95 and α_w is selected such that the OOB coverage level lies in the interval $[.94, .95]$. This calibration method always worked in Roy and Larocque [20]. But in the simulation study of this paper, in some cases, the target coverage level was not reached even when a $1-\alpha_w$ was close to 1 for the NP1 and NPK methods. In those cases, we calibrated the PIs by varying the bandwidth used in the kernel density estimate.

2.3 Simulation Study

2.3.1 Simulation Design

To assess the performance of the proposed methods, we conduct a simulation study. The study is composed of 48 scenarios which are variations of the four Data Generating Processes (DGPs) presented in Ahonen et al. [1]. These four DGPs have 2 or 3 components and are described next, but see Ahonen et al. [1] for a graphical representation of them.

DGP A

The first DGP is composed of two components. In both, the outcome is linearly dependent on the covariates. The setting serves as a benchmark to compare the predictive performance of the proposed methods in a simple linear setting. We have a p-dimensional covariate vector $\boldsymbol{x} = (x_1, \ldots, x_p)'$ where x_i, for $i = 1, \ldots, p$, are sampled from the uniform distribution between 0 and 10 with pairwise expected Pearson correlations set to 0.2. The mixture distribution of the outcome is:

$$f(y|\boldsymbol{x}) = \pi_1 \phi(y; \beta_{01} + \boldsymbol{\beta}_1' \boldsymbol{x}, \sigma^2) + \pi_2 \phi(y; \beta_{02} + \boldsymbol{\beta}_2' \boldsymbol{x}, \sigma^2)$$

where $\pi_1 = \pi_2 = \frac{1}{2}$, $\beta_{01} = 0$, $\boldsymbol{\beta}_1' = 1/p(2, \ldots, 2)$, $\beta_{02} = 5$, and $\boldsymbol{\beta}_2' = 1/p(-1, \ldots, -1)$.

DGP B

DGP B is made of one component with a quadratic relationship to the outcome and a second linear component. The settings are the same as DGP A, with one exception. To create a quadratic relationship in the second component, the following transformation is applied to the covariate vector to yield $\boldsymbol{x}^B = (x_1^2 - 10x_1, \ldots, x_p^2 - 10x_p)'$.

DGP C

The third DGP has three components. The first two are the two components of DGP B, where we have a linear and a nonlinear component. The third component is created with the transformed covariate $\boldsymbol{x}^C = (\sin(1/2x_1), \ldots, \sin(1/2x_p))'$. The outcome has a mixture distribution with three densities:

$$f(y|\boldsymbol{x}) = \pi_1 \phi(y; \beta_{01} + \boldsymbol{\beta}_1' \boldsymbol{x}, \sigma^2) + \pi_2 \phi(y; \beta_{02} + \boldsymbol{\beta}_2' \boldsymbol{x}^B, \sigma^2) + \pi_3 \phi(y; \beta_{03} + \boldsymbol{\beta}_3' \boldsymbol{x}^C, \sigma^2),$$

where $\pi_1 = \pi_2 = \pi_3 = \frac{1}{3}$, $\beta_{03} = 60$, and $\boldsymbol{\beta}_3' = (-10, 0, \ldots, 0)$.

DGP D

In the fourth DGP, we introduce an outcome that is a function of interaction terms in the two components. These interaction terms are presented as the new covariate $\boldsymbol{x}^D = (x_1x_2, x_2x_3, \ldots, x_{p-1}x_p)'$. The mixture distribution of the outcome is :

$$f(y|\boldsymbol{x}) = \pi_1 \phi(y; \beta_{03} + \boldsymbol{\beta}_3' \boldsymbol{x}^D, \sigma^2) + \pi_2 \phi(y; \beta_{04} + \boldsymbol{\beta}_4' \boldsymbol{x}^D, \sigma^2),$$

where $\pi_1 = \pi_2 = \frac{1}{2}$, $\beta_{03} = -20$, $\boldsymbol{\beta}_3' = (1/2, \ldots, 1/2)$, $\beta_{04} = 0$, and $\boldsymbol{\beta}_4' = (-1/2, \ldots, -1/2)$.

Error Variance, Sample Size, and Dimension

For each DGP, we consider three training sample sizes, $n = 100$, 250, and 1000. We also consider the two cases $p = 2$ and $p = 5$. Finally, we consider the constant variance case and a heteroscedastic pattern. In the basic scenarios with a constant error variance pattern, we set $\sigma^2 = 1$. In the heteroscedastic scenarios, we set $\sigma^2 = 1$ if $\boldsymbol{\beta}_1'\boldsymbol{x} \leq 5$ and $\sigma^2 = 9$ if $\boldsymbol{\beta}_1'\boldsymbol{x} > 5$. The combination of all these variations produces 48 scenarios (4 DGPs $\times$ 3 sample sizes $\times$ 2 values of p $\times$ 2 variance patterns). Note that only scenarios with $n = 1000$ for DGPs A, B, C, and $n = 1500$ for DGP D, along with constant variance patterns were considered in Ahonen et al. [1]. The reason we include heteroscedastic cases is because four of the five new proposed methods, the ones based on the BOP, can handle heteroscedasticity. This is because each PI is built separately using a BOP without the need of global estimates of the variability. Investigating the impact of heteroscedasticity on the performance is thus worthwhile. Moreover, we include smaller sample sizes to investigate the ability of all methods to maintain the prescribed confidence level under more difficult situations. The OOB calibration method used by the four BOP methods performed well in Roy and Larocque [20], even with small sample sizes, but here we want to see if it remains the case with mixture data.

All forests were built with the randomForest package [15]. For each forest, 500 trees were grown and the minimum number of observations in the terminal nodes was kept to the default number of 5 observations.

In addition to the five proposed methods we include two existing methods in our simulation study. The first is FMRLASSO, the L_1-penalization method for mixture models of Städler et al. [23], that was also included in the simulation in Ahonen et al. [1]. We use the package fmrlasso [22]. The second is FMRFLEX proposed in [1]. The same code as in that paper is used here.

Throughout this study, the prescribed coverage for the PIs was set to .95 for all methods and scenarios. The random forest EM algorithm for a mixture was run for 10 iterations for each data set, since preliminary simulations showed that it tends to stabilize after 3 to 5 iterations. The performance was evaluated with independent test data of size 1000, and the results are based on 100 simulation runs.

2.4 Results of the Simulation Study

The detailed results from the simulation study are presented in Table 2.1. The table provides the mean coverage of the PIs, and the mean length of the PIs on the test set, over the 100 runs, for all 48 scenarios. To be clear, for an individual run, we compute the mean coverage and the mean length with a test set of size 1000. Then we average these values over the 100 runs. The most important property of a PI, whether it is in a mixture model context or not, is its ability to attain and maintain the prescribed coverage. In Table 2.1, whenever the mean coverage value is less than .93, the values of the coverage, mean length and standard deviation of the length for that method/scenario are shown in gray. For the following discussion, in such a case, we say that the method was not able to maintain its prescribed level for that scenario. The first striking finding is that the four BOP based PI methods are able to provide the right coverage. These four methods were able to maintain the prescribed coverage in all the 48 scenarios. Since it was also the case in Roy and Larocque [20], this calibration procedure shows to be very stable and reliable. All other methods, including the other one proposed in this paper, have encountered problems maintaining their level especially with small sample sizes. Since HDR1 and FMRFLEX use the same bootstrap adjustment method, it comes to no surprise that if one of them has difficulty maintaining its level, than the other one will have problems too. When $n = 1000$, FMRFLEX is able to maintain its level in all cases but one. The sample sizes used in the simulation study of Ahonen et al. [1] were equal or greater than 1000, and the method performed well. This is why the bootstrap adjustment method was not investigated further there. But the new results presented here show that further refinements of this method seem to be required for smaller samples. The FMRLASSO method is taken “off-the-shelf". These results show that additional calibration measures are also required for it to achieve more reliability, especially for smaller samples. If we dig a little further on this aspect, we see that the three methods that had problems maintaining their level had even more trouble with the heteroscedastic scenarios. Indeed for FMRLASSO, among the 24 scenarios where it did not maintain the level, 11 of them were constant variance scenarios and 13 were heteroscedastic scenarios. For HDR1, among the 25 scenarios where it did not maintain the level, 8 of them were constant variance scenarios and 17 were heteroscedastic scenarios. For FMRFLEX, among the 14 scenarios where it did not maintain the level, 5 of them were constant variance scenarios and 9 were heteroscedastic scenarios. But without further investigations, we can not conclude whether this is due to the specific fact that we have heteroscedasticity or simply because there is just more noise in the heteroscedastic scenarios. Indeed, in the constant variance scenarios, the error variance is always 1, whereas it is sometimes 1 and sometimes 9 in the heteroscedastic scenarios, hence there is more noise. Interestingly, even though HDR1 and FMRFLEX are using the same bootstrap adjustment method, the latter was better at maintaining its level.

Table 2.1: Complete results of the simulation study aggregated by DGP, sample size and error variance setup. The column 'cov' shows the mean coverage of the PIs, 'ml' their mean length and 'sd', the standard deviation of the PI lengths. For the error variance setup, 'var'=1 is the constant variance case and 'var'=2, the heteroscedastic pattern. Grayed out cells indicate mean coverage value is less than .93.

				FMR - LASSO			HDR1			NP1			NPK			SP1			SP2			FMR-FLEX		
n	DGP	p	var	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
n	DGP	p	var	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
100	A	2	1	0.897	6.816	0.679	0.933	7.676	0.593	0.944	12.359	1.140	0.943	12.018	0.996	0.945	10.834	1.221	0.944	10.830	1.219	0.939	7.577	0.679
		2	2	0.893	14.419	1.832	0.922	15.958	1.364	0.945	19.382	2.147	0.941	18.744	2.207	0.939	17.760	1.692	0.939	17.765	1.777	0.909	15.363	1.815
		5	1	0.852	6.714	0.905	0.925	7.759	0.651	0.945	12.976	0.924	0.948	12.917	0.840	0.948	12.550	0.945	0.948	12.549	0.949	0.718	7.083	2.278
		5	2	0.850	14.331	2.163	0.905	15.456	1.150	0.939	19.555	1.979	0.938	19.278	1.946	0.941	18.707	1.557	0.941	18.722	1.484	0.805	14.127	3.143
	B	2	1	0.908	19.262	2.028	0.933	12.978	0.807	0.940	18.321	1.736	0.940	17.990	1.528	0.945	15.833	1.523	0.944	15.835	1.519	0.940	16.683	2.619
		2	2	0.896	22.492	2.303	0.908	18.785	2.374	0.934	21.763	2.297	0.931	21.467	2.046	0.933	20.059	1.899	0.933	20.078	1.920	0.912	20.418	2.664
		5	1	0.877	14.816	2.207	0.926	13.886	1.818	0.944	18.027	1.171	0.947	18.008	1.140	0.950	17.463	1.279	0.949	17.393	1.319	0.834	14.477	3.303
		5	2	0.877	19.161	2.239	0.898	18.897	2.114	0.939	20.897	1.589	0.937	20.843	1.667	0.940	20.445	1.493	0.940	20.459	1.503	0.817	16.894	3.211
	C	2	1	0.892	55.550	7.253	0.914	50.296	8.572	0.940	46.483	6.341	0.941	46.297	6.250	0.942	49.584	8.437	0.943	49.813	8.506	0.965	53.382	6.513
		2	2	0.890	58.655	6.707	0.911	54.317	6.649	0.939	50.727	6.439	0.938	50.136	5.796	0.940	54.798	7.278	0.941	54.786	6.951	0.959	57.465	5.626
		5	1	0.875	51.024	8.030	0.856	54.084	6.327	0.940	52.368	5.954	0.940	52.242	6.285	0.942	53.456	5.324	0.942	53.415	5.322	0.934	51.038	6.843
		5	2	0.890	54.252	7.681	0.854	55.738	5.050	0.936	54.622	5.518	0.935	54.323	5.247	0.943	56.426	4.897	0.944	56.501	4.813	0.935	53.541	6.471
	D	2	1	0.872	23.068	2.856	0.893	20.494	1.469	0.938	29.759	3.788	0.937	28.865	3.097	0.941	25.741	2.221	0.942	25.789	2.156	0.957	26.903	2.908
		2	2	0.884	24.948	2.841	0.901	22.920	1.579	0.936	30.687	3.801	0.935	29.951	3.154	0.941	27.801	2.289	0.942	27.817	2.287	0.956	27.851	2.428
		5	1	0.838	50.391	5.350	0.905	55.445	4.604	0.948	140.976	11.950	0.950	137.918	12.546	0.955	125.338	10.892	0.955	125.356	10.502	0.817	63.081	17.676
		5	2	0.833	51.683	6.470	0.905	57.706	5.721	0.951	142.623	12.037	0.955	140.253	12.633	0.956	125.074	11.315	0.956	125.273	11.174	0.807	63.233	18.478

Table 2.1 continued from previous page

				FMR - LASSO			HDR1			NP1			NPK			SP1			SP2			FMR- FLEX		
n	DGP	p	var	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
250	A	2	1	0.936	7.251	0.464	0.943	7.757	0.387	0.945	9.911	0.519	0.944	9.841	0.552	0.946	9.401	0.551	0.946	9.410	0.552	0.953	7.697	0.356
		2	2	0.925	15.175	0.956	0.932	16.021	0.839	0.951	18.234	1.700	0.946	17.659	1.616	0.943	16.969	1.074	0.944	17.007	1.077	0.931	15.412	0.895
		5	1	0.916	7.300	0.451	0.944	7.948	0.339	0.948	11.517	0.555	0.950	11.380	0.567	0.949	11.045	0.565	0.949	11.034	0.566	0.860	8.379	1.952
		5	2	0.909	15.241	1.188	0.920	15.604	0.653	0.939	18.250	1.068	0.939	18.118	1.006	0.941	17.717	0.945	0.940	17.693	0.954	0.851	14.134	2.473
	B	2	1	0.937	20.215	1.307	0.941	11.443	2.410	0.945	14.324	1.032	0.945	14.039	1.107	0.944	12.423	1.607	0.944	12.391	1.575	0.955	13.989	1.773
		2	2	0.935	23.581	1.087	0.919	18.313	2.577	0.945	19.400	1.277	0.943	19.307	1.262	0.940	18.147	1.414	0.940	18.147	1.371	0.928	18.766	1.578
		5	1	0.933	15.963	1.333	0.940	12.572	1.400	0.945	15.876	0.742	0.946	15.686	0.714	0.944	14.946	0.807	0.945	14.968	0.834	0.848	10.942	3.071
		5	2	0.924	20.357	1.343	0.923	18.288	1.359	0.943	19.710	1.029	0.943	19.679	1.038	0.942	19.196	1.002	0.942	19.179	1.010	0.874	16.309	2.412
	C	2	1	0.951	60.545	5.801	0.932	42.536	10.537	0.943	34.248	4.001	0.944	34.162	4.439	0.945	40.447	8.917	0.945	40.528	8.866	0.982	48.424	7.980
		2	2	0.947	63.682	4.628	0.934	46.503	7.509	0.946	41.149	4.220	0.945	40.910	4.393	0.942	45.162	7.214	0.942	45.088	7.254	0.974	50.933	6.914
		5	1	0.942	57.198	4.718	0.941	52.951	3.841	0.942	44.019	2.696	0.942	43.505	2.880	0.938	46.294	2.780	0.940	46.421	2.731	0.969	46.982	6.679
		5	2	0.940	59.348	4.471	0.937	55.031	3.865	0.945	48.821	3.278	0.946	48.873	3.157	0.943	50.333	3.122	0.943	50.382	2.948	0.963	49.727	6.315
	D	2	1	0.923	24.951	1.522	0.909	17.532	0.887	0.949	22.048	1.452	0.947	21.345	1.508	0.946	19.128	0.921	0.945	19.076	0.908	0.964	23.244	2.503
		2	2	0.934	26.967	1.569	0.916	21.169	0.991	0.944	23.818	1.413	0.941	23.676	1.401	0.943	22.132	1.147	0.943	22.146	1.133	0.960	25.906	2.332
		5	1	0.907	56.869	4.187	0.925	53.234	2.697	0.950	112.559	7.068	0.950	107.054	6.472	0.949	95.464	4.886	0.949	95.469	5.095	0.935	59.436	11.308
		5	2	0.910	59.251	4.342	0.926	55.298	2.989	0.947	113.117	7.263	0.948	108.658	6.854	0.948	96.458	4.955	0.947	96.291	4.877	0.932	61.905	12.531

Table 2.1 continued from previous page

				FMR - LASSO			HDR1			NP1			NPK			SP1			SP2			FMR-FLEX		
n	DGP	p	var	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
1000	A	2	1	0.948	7.428	0.246	0.947	7.742	0.197	0.948	8.506	0.272	0.948	8.511	0.294	0.946	8.485	0.228	0.946	8.477	0.235	0.957	7.663	0.163
		2	2	0.932	15.400	0.583	0.933	15.928	0.469	0.950	16.917	1.226	0.948	16.702	1.004	0.944	16.289	0.682	0.944	16.319	0.663	0.937	15.667	0.464
		5	1	0.943	7.629	0.234	0.949	7.888	0.187	0.951	9.921	0.270	0.950	9.790	0.280	0.950	9.615	0.247	0.950	9.606	0.252	0.953	8.489	0.555
		5	2	0.928	15.828	0.569	0.926	15.715	0.384	0.945	17.449	0.629	0.945	17.405	0.606	0.947	17.242	0.533	0.946	17.201	0.549	0.930	16.834	0.696
	B	2	1	0.951	20.742	0.585	0.941	9.331	1.434	0.945	10.375	0.417	0.946	10.305	0.409	0.948	9.931	0.980	0.948	9.942	0.999	0.950	9.397	1.256
		2	2	0.951	24.202	0.729	0.920	16.828	2.125	0.944	16.355	0.741	0.945	16.402	0.712	0.942	15.916	1.349	0.942	15.927	1.339	0.935	17.444	0.924
		5	1	0.951	16.347	0.573	0.948	11.352	1.645	0.950	13.300	0.493	0.949	13.015	0.480	0.948	12.702	0.859	0.948	12.695	0.854	0.945	10.123	1.085
		5	2	0.943	21.000	0.566	0.928	17.215	0.618	0.947	18.150	0.551	0.947	18.189	0.556	0.945	17.834	0.592	0.946	17.846	0.587	0.929	17.032	0.811
	C	2	1	0.971	64.572	3.868	0.944	33.156	12.743	0.946	20.825	2.876	0.947	21.600	3.763	0.946	29.458	9.689	0.946	29.376	9.649	0.989	34.245	5.619
		2	2	0.970	67.398	2.592	0.943	39.335	8.271	0.945	31.476	3.296	0.946	31.962	3.709	0.945	38.598	7.591	0.945	38.690	7.560	0.971	37.444	3.491
		5	1	0.972	60.322	3.074	0.966	46.122	4.863	0.948	33.885	2.438	0.949	34.111	2.435	0.944	39.506	2.123	0.945	39.562	2.142	0.990	37.592	2.449
		5	2	0.969	62.745	2.711	0.959	49.843	4.317	0.945	40.708	3.001	0.946	40.752	3.073	0.945	44.278	2.383	0.945	44.270	2.353	0.978	40.677	2.649
	D	2	1	0.948	25.787	0.807	0.941	12.407	0.459	0.947	13.666	0.716	0.947	13.570	0.746	0.949	13.111	0.441	0.949	13.118	0.449	0.974	16.189	2.875
		2	2	0.955	27.838	0.835	0.924	18.062	0.515	0.945	18.183	0.791	0.945	18.409	0.869	0.948	17.945	0.658	0.948	17.973	0.699	0.952	20.498	2.025
		5	1	0.939	61.413	2.517	0.935	45.229	1.194	0.951	78.902	3.170	0.950	75.048	2.418	0.950	69.329	2.184	0.950	69.331	1.899	0.951	25.642	3.615
		5	2	0.940	62.835	2.339	0.937	47.489	1.234	0.951	79.855	3.252	0.950	76.318	2.511	0.950	70.322	1.758	0.950	70.280	1.694	0.949	32.879	3.749

Once we have ensured that the methods produce PIs that attain the desired coverage, the best way to compare their performance is to look at the lengths of the intervals, the shorter the better. Table 2.1 includes the mean lengths. But to visually compare results of how the methods measure to one another and also make the results more comparable across scenarios, we present Figures 2.1 to 2.12. The measure used is the percentage increase in PI length of a method with respect to the best performer for a given run. This measure provides an easy and straightforward way to compare as well as aggregate results across all scenarios. For a given run, we have the mean length (on the test set) of the PIs for each method. Let ML_m be the mean length of method m, $m = 1, 2, \ldots, 7$, for this run. Let $\min_m\{\mathrm{ML}_m\}$ be the smallest mean length for this run. The percentage increase in PI length of method m with respect to best performer for this run is defined to be

$$100\frac{(\mathrm{ML}_m - \min_m\{\mathrm{ML}_m\})}{\min_m\{\mathrm{ML}_m\}}.$$

Hence, for a given run, one method will be a percent increase of 0 (the best performer for this run), and all other methods will have a positive percent increase. Hence, the smaller it is, the better is the method. Figures 2.1 to 2.12 present the distribution, across the 100 runs, of this measure for all 48 scenarios. The box-plots show, in order, the distribution of this measure for FMRLASSO, HDR1, NP1, NPK, SP1, SP2, and FMRFLEX. However, to make the comparisons fair and meaningful, in these figures, whenever the mean coverage of the method did not reach .93 for a given scenario, the box-plot representing it was removed. In fact, the methods that did not maintain their level in a given scenario were removed right from the start and were not used to compute the percent increase measure. Hence, the best performer is always a method that maintained its level for the specific scenario.

Looking at the figures, we can see that there is no one clear winner. Each method performs well in some scenarios, and not as well in others. As expected, when it is able to maintain its level, FMRLASSO is the best method for DGP A, where we have a mixture of linear models. The other DGPs either involve non-linear components or components with interactions. Either way, FMRLASSO, that uses only the main effects, can not be well specified, and this hurts its performance with longer PIs in most cases, even though it can be close to some of the RF based methods in some scenarios. The four BOP methods offer as good or better performances than the others in a vast majority of scenarios. They are not performing as well as the FMRLASSO, HDR1

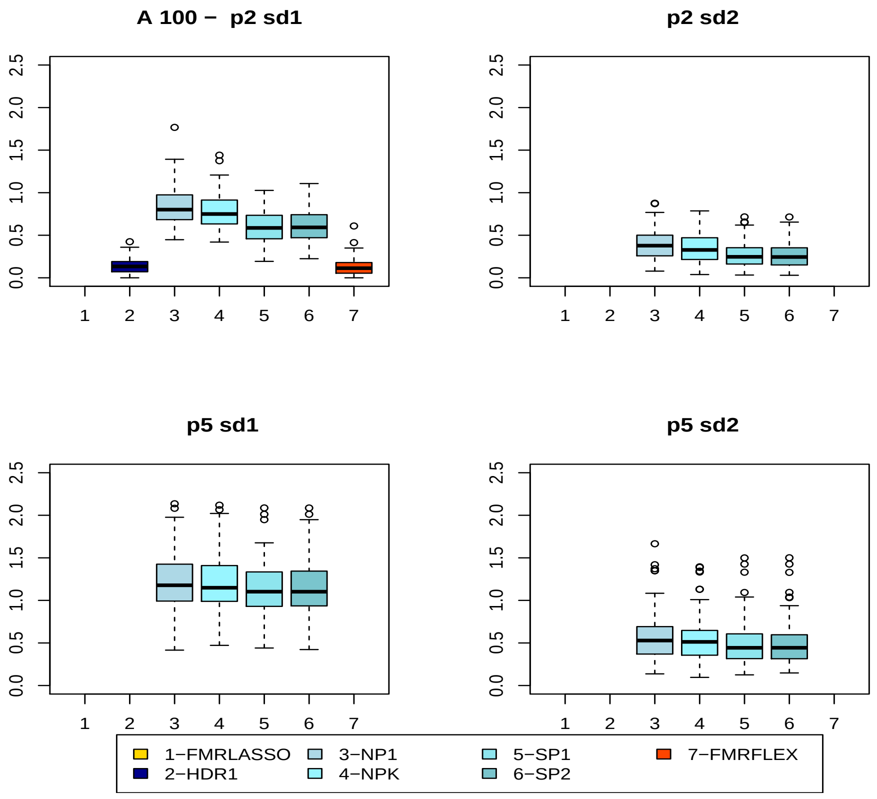


Figure 2.1: Distributions of the percentage increase in PI length with respect to best performer for each run for DGP A, training sample size $n = 100$.

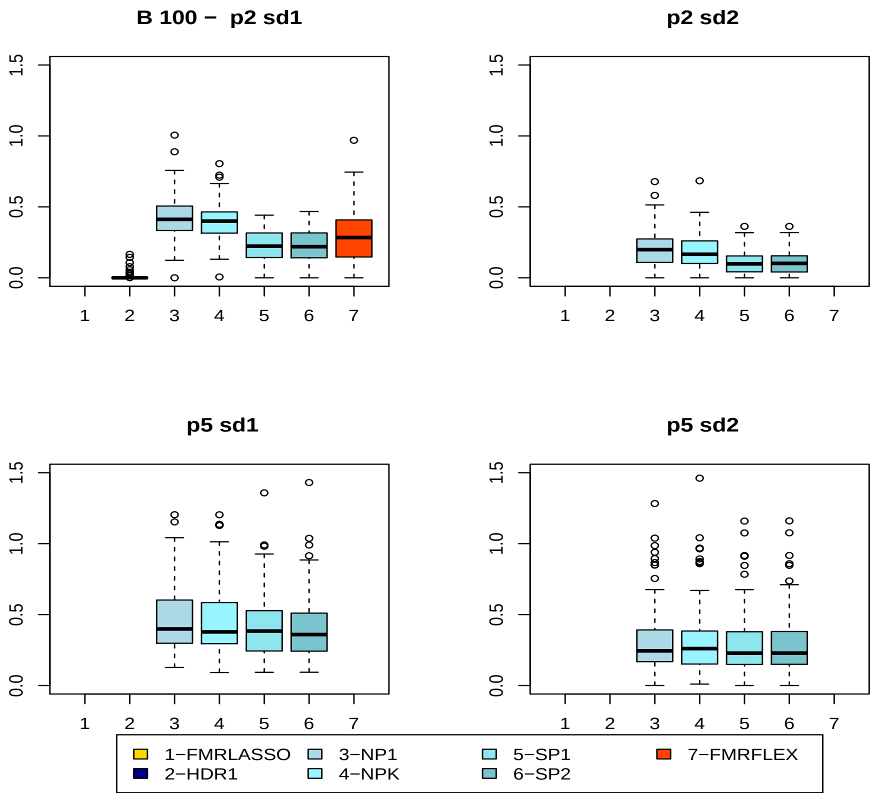


Figure 2.2: Distributions of the percentage increase in PI length with respect to best performer for each run for DGP B, training sample size $n = 100$.

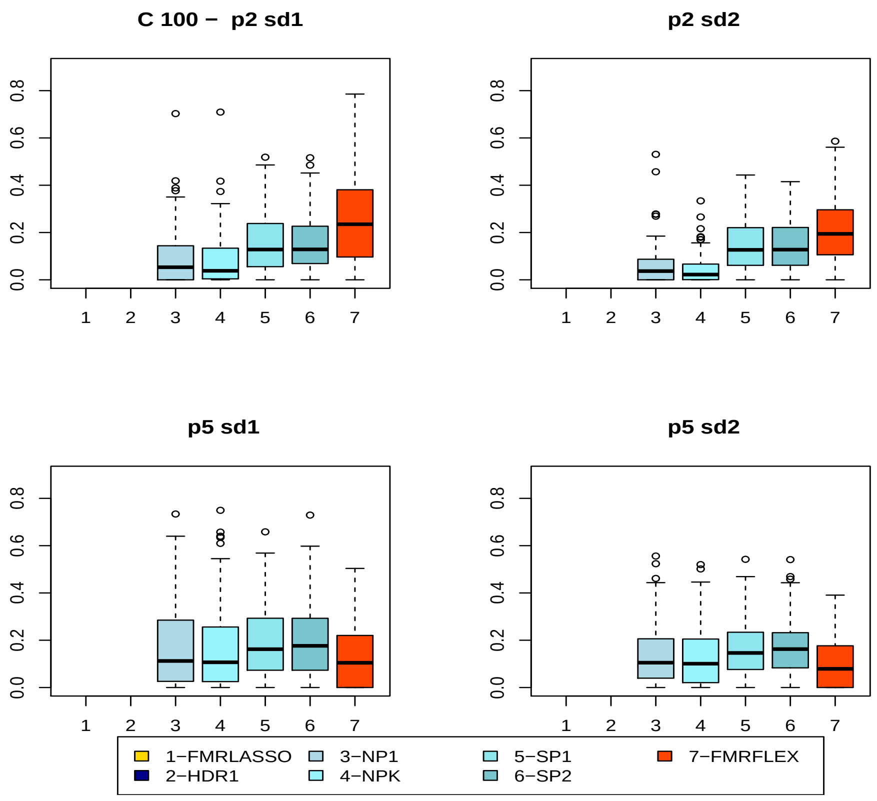


Figure 2.3: Distributions of the percentage increase in PI length with respect to best performer for each run for DGP C, training sample size $n = 100$.

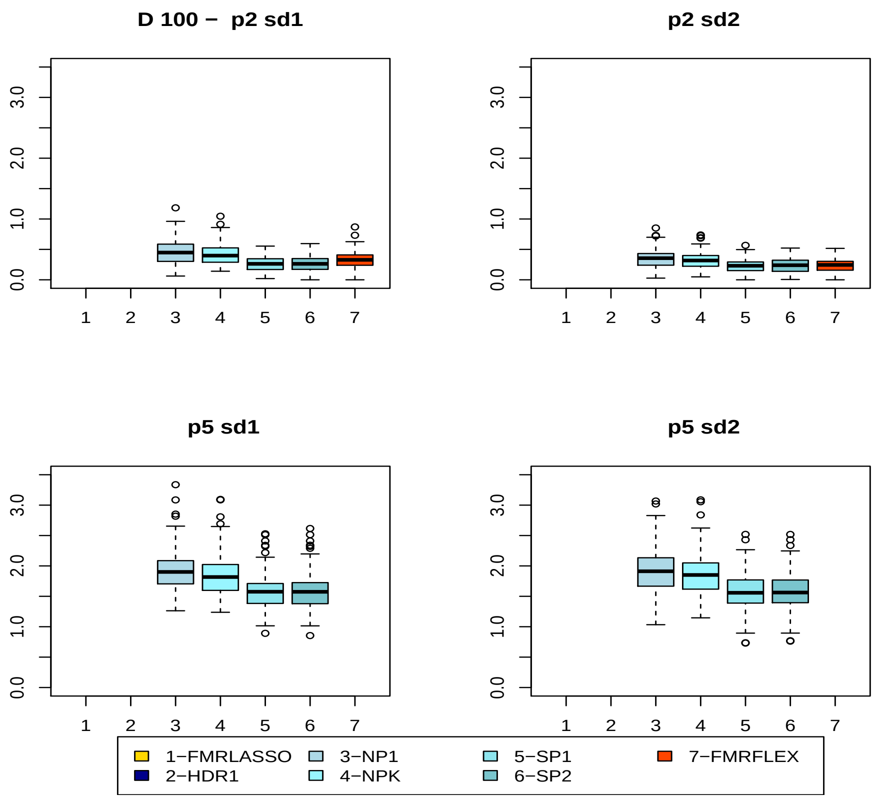


Figure 2.4: Distributions of the percentage increase in PI length with respect to best performer for each run for DGP D, training sample size $n = 100$.

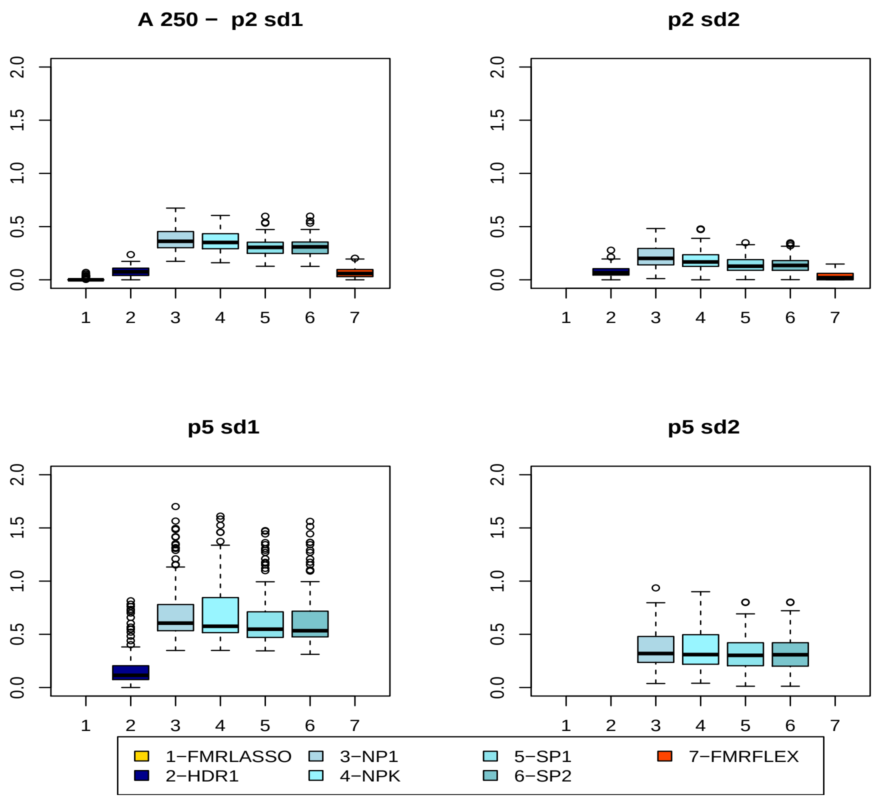


Figure 2.5: Distributions of the percentage increase in PI length with respect to best performer for each run for DGP A, training sample size $n = 250$.

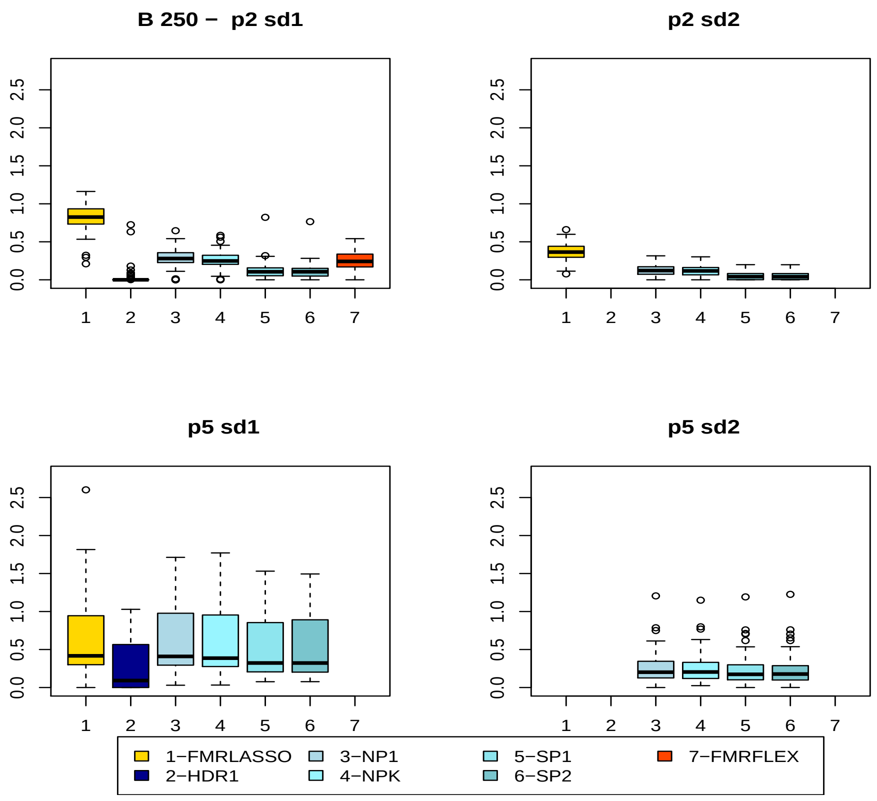


Figure 2.6: Distributions of the percentage increase in PI length with respect to best performer for each run for DGP B, training sample size $n = 250$.

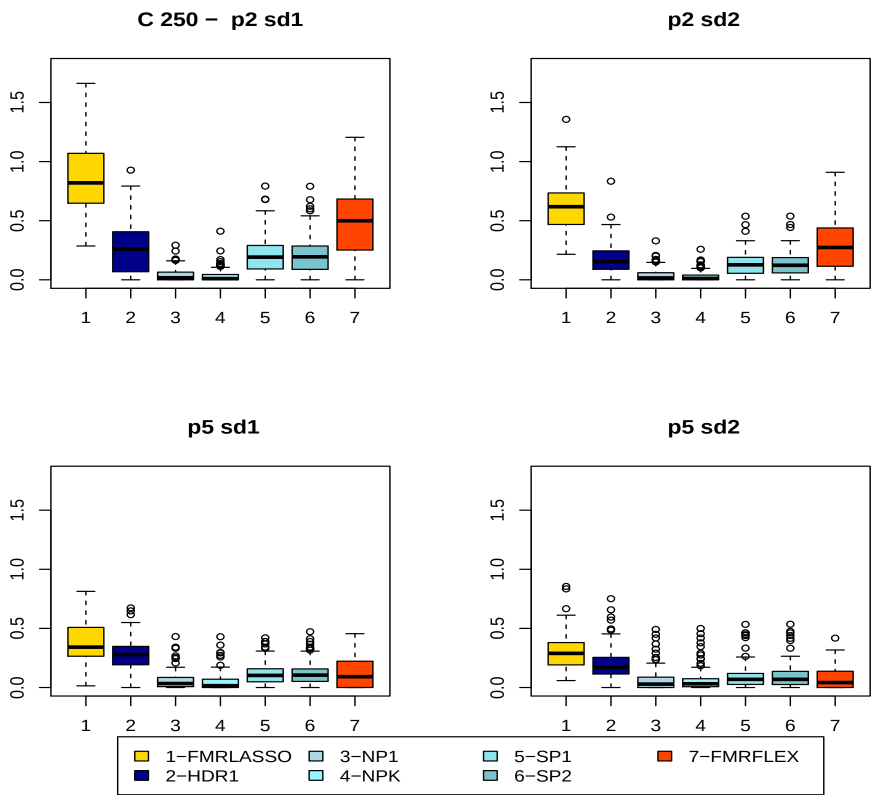


Figure 2.7: Distributions of the percentage increase in PI length with respect to best performer for each run for DGP C, training sample size $n = 250$.

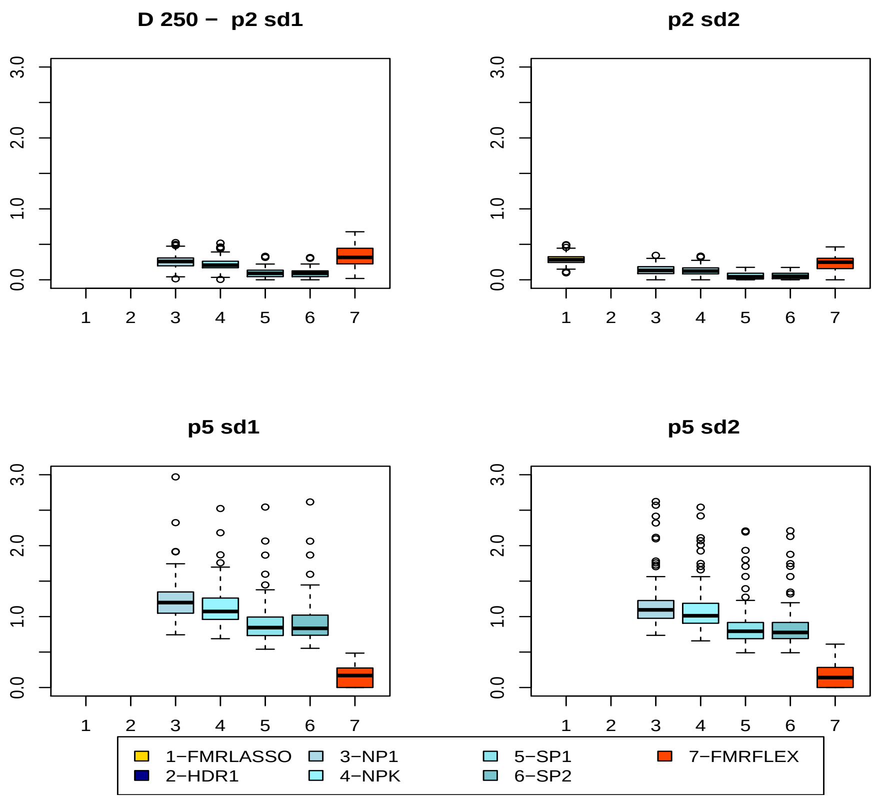


Figure 2.8: Distributions of the percentage increase in PI length with respect to best performer for each run for DGP D, training sample size $n = 250$.

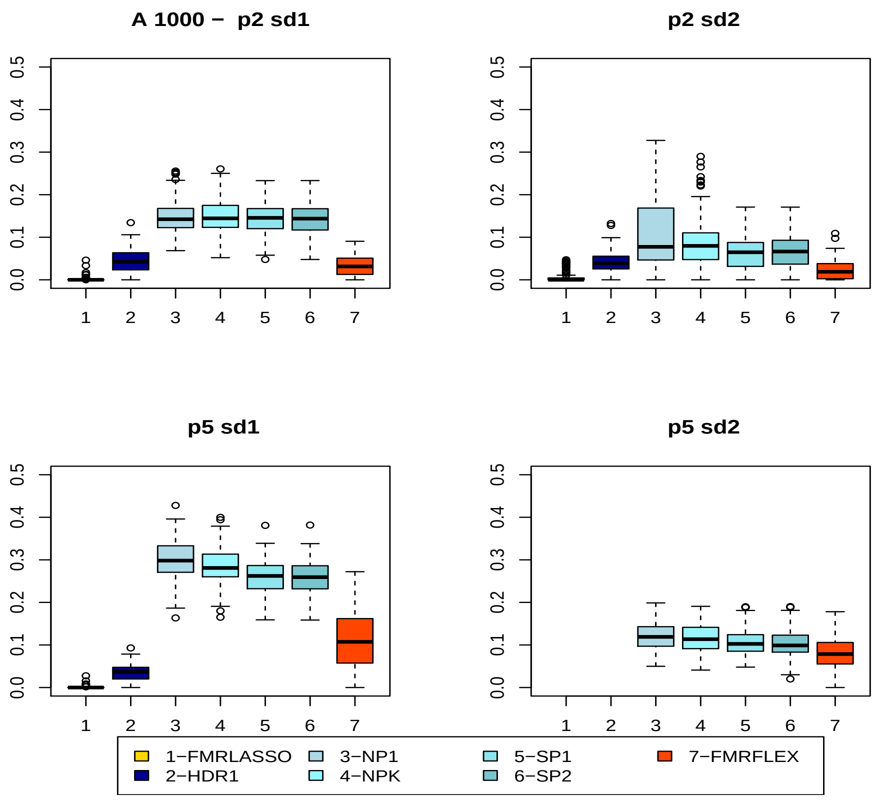


Figure 2.9: Distributions of the percentage increase in PI length with respect to best performer for each run for DGP A, training sample size $n = 1000$.

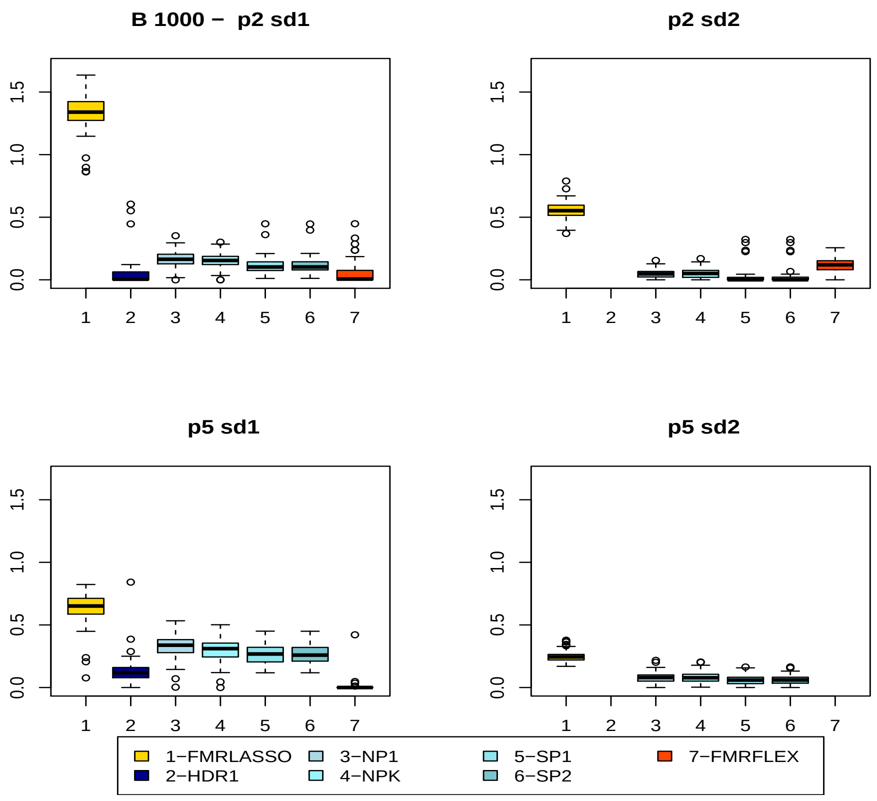


Figure 2.10: Distributions of the percentage increase in PI length with respect to best performer for each run for DGP B, training sample size $n = 1000$.

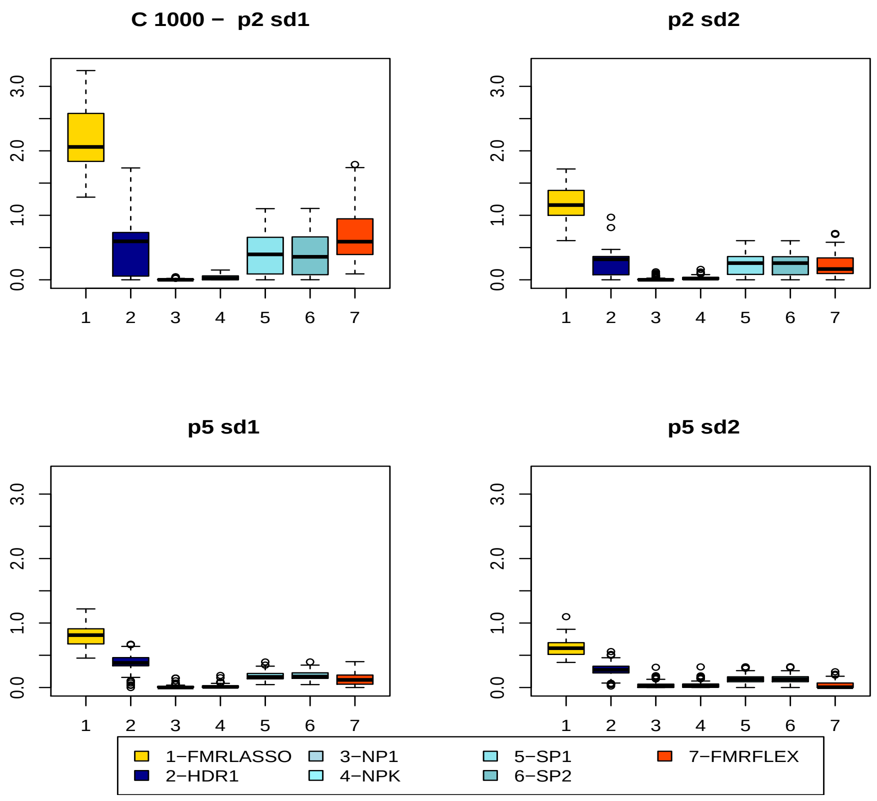


Figure 2.11: Distributions of the percentage increase in PI length with respect to best performer for each run for DGP C, training sample size $n = 1000$.

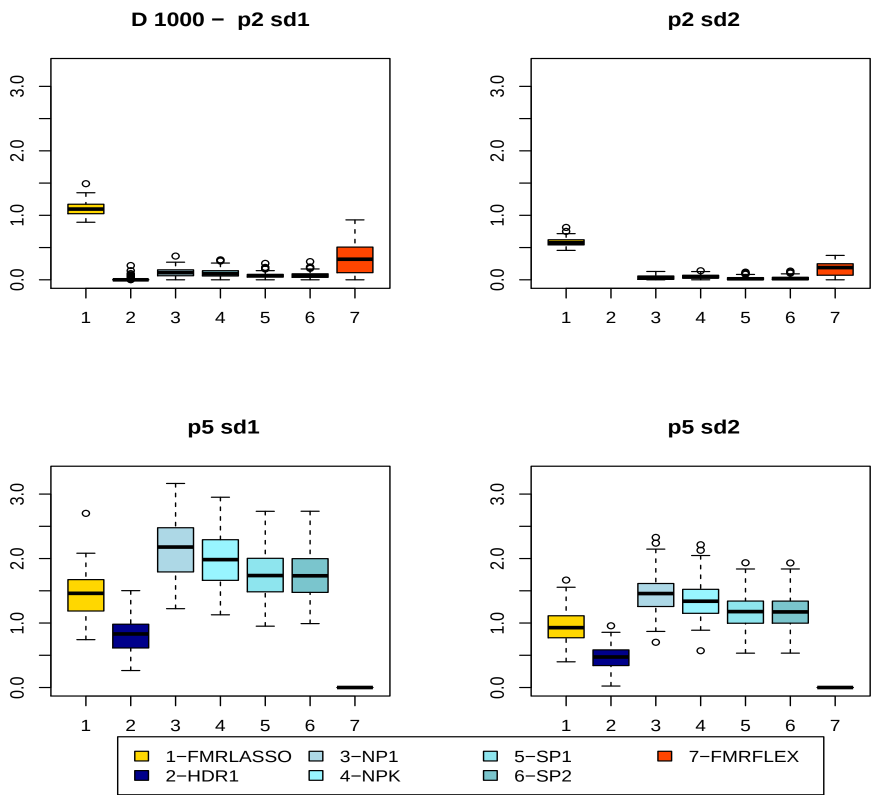


Figure 2.12: Distributions of the percentage increase in PI length with respect to best performer for each run for DGP D, training sample size $n = 1000$.

or FMRFLEX, in the two larger sample sizes with DGPs A, $p = 2$, and $p = 5$ and in the same sample sizes with DGP D and $p = 5$. HDR1 and FMRFLEX show very good performances in most of the scenarios where they reach the desired coverage.

If we look more closely at the four BOP methods, we see that SP1 and SP2 offer a very similar performance for all scenarios. These two methods build PI with a parametric HDR. SP1 estimates the parameters of the mixture from the individual class BOPs, while SP2 re-estimates the mixture distribution altogether from the adjusted (combined) BOP. Consequently, SP1 uses always the same estimated mixing proportions (the global ones), while SP2 re-estimates the mixing proportions for each adjusted BOP. In this simulation study, both approaches provided similar results. In a case where the mixing proportions are also a function of the covariates, we may expect SP2 to pick it up and use this information to provide better PIs. However, such settings were not considered in this study. The other two BOP methods, NP1 and NPK, performed similarly to each other. Both are based on a nonparametric HDR and the only difference is that NPK is limited to K intervals, whereas the number of intervals is not bounded with SP1. At first glance, one might have thought that SPK would produce longer PIs because it simply fills the gaps in the SP1 intervals to reach the maximum allowed number of intervals. But the OOB calibration procedure provides the necessary adjustment and both methods perform similarly. A similar conclusion was reached in Roy and Larocque [20]. In fact, SPK even performs slightly better than SP1 in some scenarios (e.g the two lower plots in Figure 2.12). We might conjecture that bounding the number of intervals performs some sort of regularization that improves the performance. If we compare the two nonparametric-HDR BOP methods (NP1, NPK) with the two parametric-HDR ones (SP1, SP2), then we see a bit more differences, with no winner. In most scenarios, they are close, and in a few ones NP1/NPK is better than SP1/SP2 and in some it is the other way around.

2.4.1 Computational Efficiency

Another important aspect of performance to investigate is computational speed. To measure this, we used the computational time of the FMRLASSO as the benchmark. Figure 2.13 shows how many times more, on average, a method takes to compute compared to the FMRLASSO. We computed this for the five proposed methods and for the FMRFLEX by averaging computational time for the methods for all scenarios of each sample size belonging to a DGP.

From this figure we can clearly see that the FMRFLEX and SP2 require more computational time than the other methods even with small sample sizes and they both become quite computationally intensive as the sample size grows. For FMRFLEX, the reason is that the fmrlasso becomes computationally demanding when the number of covariates is large, which is the case with FMRFLEX because a large number of dummy covariates are added. For SP2, the reason is that we have to fit a mixture model separately for each observation. On the other hand, the four other methods we propose maintain reasonable computational requirements for all sample sizes. In fact, they are only slightly more demanding than the FMRLASSO.

2.5 Conclusion

Finite mixture models are a useful and flexible tool for statistical modeling when there is heterogeneity in the data. In a general predictive context, a predictive interval is often more informative about the possible location of the response for a new subject, compared to a single point prediction. For FMR, this is even more the case because it is not clear that a single point is a meaningful prediction, especially for a well separated multi-modal distribution. A classical FMR based on linear models can be used to build PIs, but the *a priori* assumed parametric link between the covariates and the response needs to be well specified for the method to reach its full potential. A very flexible PI building method, called FMRFLEX, based on random forests was proposed in Ahonen et al. [1]. FMRFLEX worked very well in the settings studied in the original paper. In the basic non-mixture case, other promising PI building methods based on RF were proposed in Roy and Larocque [20]. The goal of this paper was thus to pursue the investigation of the use of RF to build PIs with FMR by 1) proposing a general way to build a mixture of RFs, 2) extending the Roy and Larocque [20] methods to mixture data, 3) studying the behavior of FMRFLEX and the new methods in more challenging settings, including smaller sample sizes and heteroscedastic data. A special attention was devoted to the computational time since it can become an issue with mixture data.

Here are some of the general findings. The four new methods based on the BOPs, used in conjunction with the OOB calibration method, are very reliable at maintaining the prescribed level. In fact, they were able to do it in all the 48 simulation scenarios considered, including sample sizes as low as 100 and with heteroscedastic data. The fmrlasso method, used without any adjustment, and the FMRFLEX and the new HDR1 methods, with the bootstrap adjustment of Ahonen et al. [1], all had more difficulties maintaining the prescribed level, especially with the smaller sample sizes and with heteroscedastic data. With respect to the lengths of the PIs, no method dominated the others. Each individual methods achieved a very good performance in many or a few scenarios. As expected, fmrlasso was especially good in the scenarios involving linear mixtures, but the RF based alternatives were very close to it. The FMRFLEX and the new HDR1 methods are generally good when they are able to maintain the prescribed confidence level. The four methods based on the BOPs, in addition to reliably maintaining the level, were also generally good. Moreover, four out of the five proposed methods (all but SP2), showed to have great computational efficiency, and were only slightly more time consuming than FMRLASSO and many times faster than FMRFLEX in many scenarios.

At the moment, all RF based methods can be recommend with large sample sizes. For smaller sample sizes, the four BOP methods should be used because they are able to reliably calibrate the PIs. However, among these four methods, we would recommend NPK and SP1. The reason is that NP1 and NPK are closely related and achieved a similar performance in the simulation, with perhaps a slight edge for NPK. Moreover, NPK controls the number of intervals in the PI, while NP1 does not. SP1 and SP2 are also closely related methods that achieved a similar performance, but SP2 is a lot more computationally demanding.

There are several interesting avenues for future research. First, the number of components in the mixture was assumed known in this work. It would be interesting to let the data select this number automatically. One obvious way to do it would be to build PIs for many different values of K and select one according to a criterion, for example, the value that produces the shorter PIs. Obviously this would need to be done carefully in order to avoid overfitting and to still maintain the prescribed level. A calibration method based on the OOB information similar to the one used in this paper could be designed for this problem. Second, since the FMRFLEX and HDR1 methods can be very good when they maintain the level, it would be interesting to investigate a better way to calibrate them. The current bootstrap adjustment used by them works well for large sample sizes but had problems with smaller ones. Since these two methods are not based on the BOPs, it is not obvious if it is possible to use the same calibration method based on the OOB BOP information, that works so well for the BOP methods. Third, it could be interesting to investigate even more challenging settings. In this paper we studied the performance for small samples and heteroscedastic data. But in addition to the error variances, the mixing proportions could also be a function of the covariates. In this case, the NP2 method, which re-estimates the mixing proportions for each new observation could provide an improvement that would justify its higher computational cost.

References

[1] Ahonen, I., Nevalainen, J., and Larocque, D. (2018). Prediction with a flexible finite mixture-of-regressions. *Computational Statistics & Data Analysis*.

[2] Benaglia, T., Chauveau, D., Hunter, D. R., and Young, D. (2009). mixtools: An R package for analyzing finite mixture models. *Journal of Statistical Software*, 32(6):1–29.

[3] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[4] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

[5] Galimberti, G., Montanari, A., and Viroli, C. (2009). Penalized factor mixture analysis for variable selection in clustered data. *Computational Statistics & Data Analysis*, 53(12):4301–4310.

[6] Hastie, T., Tibshirani, R., and Friedman, J. (2009). Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer.

[7] Hothorn, T., Lausen, B., Benner, A., and Radespiel-Tröger, M. (2004). Bagging survival trees. *Statistics in medicine*, 23(1):77–91.

[8] Huang, M., Li, R., Wang, H., and Yao, W. (2014). Estimating mixture of gaussian processes by kernel smoothing. *Journal of Business & Economic Statistics*, 32(2):259–270.

[9] Huang, M., Li, R., and Wang, S. (2013). Nonparametric mixture of regression models. *Journal of the American Statistical Association*, 108(503):929–941.

[10] Huang, M. and Yao, W. (2012). Mixture of regression models with varying mixing proportions: a semiparametric approach. *Journal of the American Statistical Association*, 107(498):711–724.

[11] Hyndman, R. J. (1996). Computing and graphing highest density regions. *The American Statistician*, 50(2):120–126.

[12] Khalili, A. and Chen, J. (2007). Variable selection in finite mixture of regression models. *Journal of the american Statistical association*, 102(479):1025–1038.

[13] Khalili, A., Chen, J., and Lin, S. (2010). Feature selection in finite mixture of sparse normal linear models in high-dimensional feature space. *Biostatistics*, 12(1):156–172.

[14] Leisch, F. (2004). FlexMix: A general framework for finite mixture models and latent class regression in R. *Journal of Statistical Software*, 11(8):1–18.

[15] Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

[16] Lin, Y. and Jeon, Y. (2006). Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590.

[17] McLachlan, G. and Peel, D. (2004). *Finite mixture models*. John Wiley & Sons.

[18] Moradian, H., Larocque, D., and Bellavance, F. (2017a). L_1 splitting rules in survival forests. *Lifetime data analysis*, 23(4):671–691.

[19] Moradian, H., Larocque, D., and Bellavance, F. (2017b). Survival forests for data with dependent censoring. *Statistical Methods in Medical Research*, page 0962280217727314.

[20] Roy, M.-H. and Larocque, D. (2018). Prediction intervals with random forests.

[21] Samworth, R. J. and Wand, M. P. (2010). Asymptotics and optimal bandwidth selection for highest density region estimation. *The Annals of Statistics*, 38(3):1767–1792.

[22] Städler, N. (2010). fmrlasso: Lasso for finite mixture of regressions. *R package version 1.0*.

[23] Städler, N., Bühlmann, P., and Van De Geer, S. (2010). $l-1$ penalization for mixture regression models. *Test*, 19(2):209–256.

[24] Xiang, S. (2014). *Semiparametric mixture models.* Kansas State University.

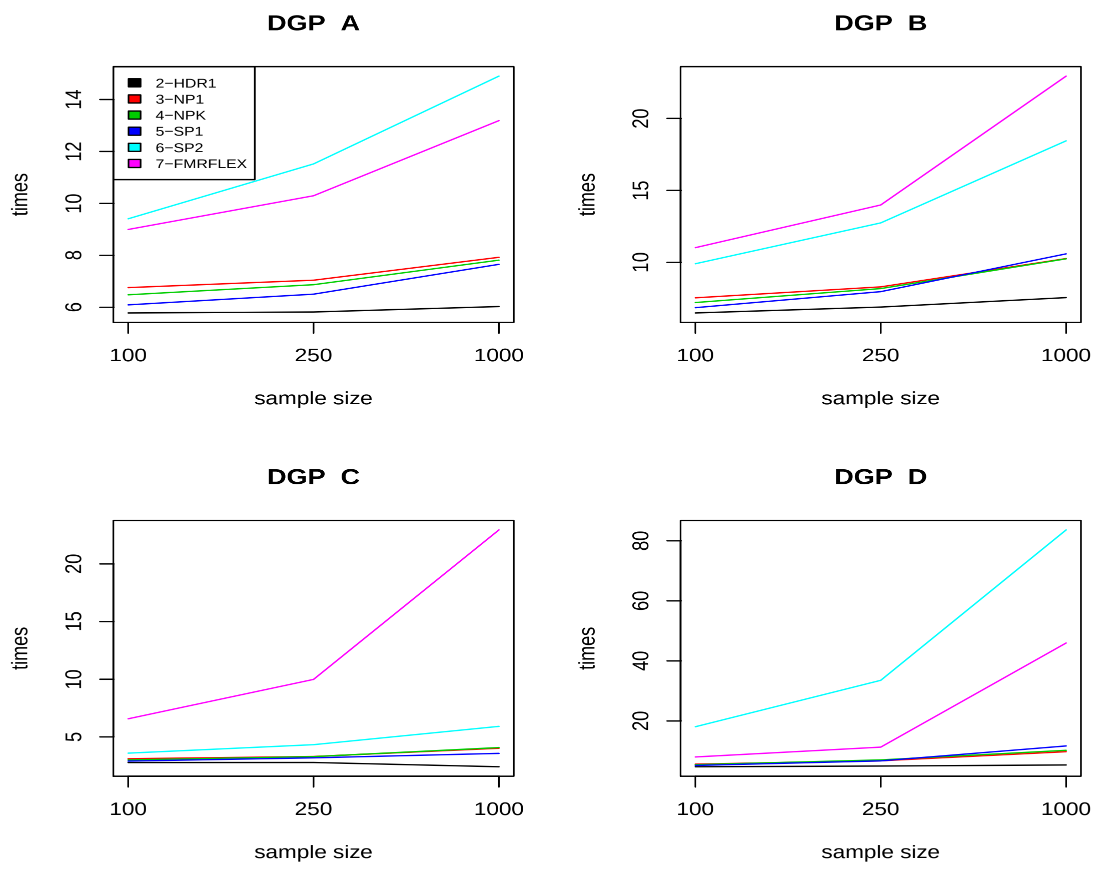


Figure 2.13: Number of times, on average, a method takes to compute compared to the FMRLASSO.

Chapter 3

High-Dimensional Variable Screening using a Robust Ensemble Method

Marie-Hélène Roy, Denis Larocque & Debbie Dupuis

Abstract

We propose a robust variable screening method for large high-dimensional data. The method is based on the very fast and robust variable selection technique by Dupuis and Victoria-Feser (2013). To produce a variable screening method, we combine this technique in an iterative ensemble scheme where we proceed to a safe and gradual elimination of covariates. This strategy insures the survival of the true model covariates through the procedure and into the final subset. The performance of the method is assessed in a simulation study on both clean and contaminated data and includes scenarios with a number of covariates up to 100,000. It is also applied to a real data set. The results show very good and consistent performance on both clean and contaminated data even with a very large number of covariates. Furthermore, the method offers built-in automatic final subset size determination. This constitutes an innovative feature for a variable screening method.

3.1 Introduction

Great advances in computing and technology in the last decade have facilitated the collection of data sets of ever growing size. Research in various fields such as health sciences, engineering, computer sciences as well as business and finance can benefit from these newly available data. The analysis of these large data sets also comes with statistical challenges. They are often high-dimensional and sparse, where the number of covariates p greatly exceeds the number of observations n and only very few of these covariates are considered to be truly relevant [5, 2]. In such cases, performing effective dimension reduction before pursuing further analysis is of foremost importance. Large data sets are also more susceptible to data contamination. The use of robust methods helps insure a better reliability of results. However, these methods often come at a computationally high cost [14], which makes it extra challenging to apply them to very large data sets. In this context, the need for dimension reduction methods that can be applied to large high-dimensional data sets and that are robust to data contamination is becoming increasingly important.

To facilitate variable selection when the number of covariates is very large, especially when $p >> n$, a preliminary major dimensionality reduction should be performed. This step is called variable screening. The main purpose of variable screening is to reduce the number of variables at least below sample size without discarding any true covariates. Thereafter, if needed, traditional model selection techniques can be readily applied.

In recent years, several variable screening methods for high-dimensional data have been proposed. Fan and Lv [6] initiate the screening approach with a novel proposal called Sure Independence Screening (SIS). SIS focuses on a quick dimensionality reduction with the use of “correlation learning". The marginal correlation between each covariate and the response, computed using the Pearson coefficient, is employed to rank the covariates. Subsequently the dimensionality is reduced to a prespecified size d. The authors suggest using a subset size $d = n-1$ or $d = n/log(n)$. The procedure is shown to achieve the sure screening property under certain conditions. This property implies that all the important variables survive after the screening step with a probability tending to 1. That is, the probability of including the true model in the final subset tends to 1 as $n \to \infty$.

Various SIS extensions have been proposed, for example, for the generalized linear and additive models [7, 8, 15]. Other extensions are screening via distance correlation learning [17], using the ranks of the generalized empirical correlations [13], model-free screening [28] and pairwise sure independence screening for linear discriminant analysis [23].

Among other popular approaches, Wang [27] shows that forward regression, when combined with the Extended BIC of Chen and Chen [1], can also be an interesting alternative for ultra-high dimensional screening. Mai and Zou [21] propose an innovative nonparametric method using the fused Kolmogorov filter to screen variables.

Although a number of methods can be used to perform high dimensional variable screening, the number of robust alternatives remains limited.

Gather and Guddat [10] suggest a simple robustification of SIS by replacing non robust estimators by their robust counterparts. That is, replacing the sample mean by the sample median, the standard deviation by the median absolute deviation and the correlation estimator with the Gnanadesikan-Kettenring (GK) robust correlation estimator [11]. They show that this simple robustified version of SIS improves performance in the presence of outliers in the response. The computational load of the GK estimator is however considerably higher than its non robust counterpart [22].

A second robust SIS variation, MCD-RoSIS, was proposed by Guddat et al. [12]. The method uses the Minimum Covariance Determinant (MCD) estimator by Rousseeuw [24] to provide a robust estimation of the correlation. In their simulation study, where both contamination in the response and in the covariates are introduced, the MCD is shown to be more robust than the correlation estimated with the GK.

Li et al. [17] introduce the Robust Rank Correlation Screening (RRCS). To provide robustness, their procedure uses the Kendall τ correlation coefficient. They show that their method possesses the sure screening property for linear regression models under the assumptions of Fan and Lv [6]. They also show through a simulation study that their method provides robustness against contamination in the response.

Mu and Xiong [22] propose two new robust screening methods based on SIS. The first replaces the sample correlation in SIS by the bivariate winsorized correlation estimator (BW). The second introduces a new measure called median of component-wise products (MCP). Like the original SIS, both methods rank variables and establish final selection according to a pre-specified threshold. BW and MCP show a better overall performance when compared to the GK estimator and the original non-robust SIS. Mu and Xiong [22] also compare computational times. The comparison includes the original sample correlation used by SIS, the GK correlation estimator, the BW correlation estimator and the MCP. Their two proposed methods, the BW and MCP, are shown to be faster than the GK correlation estimator used in Gather and Guddat [10] but, as expected, not as efficient in computation as the (non robust) original sample correlation in SIS.

Ma and Zhang [20] also propose a robust SIS variable screening procedure (QCS). The method is based on quantile correlation. Sun [26] introduces the Spearman correlation screening procedure and presents a simulation study that compares this method with the SIS, the RRCS and QCS in normally distributed scenarios as well as contaminated scenarios. He shows that the SIS and the QCS have the best performances in terms of variable screening in the non-contaminated scenarios and that his proposed method, the RRCS and the QCS had the best performances in contaminated scenarios.

All of these methods are based on ranking the variables and selecting the final subset according to a prespecified threshold. None offer a data-driven or automatic way to establish the size of the final subset. The choice of $d = n-1$ or $d = n/log(n)$ suggested by Fan and Lv [6] and used and recommended by other authors can be problematic. Because it is a function of the sample size, it can lead to an excessively small or large final subset size regardless of the initial number of covariates or their relative importance. Our method presents itself as an innovative robust variable screening alternative that does not require the choice of a prespecified final subset size.

We propose major dimensionality reduction by using the very fast and robust variable selection method of Dupuis and Victoria-Feser [4] in an ensemble scheme. This method, the robust variance inflation factor regression (robVIF), is a robust version of the variance inflation factor (VIF) regression of Lin et al. [19]. The VIF regression uses a streamwise regression approach. It sweeps through all covariates, testing each potential true covariate for inclusion in the model according to a test statistic based on the VIF. This one-pass search allows one to quickly identify and select the covariates that reduce a statistically sufficient part of the variance in the predictive model. One of the drawbacks of streamwise approaches is the impact of the covariate order of entry. To address this issue we combine the robVIF with an ensemble approach.

Our method, named Screening Ensemble Robust VIF (SERVIF), performs a gradual but massive elimination of spurious variables. To do that, we use what we call the *Two-step Elimination* procedure in an iterative fashion. For each iteration, the first step is to use the robVIF in an ensemble scheme. A robVIF is performed B times on the data thus allowing the procedure to be parallelized. Each time, the order of entry of the covariates in the streamwise procedure is permuted. This permutation has the advantage of offsetting robVIF's sensitivity to the covariates' order of entry. The results of the B robVIF trials are aggregated. These results provide valuable information on the relative importance of each covariate. We use this information to perform a first (for this iteration) conservative elimination of covariates as well as for ranking purposes. The second step of each iteration consists of a one-pass robVIF where a more stringent elimination of covariates is done.

Another important feature of the SERVIF is that each iteration is performed independently on subsets of covariates. At the beginning of each iteration, the active set of covariates (comprised of the initial or surviving covariates) is partitioned in R sets and the two-step elimination procedure is performed separately on every set. At the end of an iteration the surviving covariates are regrouped. At the last iteration, this subset of covariates represents the final selection. Thus, by selecting the final number of covariates according to their survival through the iterative process we remove the need for the delicate prior choice of d.

With this iterative framework, we obtain a robust variable screening method for sparse large high-dimensional datasets. By using a partitioning scheme, our method has the ability to work with a ultra-high number of covariates, reaching 100,000 or more, while maintaining good performance even in the presence of data contamination.

The remainder of the paper is organized as follows. Section 2 contains the methodology. The method's good performance is then showed in a simulation study in Section 3. Section 4 provides results for real data. Discussion and closing remarks are presented in Section 5.

3.2 Methodology

Let us consider a classical linear model,

$$\boldsymbol{y} = \boldsymbol{\beta_0} + \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{3.1}$$

with n observations $\boldsymbol{y} = (y_1, ..., y_n)'$. For each observation there are p potential true covariates $(x_1, ..., x_p)' = \boldsymbol{x}$ and $\boldsymbol{X} = [\boldsymbol{x_1}, ..., \boldsymbol{x_p}]$ is the np design matrix. The vector of coefficients is $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)'$ and the error $\boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \sigma^2 \boldsymbol{I_n})$.

The method is based on the robVIF. Accordingly, an overview of the key elements of the robVIF are presented.

3.2.1 Robust VIF Regression

The robust VIF is a forward streamwise regression method. It evaluates one covariate x_k, $k = 1, ..., p$, at a time for addition to the model, and then moves on to the next covariate. The decision to add x_k to the model or not is based on a rejection quantile that follows an α-investing rule.

An α-investing rule is an adaptive sequential procedure for multiple hypotheses testing [9]. For robVIF, the α-investing procedure starts with an initial wealth a_0 and test level α_k is set at $\alpha_k = a_k(1 + k - f)$ where f has an initial value of 0 and represents the last step k at which a covariate was included. When a covariate is added to the model, a pay-out Δa is given and added to the current wealth a_k. When a covariate is not added to the model, the current wealth a_k is reduced by $\alpha_k/(1 - \alpha_k)$. The procedure continues until $a_k \leq 0$.

To limit the influence of outliers, robVIF uses a weighted LS estimator of the form

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^{wT}\boldsymbol{X}^{\boldsymbol{w}})^{-1}\boldsymbol{X}^{wT}\boldsymbol{y}^{\boldsymbol{w}} \tag{3.2}$$

where $\boldsymbol{X}^w$ and $\boldsymbol{y}^w$ represent respectively a weighted design matrix and a weighted response vector. These weights are computed using Tukey's redescending biweight weights. Marginal weights, obtained by fitting p marginal models $y = \beta_{01} + x_1\beta_1 + \varepsilon_1, ..., y = \beta_{0p} + x_p\beta_p + \varepsilon_p$, are also computed. For these, for better computational efficiency, Dupuis and Victoria-Feser [4] have chosen to use Huber's weights. The purpose of the two types of weights is to limit the influence of extreme observations in the response and/or in the covariates on the value of the coefficients $\hat{\boldsymbol{\beta}}$.

For clarity purposes, we will divide the robVIF procedure into two main parts. The first part begins by computing the marginal weights necessary for the evaluation procedure of each covariate. The second part begins by testing all covariates one at a time through a robust evaluation procedure. If x_k is not added to the model, the method moves on to the next covariate. When a covariate is added to the model, the method performs additional computation to find the new coefficients of the revised model as well as update the value of the current wealth a_k before moving on to the next covariate. The procedure ends when all covariates have been tested or if $a_k \leq 0$.

```
Part 1  Initialize and Weight
Input  data $\boldsymbol{y}, \boldsymbol{x}_1, \boldsymbol{x}_2, ...$ (standardized)
Set  f=0, initial wealth $a_0$ and pay-out $\Delta a$
Get  All marginal weights using Huber's weights by fitting $p$ marginal models $y = \beta_{01} + x_1\beta_1 +$
        $\varepsilon_1, ..., y = \beta_{0p} + x_p\beta_p + \varepsilon_p$ with the weighted LS estimator in (2).
Part 2  Testing and Computation Loop
repeat  for $k = 1, ..., p$
    set  $\alpha_k = a_k/(1+k-f)$
    obtain  p-value for $x_k$ from marginal model.
    test  $x_k$ for inclusion
    if  $x_k$ is entered
        then
          Compute new $\hat{\beta}_0$ and $\hat{\boldsymbol{\beta}}$ for the updated model with the weighted LS estimator in (3.2)
              using Tukey's biweights.
          update $a_{k+1} = a_k + \Delta a$
          update $f = k$
    else  $a_{k+1} = a_k - \alpha_k(1-\alpha_k)$
    end if  $a_k \leq 0$
     $k = k+1$
```

3.2.2 The SERVIF Algorithm

We can now describe the proposed method. The algorithm is described first, then detailed explanations of the major steps are given.

The Algorithm

1. Input data $\boldsymbol{y}, \boldsymbol{x}_1, \boldsymbol{x}_2, ...$ (standardized), set iteration $i = 1$ and $q_i = p$ the number of active covariates (the initial or surviving covariates). Let I_i be the index set of active covariates at iteration i. Set value of the partition size c. If $q_i < c$ then $c = q_i$.

2. Carry out *Part 1: Initialize and Weight* of the robVIF on all covariates to obtain marginal weights.

While $q_i > c$ and $q_i < q_{i-1}$

{

3. Randomly partition the q_i covariates in R_i near-equal sets of size c or less, where $R_i = \lceil q_i/c \rceil$. Let $I_{i,j}$ be the index set of the $q_{i,j}$ covariates contained in the j^{th} set, $j = 1, ..., R_i$.

4. Perform the following *Two-step Elimination* procedure independently on each set of covariates.

<u>I - Ensemble step</u>

a) Generate B classic bootstrap samples (n-out-of-n observations with replacement) and randomly permute covariates in each sample. One also has the option to only permute the covariates and use the original sample. That is B permutations are considered and no bootstrap is performed. The index set of the b^{th} permuted sample is $I^b_{i,j}$.

b) Perform *Part 2: Testing and Computation Loop* of the robVIF on each sample.

c) Compute the inclusion frequencies, that is the number of times each covariate is retained as a potential true covariate throughout the B samples.

d) Eliminate covariates with inclusion frequency equal to 0. The index set of the $q^*_{i,j}$ surviving covariates is $I^*_{i,j}$.

e) Compute mean of p-values over the B samples on the surviving covariates, $\bar{\boldsymbol{p}}_{i,j} = (\bar{p}_{i,j_1}, ..., \bar{p}_{i,j_{q^*_{i,j}}})$.

f) Rank the $q^*_{i,j}$ covariates according to mean p-value, $\bar{p}_{i,j_k}$, $k = 1, \ldots, q^*_{i,j}$, in ascending order.

<u>II - RobVIF step</u>

a) Perform a one pass robVIF model selection on the $q^*_{i,j}$ ranked covariates. The index set of the $q^{**}_{i,j}$ selected covariates is $I^{**}_{i,j}$.

5. Regroup covariates selected from the R_i sets. The total number of active covariates is now $q_{i+1} = \sum_{j=1}^{R_i} q_{i,j}^{**}$ and the index set of active covariates is $\cup_{j=1}^{R_i} I_{i,j}^{**} = I_{i+1}$.

6. Set $i = i+1$

}

7. Perform a partial *Ensemble Step* once on the covariates in the final active set $I_i = \{i_1, \dots, i_{q_i}\}$.

   a) Generate B classic bootstrap samples (n-out-of-n observations with replacement) and randomly permute covariates in each sample.

   b) Perform *Part 2: Testing and Computation Loop* of the robVIF on each sample.

   c) Compute mean of p-values over the B samples, to get $\bar{p}_{i_k}$, $k = 1, \dots, q_i$.

   d) Rank covariates according to mean p-value, $\bar{p}_{i_k}$, in ascending order.

More detailed explanations of the algorithm are now provided. To gradually eliminate unwanted covariates without discarding any true covariates, our screening approach uses an iterative procedure. Additionally, in each iteration the covariates are partitioned. This partitioning of covariates allows the method to handle a very large p while maintaining a good performance. It also diminishes computational time.

The first step of the algorithm is to carry out *Part 1-Initialize and Weight* of the robVIF on all covariates to obtain marginal weights. For computational efficiency, the marginal weights are calculated only once for each covariate and will be used throughout the procedure.

The Two-step Elimination procedure consists of the *Ensemble Step* followed by the *robVIF Step*. Both steps are performed independently on each set of covariates in the partition.

The first step of the *Ensemble Step* is to generate B samples from the data. Each sample is generated by resampling (with replacement) the n observations and randomly permuting the covariates. The original sample can also be used B times with randomly permuted covariates in each instance. In this ensemble scheme, presenting the covariates in a different order B times has the advantage of offsetting robVIF's sensitivity to the covariates' order of entry.

The second part of the robVIF, *Testing and Computation Loop* is performed on each sample. We use the inclusion frequency of each covariate to perform a first elimination. Covariates that have not been retained at least once throughout the B samples are discarded.

Step II of the *The Two-step Elimination* procedure, the *robVIF Step*, consists of performing a unique robVIF model selection on the $q^*_{i,j}$ surviving covariates ordered according to their respective mean p-values, $\bar{p}_{i,j_k}$, obtained in 4(e). Ranking the covariates according to the mean p-value helps insure that the potential true covariates are tested first, at the highest α levels, and therefore have a higher probability of being added to the model. This *robVIF step* leads to a considerable elimination of covariates.

After this elimination, the surviving covariates from all sets in the partition are regrouped. Steps 3 to 6 are repeated until either one of the following two conditions (the while loop conditions) is not met: The size of the active set of covariates is greater than the partition size, $q_i > c$ or at least one covariate is eliminated at iteration i, $q_i < q_{i-1}$. When a condition is not met, the procedure moves on to its final step.

The last step of the proposed method consists of doing a partial *Ensemble Step* on the covariates in the final set. Its objective is to rank the final selection in order of importance according to the mean p-value.

Wealth Attribution

The pay-out Δa is fixed to 0 throughout the procedure. The rationale is simple, the ensemble framework with the random permutation for covariates in each sample gives multiple opportunities for every covariate to be presented in different positions, thus at various α test levels. A positive pay-out Δa, which leads to a momentary increase of the test level α, is not justified and thus Δa is set to 0 throughout the procedure.

The wealth attribution structure in the *Ensemble Step* has been slightly modified from the original streamwise setup of the robVIF. We wanted to ensure that each covariate had 95% probability of being tested for inclusion as a potential true covariate at the initial α test level in at least one of the B samples. Recall that when pay-out $\Delta a = 0$, if x_k is included, $a_{k+1} = a_k$, the current wealth does not change, otherwise, if x_k is not included, $a_{k+1} = a_k - \alpha_k/(1-\alpha_k)$, the current wealth decreases. Hence, to ensure the 95% probability of being tested for inclusion at the initial α level for each x_k, the wealth has been fixed for the $\lceil p * g \rceil$ first covariates. The choice of g was done following the binomial distribution. If S is the number of times that a covariate is tested at the initial α level over the B samples, then $S \sim Bin(B, g)$, and we want to obtain a minimum of one success (i.e. $S \geq 1$) with cumulative probability of 95%. Thus the wealth a_k is fixed for the first $\lceil p * g \rceil$ covariates. Thereafter, at $k = \lceil p * g \rceil$+1 we resume the wealth attribution original streamwise setup of the robVIF with $\alpha_k = a_0/(1+k-f)$.

3.3 Simulation Study

To assess the performance of the proposed screening method we carry out a simulation study. We compare the performance of the SERVIF with the original SIS, the robustified version of the SIS using Kendall tau (SIS-Kendall) proposed by Li et al. [17] and the robust version (SIS-BW) proposed by Mu and Xiong [22].

For the implementation of our method, the R language code for the robVIF was obtained from the authors. The main structure of their code is a *for* loop that sweeps through all covariates, one at a time. This code is computationally very fast, especially when considering a sole pass. However, considering our methods' iterative structure that is moreover combined in an ensemble scheme, a modification is required to make our method computationally efficient. The code, except for the initialization, is converted to the more efficient Fortran programming language. This resulted in substantial gains in computational speed. We use this implementation of the method for the simulations presented in this section. Regarding the implementation of the competing methods, we use functions obtained from the “SIS” CRAN package [25]. The two robust competitor methods included in our study, SIS-Kendall and SIS-BW, are easily integrated into the SIS package code as they only require a different calculation for the correlation. For each of these correlation estimators, the Kendall τ correlation coefficient and the bivariate winsorized correlation estimator, the fastest available R function at the moment of the study are used for this simulation.

The Data Generating Process (DGP) used is the linear model employed in Dupuis and Victoria-Feser [3] and Dupuis and Victoria-Feser [4]. We have

$$y = X_1 + X_2 + \ldots + X_k + \sigma\varepsilon \tag{3.3}$$

where $X_1, X_2, \ldots, X_k$ are multivariate normal (MVN) with $E(X_i) = 0$, $Var(X_i) = 1$, $corr(X_i, X_j) = \theta$, $i \neq j, i, j = 1, \ldots, k$ and ε an independent standard normal variable. The covariates $X_1, \ldots, X_k$ are our k target covariates.

θ was chosen to produce a theoretical $R^2 = (Var(y) - \sigma^2)/Var(y) = 0.20$. The value of σ was chosen to give t values for the target regressors of about 6 under normality according to Khan et al. [16]. Let $e_{k+1}, \ldots, e_p$ be independent standard normal variables.

The next $2k$ are noise covariates that are correlated with our target covariates:

$$\begin{aligned}
X_{k+1} &= X_1 + \lambda e_{k+1}, X_{k+2} = X_1 + \lambda e_{k+2}, \\
X_{k+3} &= X_2 + \lambda e_{k+3}, X_{k+4} = X_2 + \lambda e_{k+4}, \\
&\vdots \\
X_{3k-1} &= X_k + \lambda e_{3k-1}, X_{3k} = X_k + \lambda e_{3k};
\end{aligned}$$

where $\lambda = 3.18$ so that $corr(X_1, X_k + 1) = corr(X_1, X_{k+2}) = corr(X_2, X_{k+3}) = \ldots = corr(X_k, X_{3k}) = 0.3$.

The last $p-3k$ covariates are independent noise covariates

$$X_i = e_i, i = 3k+1, ..., p. \tag{3.4}$$

Samples with and without contamination are generated. In the non-contaminated case $\varepsilon \sim N(0,1)$.

For the samples with contamination we introduce 5% of contaminated cases. These cases are contaminated with both outliers and high leverage X-values. These samples are generated with $\varepsilon \sim 95\%N(0,1) + 5\%N(30,1)$ and for the contamination in the X direction, $X_1, ..., X_k$ follows a MVN as before except that $Var(X_i) = 5$, $i = 1, ..., k$.

We run simulations for n = 1000, k = 5, and p ranging from 100 up to 100,000 for both non-contaminated and contaminated settings.

The partition size c is chosen to be 100 in all instances. The final number of covariates selected, d, has to be chosen for the SIS and the two robust SIS variations. The value of d is chosen to be $d = n/log(n)$ as suggested by Fan and Lv [6]. In all instances the results are based on 200 simulations.

In the non-contaminated setting, our method and its three competitors all succeed in selecting the five target covariates for all p dimensions. The only difference between the methods is the value of d. For the SIS and the two robust SIS versions, with $d = n/log(n)$ being a function of n only, for all scenarios the final number of variables with these three methods is $d = 144$.

For the contaminated scenarios, as expected, the original SIS performs poorly. It misses most target covariates for the majority, if not all, simulations. Thus, for the contaminated scenarios only SIS-Kendall and SIS-BW are considered as competitors.

For the contaminated scenarios, our method has no missing target covariates in any of the simulations for $p = 1000$ and 10,000. With $p =$ 25,000, our method has one missing target covariate in 1% of the simulations. For the $p =$ 50,000 scenario, we have one missing target covariate in 3%

of the simulations. The SIS-Kendall and SIS-BW variations have no missing target covariates in all cases.

The most significant difference between the methods is the number of covariates selected. Our method only retains the covariates that are identified as meaningful according to their p-value. As the number of spurious covariates highly correlated with the response increases with the number of initial covariates, the size of the final subset selected by our method also increases.

The results are presented in Figures 3.1 and 3.2, and Tables 3.1 and 3.2. Tables 3.1 and 3.2 present the average number of covariates in the final subset for non-contaminated and contaminated scenarios. Figures 3.1 and 3.2 show the distribution of the number of covariates included in the final subset for the SERVIF and the number of covariates included in the final subset for its competitors.

Table 3.1: Mean number of covariates in the final subset - non-contaminated scenario, $n = 1000$, $p = 1000$ to 100,000 for 200 simulations.

Mean number of covariates in the final subset					
p	1000	10,000	25,000	50,000	100,000
SERVIF	20.11	58.29	82.86	98.58	128.25
SIS and robust SIS variations	144	144	144	144	144

Table 3.2: Mean number of covariates in the final subset - contaminated scenario, $n = 1000$, $p = 1000$ to 100,000 for 200 simulations.

Mean number of covariates in the final subset					
p	1000	10,000	25,000	50,000	100,000
SERVIF	27.27	71.92	85.13	89.11	92.61
SIS and robust SIS variations	144	144	144	144	144

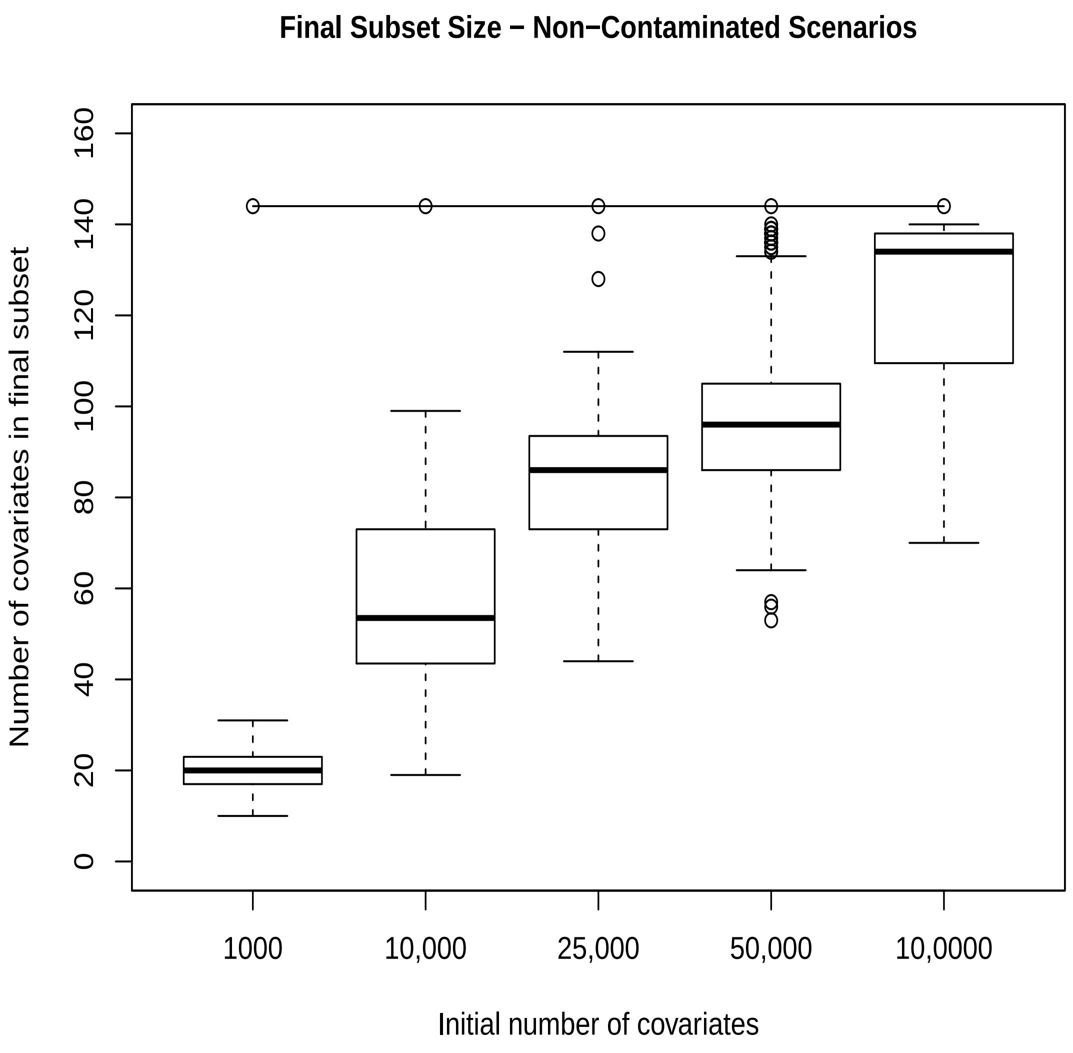


Figure 3.1: Number of selected covariates for non-contaminated scenarios for $n = 1000$, $p = 1000$ to 100,000, for 200 simulations. The boxplots show the distribution of the number of covariates for the SERVIF and the line, the number of selected covariates by the SIS and the two robust SIS variations.

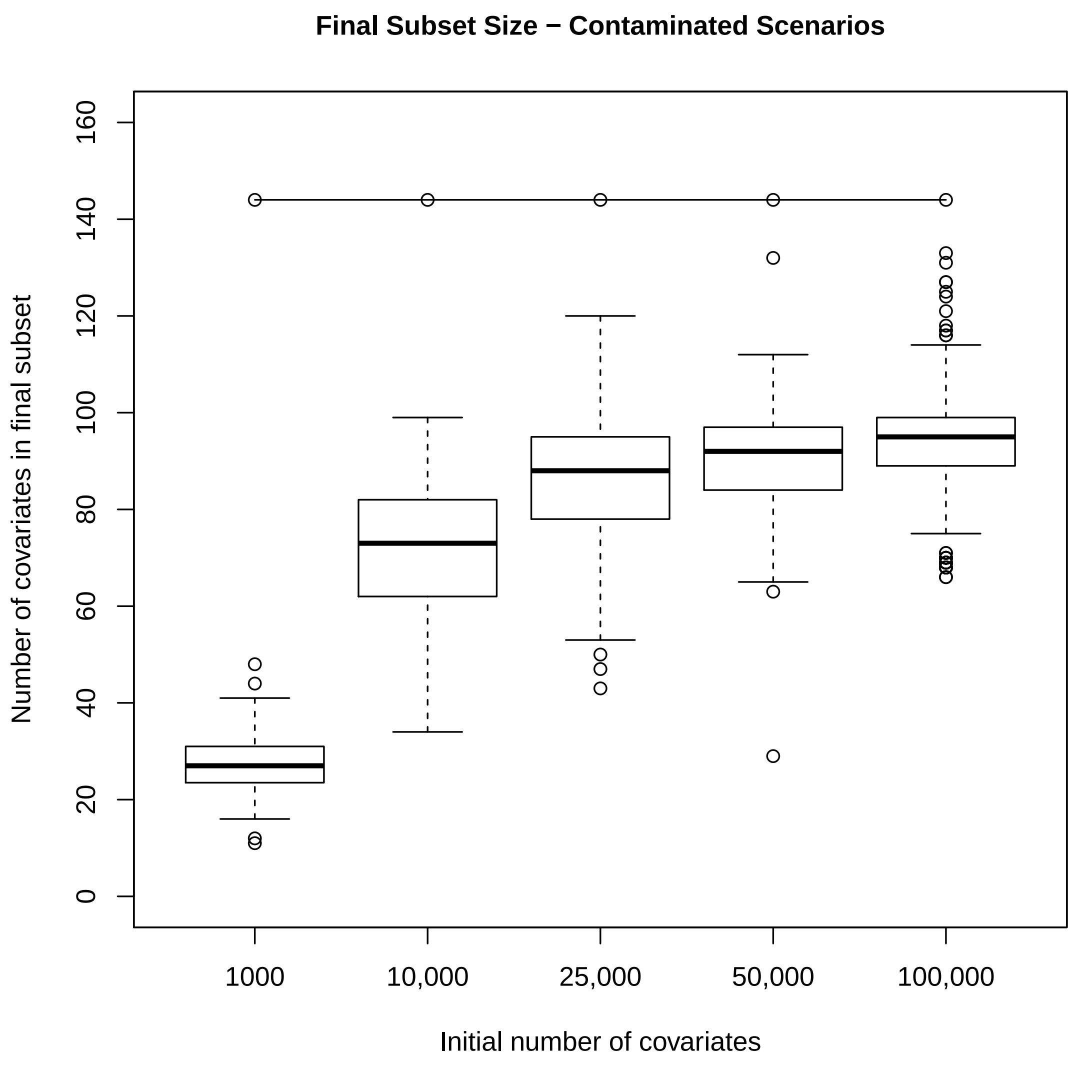


Figure 3.2: Number of selected covariates for contaminated scenarios for $n = 1000$, $p = 1000$ to 100,000, for 200 simulations. The boxplots show the distribution of the number of covariates for the SERVIF and the line, the number of selected covariates by the SIS and the two robust SIS variations.

3.4 Real Data

In this section we illustrate the use of the SERVIF on a real data set also used in Dupuis and Victoria-Feser [4]. We consider the *Communities & Crime* data set from the UCI Machine Learning Repository [18]. The data set has 1994 observations, which are split into $n = 1000$ for the training set and $n = 994$ for the test set. The response variable of the data set is the per capita violent crimes rate. The dependant variables are economic, demographic, community and law enforcement variables. After removing text variables as well as variables with missing data, we have 99 covariates. To test our screening method we use the data set with these 99 covariates and their second-order interactions terms. Removing those that are constant, p rises to 4077. The variables are standardized.

The SIS, the SIS-Kendall and the SIS-BW are used for comparative purposes. For these methods the size of the final subset with the training data is set to $d = n/log(n) = 144$.

Our method selects a total of 77 covariates. Among these covariates, 29 are also found among the 144 in the SIS final subset, 32 are in the final subset of the SIS-Kendall method and 24 are also found in the SIS-BW final subset.

The true model covariates are unknown for this dataset, hence an evaluation of the selected covariates provides very limited insight on the quality of the selection. To provide a more effective comparison of the methods, we compare the predictive power of the subset of covariates selected by each method. We obtain a robust fit of the linear regression model using M-estimation with the four respective selected subsets using the training set.

Each fitted model then does prediction on the test set. Table 3.3 shows the mean squared error (MSE) and the mean absolute error (MAE) for the predictions.

Table 3.3: The predictive error estimated by the MSE and the MAE for the SERVIF and its three competitors.

Predictive Error Estimation			
Method	Number of covariates	MSE	MAE
SERVIF	77	0.357	0.251
SIS	144	0.374	0.242
SIS-Kendall	144	1.674	0.531
SIS-bw	144	2.071	0.602

While, as noted earlier, we can't have a verdict on which method selected the best set of covariates, the predictive accuracy presented in Table 3 provides some insight into the quality of this selection. Despite having a much smaller subset, with 77 covariates versus 144 for the other methods, SERVIF shows very competitive predictive power. It has the lowest MSE and the second lowest MAE, surpassed only by its non-robust competitor, the original SIS. The robust competitors, SIS-Kendall and SIS-BW, are relatively far behind, both yielding fairly high MSE and MAE values. We can conclude that in spite of keeping a much smaller set of covariates, our method does not seem to eliminate any important covariates as it keeps good predictive accuracy.

3.5 Concluding remarks

Variable screening methods for large high dimensional data that are robust to data contamination and computationally efficient are becoming increasingly important. The existing methods require the sensitive choice of a final subset size. We present a method that automatically chooses the final subset size by choosing the most important covariates with regards to the response variable. In the simulation study SERVIF shows very good performance with regards to the inclusion of all true model covariates in the absence or presence of data contamination. Moreover, the automatic built-in subset size determination method selects much fewer covariates then the ad-hoc $n/log(n)$ rule used by the other methods presented in this study.

Possible avenues for future research include testing the method's sensitivity to the initial wealth parameters as well as its sensitivity to the size of the partitions. Moreover, extensions of the method to other models such as the generalized linear models, additive models or Cox proportional hazards models would also be of great interest.

References

[1] Chen, J. and Chen, Z. (2008). Extended Bayesian information criteria for model selection with large model spaces. *Biometrika*, 95(3):759–771.

[2] Donoho, D. L. (2000). High-dimensional data analysis: The curses and blessings of dimensionality. *AMS Math Challenges Lecture*, 1:32.

[3] Dupuis, D. J. and Victoria-Feser, M.-P. (2011). Fast robust model selection in large datasets. *Journal of the American Statistical Association*, 106(493):203–212.

[4] Dupuis, D. J. and Victoria-Feser, M.-P. (2013). Robust VIF regression with application to variable selection in large data sets. *The Annals of Applied Statistics*, 7(1):319–341.

[5] Fan, J. and Li, R. (2006). Statistical challenges with high dimensionality: Feature selection in knowledge discovery. *arXiv preprint math/0602133*.

[6] Fan, J. and Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(5):849–911.

[7] Fan, J., Samworth, R., and Wu, Y. (2009). Ultrahigh dimensional feature selection: beyond the linear model. *Journal of Machine Learning Research*, 10(Sep):2013–2038.

[8] Fan, J. and Song, R. (2010). Sure independence screening in generalized linear models with np-dimensionality. *The Annals of Statistics*, 38(6):3567–3604.

[9] Foster, D. P. and Stine, R. A. (2008). α-investing: a procedure for sequential control of expected false discoveries. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(2):429–444.

[10] Gather, U. and Guddat, C. (2008). Discussion on “Sure independence screening for ultrahigh dimensional feature space" by J. Fan and J. Lv. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(5):893–895.

[11] Gnanadesikan, R. and Kettenring, J. R. (1972). Robust estimates, residuals, and outlier detection with multiresponse data. *Biometrics*, 28:81–124.

[12] Guddat, C., Gather, U., and Kuhnt, S. (2010). *MCD-RoSIS – A robust procedure for variable selection*, volume Volume 7 of *Collections*, pages 75–83. Institute of Mathematical Statistics, Beachwood, Ohio, USA.

[13] Hall, P. and Miller, H. (2009). Using generalized correlation to effect variable selection in very high dimensional problems. *Journal of Computational and Graphical Statistics*, 18(3):533–550.

[14] Hubert, M., Rousseeuw, P. J., and Van Aelst, S. (2008). High-breakdown robust multivariate methods. *Statistical Science*, 23(1):92–119.

[15] Ke, T., Jin, J., and Fan, J. (2014). Covariance assisted screening and estimation. *Annals of Statistics*, 42(6):2202.

[16] Khan, J. A., Van Aelst, S., and Zamar, R. H. (2007). Robust linear model selection based on least angle regression. *Journal of the American Statistical Association*, 102(480):1289–1299.

[17] Li, G., Peng, H., Zhang, J., and Zhu, L. (2012). Robust rank correlation based screening. *The Annals of Statistics*, 40(3):1846–1877.

[18] Lichman, M. (2013). UCI Machine Learning Repository.

[19] Lin, D., Foster, D. P., and Ungar, L. H. (2011). VIF regression: a fast regression algorithm for large data. *Journal of the American Statistical Association*, 106(493):232–247.

[20] Ma, X. and Zhang, J. (2016). Robust model-free feature screening via quantile correlation. *Journal of Multivariate Analysis*, 143:472–480.

[21] Mai, Q. and Zou, H. (2015). The fused Kolmogorov filter: A nonparametric model-free screening method. *The Annals of Statistics*, 43(4):1471–1497.

[22] Mu, W. and Xiong, S. (2014). Some notes on robust sure independence screening. *Journal of Applied Statistics*, 41(10):2092–2102.

[23] Pan, R., Wang, H., and Li, R. (2016). Ultrahigh-dimensional multiclass linear discriminant analysis by pairwise sure independence screening. *Journal of the American Statistical Association*, 111(513):169–179.

[24] Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American Statistical Association*, 79(388):871–880.

[25] Saldana, D. F. and Feng, Y. (2018). SIS: An R package for sure independence screening in ultrahigh dimensional statistical models. *Journal of Statistical Software*, 83(2):1–25.

[26] Sun, J. (2016). *Robust Feature Screening Procedures for Mixed Type of Data*. PhD thesis, Virginia Tech.

[27] Wang, H. (2009). Forward regression for ultra-high dimensional variable screening. *Journal of the American Statistical Association*, 104(488):1512–1524.

[28] Zhu, L.-P., Li, L., Li, R., and Zhu, L.-X. (2011). Model-free feature screening for ultrahigh-dimensional data. *Journal of the American Statistical Association*, 106(496):1464–1475.

General Conclusion

This thesis proposed innovative methods of building prediction intervals and a robust variable screening method for high-dimensional data.

The first chapter explored two aspects: The splitting rule and the method used to build the prediction interval. Fifteen method combinations were proposed, thoroughly investigated, and compared to four alternative methods through simulation studies and with real data. The results showed that the proposed methods are very competitive. They outperform commonly used methods in both in simulation settings and with real data.

The second chapter extended the investigation of prediction intervals to finite mixture regression models. We proposed a new nonparametric regression method for finite mixture models that capture nonlinear dependencies links between the response and the covariates. The performance of the methods were assessed in an extensive simulation study. The results showed equal performance to existing finite mixture regression models methods when the true underlying functions are linear and greatly improved performance in the presence of nonlinearity and heteroscedasticity. The method calibration method also showed to be very reliable in attaining the prescribed coverage. Moreover, four out of the five methods also showed very good computational efficiency.

The third chapter proposed a robust variable screening method for large data sets. The method uses an iterative ensemble scheme for safe and gradual elimination of spurious covariates. An innovative feature of the method is its built-in automatic final subset size determination. The performance of the method was assessed in a simulation study with both clean and contaminated data and with a real data set. The results showed very good and consistent performance on both clean and contaminated data.

Many interesting avenues for future research ar possible. Extending the random forest based prediction intervals to censored survival data would be interesting and useful. In the case of mixture distributions, exploring ways to select the number of components automatically would be interesting. Improving the bootstrap adjustment/calibration method of Ahonen et al. (2018), for smaller sizes, would be useful. Generalizing the method to more challenging settings like the case of covariates dependent mixing proportions would be interesting. Finally, extending the robust variable screening to other models in the generalized linear models family would also be worthy of investigation.

Appendix A

Table A: Complete results of the simulation study aggregated by DGP and sample size. The column 'cov' shows the mean coverage of the PIs, 'ml' their mean length and 'sd', the standard deviation of the PI lengths. The 15 proposed method combinations are presented in a matrix format where the line (LS, L1, SPI) represent the splitting rule and the column (HDR, CHDR, Quant, SPI, LM) represent the method used to build the interval. The 4 competitors (QRF, CI-jack, CI-split, IF-jack) are presented right under the proposed methods.

ntrain	DGP																
*			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
150	1	LS	0.946	8.059	1.04	0.947	10.42	1.52	0.940	17.60	2.12	0.938	13.63	1.908	0.947	10.80	0.84
		L1	0.949	7.959	0.93	0.946	9.522	1.26	0.942	18.68	1.90	0.942	13.44	1.124	0.947	11.91	0.77
		SPI	0.947	7.747	0.89	0.946	9.515	1.19	0.941	17.82	1.82	0.941	13.26	1.248	0.949	11.55	0.85
			QRF			CI-jack			CI-split			IF-jack					
			0.975	30.30	0.72	0.945	15.85	1.12	0.960	21.99	2.85	0.953	16.10	1.171			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
150	2	LS	0.941	8.367	1.54	0.948	10.88	1.69	0.941	17.57	1.88	0.939	13.00	1.692	0.948	11.14	1.01
		L1	0.945	8.200	1.38	0.948	9.902	1.44	0.941	18.51	1.93	0.943	13.05	1.063	0.947	12.05	0.83
		SPI	0.944	7.974	1.29	0.947	9.776	1.25	0.940	17.72	1.89	0.942	12.76	1.200	0.948	11.69	0.92
			QRF			CI-jack			CI-split			IF-jack					
			0.974	30.24	0.71	0.946	15.87	1.12	0.959	21.96	3.22	0.954	16.15	1.125			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
150	3	LS	0.943	11.29	0.74	0.944	11.25	0.73	0.944	12.15	0.81	0.943	11.94	0.872	0.943	11.23	0.67
		L1	0.945	12.09	0.81	0.945	12.06	0.82	0.944	12.90	0.80	0.943	12.64	0.914	0.944	12.11	0.72
		SPI	0.944	11.97	0.78	0.945	11.94	0.77	0.942	12.83	0.80	0.943	12.65	0.877	0.945	12.05	0.71

ntrain	DGP																
			QRF			CI-jack			CI-split			IF-jack					
			0.973	15.59	0.64	0.948	10.87	0.66	0.962	13.39	1.47	0.948	10.85	0.650			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
150	4	LS	0.943	679.8	62.21	0.943	681.7	61.97	0.942	788.4	80.54	0.941	737.0	69.479	0.944	679.6	57.90
		L1	0.944	684.9	53.40	0.944	684.5	53.08	0.942	781.7	67.47	0.941	731.1	67.932	0.946	695.7	47.45
		SPI	0.943	679.5	54.64	0.943	679.2	54.29	0.942	781.6	76.51	0.941	723.8	67.647	0.947	692.1	50.98
			QRF			CI-jack			CI-split			IF-jack					
			0.965	1120	44.86	0.944	800.7	64.72	0.960	1046	171.4	0.942	784.8	62.345			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
150	5	LS	0.946	0.719	0.08	0.941	0.667	0.07	0.941	0.711	0.07	0.943	0.692	0.065	0.940	0.654	0.06
		L1	0.942	0.734	0.08	0.939	0.679	0.06	0.941	0.745	0.07	0.940	0.701	0.061	0.941	0.674	0.05
		SPI	0.944	0.731	0.08	0.942	0.672	0.06	0.941	0.722	0.07	0.943	0.692	0.055	0.942	0.661	0.05
			QRF			CI-jack			CI-split			IF-jack					
			0.955	0.900	0.06	0.944	0.776	0.09	0.964	1.146	0.27	0.939	0.724	0.089			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
150	6	LS	0.944	15.11	1.45	0.949	12.69	1.08	0.939	17.66	0.60	0.952	11.80	0.894	0.948	12.97	0.57
		L1	0.942	17.09	1.55	0.948	14.02	1.13	0.940	19.73	0.61	0.951	13.50	1.032	0.949	15.03	0.61
		SPI	0.942	16.08	1.42	0.948	13.44	1.21	0.940	18.80	0.57	0.952	13.02	0.978	0.948	14.03	0.61
			QRF			CI-jack			CI-split			IF-jack					
			0.968	21.19	0.30	0.943	14.02	0.64	0.962	17.03	1.35	0.952	14.31	0.665			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
500	1	LS	0.945	5.693	0.41	0.945	7.506	0.58	0.945	10.86	0.78	0.945	8.110	0.633	0.955	6.889	0.34
		L1	0.950	5.599	0.36	0.949	7.054	0.50	0.944	10.75	0.77	0.945	7.845	0.514	0.953	7.264	0.33
		SPI	0.949	5.550	0.38	0.948	7.193	0.54	0.944	10.47	0.74	0.947	7.900	0.543	0.954	7.152	0.34
			QRF			CI-jack			CI-split			IF-jack					
			0.984	25.44	0.67	0.950	9.865	0.44	0.955	13.57	1.32	0.960	10.08	0.472			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		

ntrain	DGP																
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
500	2	LS	0.947	5.902	0.80	0.945	7.441	0.61	0.946	10.59	0.76	0.944	7.346	0.653	0.953	7.050	0.46
		L1	0.948	5.745	0.65	0.951	7.218	0.60	0.946	10.89	0.72	0.947	7.261	0.571	0.952	7.433	0.41
		SPI	0.948	5.617	0.65	0.949	7.165	0.62	0.946	10.48	0.77	0.949	7.346	0.541	0.953	7.302	0.46
			QRF			CI-jack			CI-split			IF-jack					
			0.984	25.33	0.62	0.950	10.05	0.48	0.953	13.40	1.28	0.959	10.26	0.537			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
500	3	LS	0.944	8.971	0.29	0.944	8.955	0.29	0.942	9.371	0.37	0.943	9.339	0.382	0.944	8.989	0.26
		L1	0.944	9.683	0.32	0.944	9.677	0.31	0.943	10.22	0.36	0.942	10.043	0.398	0.944	9.786	0.28
		SPI	0.946	9.573	0.32	0.946	9.571	0.32	0.943	10.14	0.35	0.941	9.904	0.385	0.946	9.702	0.31
			QRF			CI-jack			CI-split			IF-jack					
			0.986	13.84	0.28	0.947	8.769	0.32	0.950	10.07	0.66	0.945	8.732	0.323			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
500	4	LS	0.943	588.1	30.0	0.943	588.9	29.9	0.944	627.9	32.3	0.944	620.1	33.6	0.942	581.5	29.2
		L1	0.943	584.6	29.3	0.943	584.5	29.3	0.943	620.9	27.6	0.943	610.9	28.4	0.941	579.8	24.0
		SPI	0.943	581.1	27.5	0.943	581.1	27.3	0.943	616.3	28.5	0.944	607.8	28.5	0.942	575.2	24.4
			QRF			CI-jack			CI-split			IF-jack					
			0.974	992.2	25.0	0.948	650.4	26.3	0.951	752.3	64.3	0.947	642.8	23.6			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
500	5	LS	0.948	0.582	0.03	0.946	0.577	0.03	0.947	0.598	0.03	0.947	0.592	0.027	0.945	0.565	0.03
		L1	0.946	0.598	0.03	0.944	0.592	0.03	0.944	0.620	0.03	0.945	0.608	0.029	0.944	0.577	0.03
		SPI	0.946	0.585	0.03	0.945	0.580	0.03	0.945	0.606	0.03	0.945	0.595	0.026	0.944	0.565	0.02
			QRF			CI-jack			CI-split			IF-jack					
			0.965	0.789	0.03	0.949	0.625	0.03	0.951	0.748	0.10	0.947	0.597	0.033			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
500	6	LS	0.955	10.65	0.95	0.950	9.127	0.48	0.948	14.56	0.30	0.951	8.471	0.382	0.952	9.927	0.26
		L1	0.954	12.83	1.08	0.951	10.675	0.50	0.944	16.75	0.32	0.950	9.715	0.449	0.951	11.70	0.31
		SPI	0.955	12.06	0.97	0.950	10.208	0.49	0.946	15.88	0.29	0.951	9.512	0.492	0.950	11.05	0.32

ntrain	DGP																
			QRF			CI-jack			CI-split			IF-jack					
			0.979	18.62	0.24	0.949	11.02	0.32	0.952	12.88	0.75	0.957	11.27	0.325			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
1000	1	LS	0.947	5.111	0.28	0.946	6.543	0.45	0.946	8.425	0.54	0.947	6.614	0.440	0.958	6.072	0.25
		L1	0.948	4.947	0.26	0.947	6.093	0.39	0.946	8.264	0.46	0.947	6.233	0.367	0.955	5.998	0.23
		SPI	0.948	4.964	0.26	0.948	6.256	0.41	0.946	8.058	0.48	0.947	6.334	0.398	0.957	6.122	0.27
			QRF			CI-jack			CI-split			IF-jack					
			0.985	21.24	0.78	0.950	7.549	0.28	0.951	9.922	0.71	0.959	7.681	0.341			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
1000	2	LS	0.948	4.851	0.39	0.944	6.195	0.45	0.947	8.191	0.53	0.946	5.716	0.431	0.955	6.007	0.39
		L1	0.947	4.653	0.37	0.948	5.937	0.46	0.947	8.281	0.48	0.946	5.470	0.388	0.954	5.984	0.29
		SPI	0.948	4.634	0.36	0.948	6.021	0.44	0.947	7.991	0.51	0.947	5.605	0.402	0.954	6.045	0.36
			QRF			CI-jack			CI-split			IF-jack					
			0.984	21.15	0.79	0.948	7.672	0.32	0.952	10.16	0.76	0.956	7.809	0.351			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
1000	3	LS	0.946	8.048	0.20	0.946	8.043	0.20	0.944	8.298	0.21	0.945	8.330	0.240	0.948	8.113	0.19
		L1	0.947	8.750	0.22	0.948	8.750	0.22	0.944	9.087	0.19	0.944	8.952	0.238	0.948	8.839	0.23
		SPI	0.949	8.615	0.22	0.949	8.614	0.22	0.946	9.026	0.19	0.944	8.839	0.219	0.947	8.665	0.22
			QRF			CI-jack			CI-split			IF-jack					
			0.990	12.85	0.18	0.949	7.944	0.19	0.949	8.867	0.48	0.946	7.879	0.223			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
1000	4	LS	0.946	559.7	18.7	0.946	559.9	19.0	0.947	587.5	23.4	0.947	578.7	21.5	0.944	553.5	16.9
		L1	0.945	556.5	17.5	0.946	556.6	17.8	0.946	577.0	20.3	0.948	577.3	18.8	0.944	551.2	16.2
		SPI	0.945	553.9	16.8	0.945	553.8	16.7	0.945	573.4	17.1	0.946	568.8	19.5	0.944	546.0	16.1
			QRF			CI-jack			CI-split			IF-jack					
			0.976	906.5	20.4	0.951	597.9	15.9	0.953	664.3	39.4	0.948	586.7	17.6			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		

ntrain	DGP																
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
1000	5	LS	0.947	0.535	0.02	0.946	0.537	0.02	0.946	0.551	0.02	0.947	0.548	0.02	0.945	0.528	0.02
		L1	0.946	0.547	0.02	0.944	0.549	0.02	0.945	0.572	0.02	0.946	0.565	0.02	0.944	0.538	0.02
		SPI	0.945	0.537	0.02	0.945	0.540	0.02	0.945	0.559	0.02	0.946	0.554	0.02	0.944	0.529	0.02
			QRF			CI-jack			CI-split			IF-jack					
			0.97	0.73	0.02	0.9487	0.55	0.02	0.95	0.6359	0.06	0.95	0.5287	0.02			

ntrain	DGP																
			HDR			CHDR			Quant			SPI			LM		
			cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd	cov	ml	sd
1000	6	LS	0.954	8.760	0.59	0.950	7.940	0.32	0.950	12.89	0.34	0.951	7.357	0.24	0.953	8.596	0.18
		L1	0.954	10.60	0.78	0.949	9.404	0.38	0.949	15.42	0.24	0.951	8.657	0.30	0.951	10.44	0.20
		SPI	0.954	10.05	0.71	0.949	9.013	0.38	0.950	14.62	0.26	0.950	8.412	0.31	0.951	9.875	0.20
			QRF			CI-jack			CI-split			IF-jack					
			0.984	17.38	0.214	0.950	9.544	0.227	0.951	11.07	0.425	0.959	9.859	0.247			