UCLA
UCLA Electronic Theses and Dissertations

Title
High-dimensional Inferences via Estimator Augmentation: Post-selection and Group Structure

Permalink
https://escholarship.org/uc/item/75v0b921

Author
Min, Seunghyun

Publication Date
2019

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA

Los Angeles

High-dimensional Inferences via Estimator Augmentation: Post-selection and Group Structure

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Statistics

by

Seunghyun Min

2019

© Copyright by
Seunghyun Min
2019

ABSTRACT OF THE DISSERTATION

High-dimensional Inferences
via Estimator Augmentation:
Post-selection and Group Structure

by

Seunghyun Min
Doctor of Philosophy in Statistics
University of California, Los Angeles, 2019
Professor Qing Zhou, Chair

Making statistical inference on high-dimensional data has been an interesting topic in recent days. To support this theme, this dissertation consists of three main components; (1) a new post-selection inference method, (2) group inference methods, and (3) a new R package.

First, a new method to construct confidence sets after lasso variable selection is developed, with strong numerical support for its accuracy and effectiveness. A key component of my method is to sample from the conditional distribution of the response y given the lasso active set, which, in general, is very challenging due to the tiny probability of the conditioning event. This technical difficulty is overcome by using estimator augmentation to simulate from this conditional distribution via Markov chain Monte Carlo given any estimate $\tilde{\mu}$ of the mean μ_0 of y. A randomization step for the estimate $\tilde{\mu}$ is then incorporated in my sampling procedure, which may be interpreted as simulating from a posterior predictive distribution by averaging over the uncertainty in μ_0. My Monte Carlo samples offer great flexibility in the construction of confidence sets for multiple parameters. Extensive numerical results show that my method is able to construct confidence sets with the desired coverage rate and, moreover, that the diameter and volume of my confidence sets are substantially smaller in comparison with a state-of-the-art method.

Second, the advantages of grouping variables are advocated by presenting extensive nu-

merical results. The parametric bootstrap with refitted thresholded group lasso estimator is compared with competitor methods. Then applications of estimator augmentation in group lasso are introduced which includes importance sampler and de-biased parametric bootstrap. The importance sampler introduced is shown to outperforms sampling directly from the target distribution by several orders.

Lastly, an R package `EAinference` which stems from estimator augmentation methods is introduced. The package contains a parametric bootstrap, an importance sampler, Metropolis Hastings sampler and many related simulation-based inference tools. The package is available on CRAN.

The dissertation of Seunghyun Min is approved.

Wotao Yin

Jingyi J. Li

Arash A. Amini

Qing Zhou, Committee Chair

University of California, Los Angeles

2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

VITA

2011 B.A., Major in Applied Statistics, Yonsei University, Seoul, Korea.

2014 M.A., Major in Applied Statistics, Yonsei University, Seoul, Korea.

2019 Expected Ph.D., Major in Statistics, UCLA, Los Angeles, California.

PUBLICATIONS

Park, T. and Min, S. (2016). *Partially Collapsed Gibbs Sampling for Linear Mixed-effects Models.* Communications in Statistics-Simulation and Computation, 45(1), 165-180.

Min, S. and Park, T. (2017). *Bayesian Variable Selection in Poisson Change-Point Regression Analysis.* Communications in Statistics-Simulation and Computation, 46(3), 2267-2282.

Zhou, Q. and Min, S. (2017). *Uncertainty Quantification Under Group Sparsity.* Biometrika, 104(3), 613-632.

Zhou, Q. and Min, S. (2017). *Estimator Augmentation with Applications in High-Dimensional Group Inference.* Electronic Journal of Statistics, 11(2), 3039-3080.

CHAPTER 1

Introduction

Technological advances in data processing, collection, and storage have led to torrential streams of data worldwide on unprecedented scales. Accordingly, high-dimensional data has become very common for exploring interesting and complex phenomena not possible before. Aligning with this trend, the importance of high-dimensional statistics has risen due to the complexity of data nowadays. Suppose the model of interest is:

$$y = \mu_0 + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_n),$$

where $y \in \mathbb{R}^n$ is the response variable. Often, μ_0 is assumed to be a form of a linear combination of explanatory variables X_i for $i = 1, \ldots, p$, i.e. $\mu_0 = X\beta_0$, where $X = [X_1 \mid \cdots \mid X_p] \in \mathbb{R}^{n \times p}$ is the design matrix and and $\beta_0 \in \mathbb{R}^p$ is the coefficient vector. Naturally, estimating β_0 has always been one of the most classical topics in statistics. However, under the high-dimensional setting when the number of samples are smaller than the number of variables, conventional methods like a least square estimation cannot be used due to the unidentifiable issue. To overcome this issue, penalization models has been extensively studied; see Tibshirani (1996), Fan and Li (2001), Efron et al. (2004), Zou and Hastie (2005) and Zhang (2010). These methods combine different penalty functions with the loss function of the least squares to define a sparse estimator. Hastie et al. (2015) provides a thorough summary of these penalized methods.

Recently, people's attention has moved to inference on sparse estimation. Beyond providing an estimator, studying the distribution of an estimator or quantifying uncertainty of an estimator becomes an important challenge.

In order to test the significance of each component of β_0, Wasserman and Roeder (2009) and Meinshausen et al. (2009) proposed a data splitting approach which uses half the data to select covariates and uses the other half to compute the p-value so that the variable selection procedure is independent from the inference procedure.

Another group focused on adjusting the bias of lasso and deriving the asymptotic distribution of the *debiased* lasso estimator. Zhang and Zhang (2014), Javanmard and Montanari (2014), and van de Geer et al. (2014) showed that the de-biased lasso estimator is asymptotically unbiased to β_0 and its asymptotic distribution is a normal distribution. This idea is further extended to the group lasso and the scaled group lasso by Mitra and Zhang (2016). With a similar logic, van de Geer and Stucky (2016) proposed a method to construct a confidence set for a subset of β_0 using the de-biased square-root lasso.

Bootstrap also has been a popular method to infer β_0. Chatterjee and Lahiri (2011) proposed using a modified bootstrap with thresholded lasso estimator. Zhou and Min (2017b) used a similar idea with group lasso to measure the significance of each group of variables. Zhang and Cheng (2017) and Dezeure et al. (2017) integrated bootstrap step to the de-biased lasso for a simultaneous inference.

Another group focuses on testing the significance of the additional variable from the sequential regression. Lockhart et al. (2014) developed a test statistics for a lasso solution path so that one can test whether to include more variables or not. This is further generalized to lars and forward stepwise regression by Tibshirani et al. (2016).

Another big stream of high-dimensional inference is post-selection inference. It focuses on the inference of the least square estimation when the explanatory variables are selected via penalized methods (Berk et al., 2013; Lee et al., 2016; Liu et al., 2018; Taylor and Tibshirani, 2018). Thus, the target interest becomes

$$X_A^+ \mu_0 = \underset{\beta}{\operatorname{argmin}}\, \mathbb{E}\|y - X_A\beta\|^2,$$

where A is selected variable indices, $X_A = (X_i)_{i\in A} \in \mathbb{R}^{n\times|A|}$ is a subset of X which consists of selected covariates and $X_A^+ \in \mathbb{R}^{|A|\times n}$ is Moore-Penrose pseudo inverse matrix. Unlike

conventional inference methods, inference is studied conditioned on the selection event. This is because the same dataset is used for the selection procedure and also for inference. More detailed review in post-selection inference will be done in Chapter 2.

Zhou (2014) derives the density of the *augmented* estimator, the lasso estimator and its subgradient, given a point estimate of β_0. Zhou and Min (2017a) generalized this idea to block lasso. Knowing the density of augmented estimator gives great flexibility in developing applications such as importance sampler and Markov chain Monte Carlo sampler.

1.1 Notation

Notation used throughout the dissertation is defined here. Let $\mathbb{N}_k$ denote the set $\{1, \cdots, k\}$ and $\mathbf{1}_{[k]}$ be a k-vector of ones. Denoted by Z_i the i-th column or the i-th component of Z when Z is a matrix or a vector, respectively. Correspondingly, I define $Z_A := (Z_i)_{i \in A}$ and $Z_{-i} := (Z_j)_{j \neq i}$. For a matrix Z, let Z_{AB} be the submatrix consisting rows in A and columns in B. Denote by $\text{row}(Z)$ and $\text{null}(Z)$ the row space and the null space of a matrix Z, respectively. The superscript $+$ is used for Moore-Penrose inverse.

Let n and p be a number of samples and a number of covariates, respectively. Denote lasso-type estimators and corresponding subgradients by $\hat{\beta}$ and S, respectively. Denote a group structure by $\mathcal{G} = \{\mathcal{G}_j\}_{j=1}^J$ with J as the number of groups where each $\mathcal{G}_j$ is a set of non-overlapping indices such that $\cup_{j=1}^J \mathcal{G}_j = \mathbb{N}_p$. Let p_j be the number of indices in j-th group, i.e. $p_j = |\mathcal{G}_j|$. Given the group structure $\mathcal{G}$, let $\mathcal{G}_B = \cup_{j \in B} \mathcal{G}_j \subset \mathbb{N}_p$ for $B \subset \mathbb{N}_J$. For a vector $v = (v_j)_{1:p}$, define $v_{(B)} = v_{\mathcal{G}_B}$ and, in particular, $v_{(j)} = v_{\mathcal{G}_j}$.

1.2 Review : Estimator Augmentation

The lasso type estimators can be defined by concatenating the least square loss and a regularization term. I will be using four lasso estimators; the lasso, the group lasso, the scaled lasso, the scaled group lasso. Since the lasso and the scaled lasso are the special cases of the group lasso and the scaled group lasso, respectively, instead of going over all four estimators,

only the group lasso and the scaled group lasso are reviewed.

Given a group structure $\{\mathcal{G}_j\}_{j=1}^J$, the group lasso estimator (Yuan and Lin, 2006) is defined

$$\hat{\beta} \in \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ \frac{1}{2}\|y - X\beta\|_2^2 + n\lambda \sum_{j=1}^{J} w_j \|\beta_{(j)}\|_2 \right\}, \tag{1.1}$$

where $w_j > 0$ and are set to 1 by default.

The scaled group lasso (Sun and Zhang, 2012) is a variation of the group lasso which is scale invariant and which simultaneously estimates σ^2 along with $\hat{\beta}$. It is often used to estimate the variance of the error term ε in a high-dimensional setting. Again, given a group structure $\{\mathcal{G}_j\}_{j=1}^J$, the scaled group lasso estimator is defined

$$(\hat{\beta}, \hat{\sigma}) \in \underset{\beta \in \mathbb{R}^p,\ \sigma \in \mathbb{R}^+}{\operatorname{argmin}} \left\{ \frac{1}{2\sigma}\|y - X\beta\|_2^2 + \frac{\sigma}{2} + n\lambda \sum_{j=1}^{J} w_j \|\beta_{(j)}\|_2 \right\}. \tag{1.2}$$

By letting $p_j = 1$ for every j, the group lasso and the scaled group lasso reduce to the lasso (Tibshirani, 1996) and the scaled lasso, respectively.

The method of estimator augmentation links the distribution of ε and that of the augmented estimator $(\hat{\beta}, S)$. For the computational efficiency in Metropolis Hastings samplers which will be introduced in Chapter 2, I review both estimator augmentation in lasso and estimator augmentation in group lasso.

1.2.1 Estimator augmentation in lasso

Let $\Psi = X^\mathsf{T} X/n$ and $\hat{\beta}$ be the lasso estimator defined in (1.1) by letting $p_j = 1$ for every j. I start from the Karush-Kuhn-Tucker (KKT) condition for the lasso,

$$\frac{1}{n} X^\mathsf{T} y = \Psi \hat{\beta} + \lambda W S, \tag{1.3}$$

where $W = \text{diag}(w_i)_{i=1}^p$ and S is the subgradient of the ℓ_1 norm at $\hat{\beta}$:

$$\begin{cases} S_i = \text{sgn}(\hat{\beta}_i) & \text{if } \hat{\beta}_i \neq 0, \\ S_i \in [-1, 1] & \text{otherwise.} \end{cases} \tag{1.4}$$

Zhou (2014) inverted the KKT condition to find the sampling distribution of the so-called augmented estimator, $(\hat{\beta}, S)$, linking its density to that of $X^\mathsf{T} y$. Let $U = \frac{1}{n} X^\mathsf{T} \varepsilon$ and $\Theta = (\hat{\beta}_\mathcal{A}, S_\mathcal{I})$, where both $\mathcal{A} = \text{supp}(\hat{\beta})$ and $\mathcal{I} = \mathbb{N}_p \setminus \mathcal{A}$ are random as functions of $\hat{\beta}$. Note that $(\Theta, \mathcal{A})$ gives a parameterization of $(\hat{\beta}, S)$ due to the definition of the subgradient S. The KKT condition can be rewritten,

$$U = \Psi\hat{\beta} + \lambda W S - \frac{1}{n} X^\mathsf{T} \mu_0 := H(\Theta, \mathcal{A}; \mu_0, \lambda). \tag{1.5}$$

Unless otherwise noted, I assume that $n < p$ and X has full row rank, i.e. $\text{rank}(X) = n$. Under this setting, the vector $U \in \text{row}(X)$, an n-dimensional subspace of $\mathbb{R}^p$. Let V be a $p \times p$ orthogonal matrix such that (i) the first n columns of V, indexed by $R = \{1, \ldots, n\}$, consist of n orthonormal eigenvectors associated with the positive eigenvalues of Ψ, and (ii) the last $p-n$ columns, indexed by $N = \{n+1, \ldots, p\}$, are a collection of orthonormal vectors that forms a basis of $\text{null}(X)$. Then U can be re-expressed by its coordinates with respect to V_R as

$$R = V_R^\mathsf{T} U \sim \mathcal{N}_n(0, \sigma^2 \Lambda / n), \tag{1.6}$$

where $\Lambda = \text{diag}(\Lambda_i)_{i=1}^n$ and Λ_i's are the positive eigenvalues of Ψ. Let f_R be the density of R. Equation (1.5) enforces a set of constraints on the p-vector S, i.e. $V_N^\mathsf{T} W S = 0$, since WS must lie in $\text{row}(X)$. Denote the value of the random vector $(\hat{\beta}, S)$ by (b, s) and the corresponding value of $(\Theta, \mathcal{A})$ by $(\theta, A) = (b_A, s_I, A)$, where $A \subset \mathbb{N}_p$ and $I = \mathbb{N}_p \setminus A$. Then

θ must satisfy the constraints

$$V_{AN}^{\mathsf{T}} W_{AA} \operatorname{sgn}(b_A) + V_{IN}^{\mathsf{T}} W_{II} s_I = 0, \tag{1.7}$$

$$\|s_I\|_\infty \leq 1. \tag{1.8}$$

Let $q = |A| \leq n$ (Remark 1). Differentiating (1.7), one sees that $ds_I \in \operatorname{null}(V_{IN}^{\mathsf{T}} W_{II})$, which is an $(n-q)$-dimensional subspace of $\mathbb{R}^{|I|}$. Thus s_I can be parameterized by $s_F \in \mathbb{R}^{n-q}$ such that $ds_I = B(I) ds_F$, where F is a size-$(n-q)$ subset of I and $B(I) \in \mathbb{R}^{|I| \times (n-q)}$ is an orthonormal basis of $\operatorname{null}(V_{IN}^{\mathsf{T}} W_{II})$. Under mild conditions H defined in (1.5) is a bijection (Lemma 3 in Zhou (2014)), which is used to derive the distribution for $(\Theta, \mathcal{A})$ from the density f_R. To ease my notation, let $d\theta := db_A ds_F$ be a differential form of order n. Zhou (2014) showed that the distribution of $(\Theta, \mathcal{A})$ can be represented by such n-forms:

Theorem 1 (Theorem 2 in Zhou (2014))**.** *Assume $p > n$ and that every n columns of X are linearly independent. If $y \sim \mathcal{N}_n(\mu_0, \sigma^2 \mathbf{I}_n)$ and $\lambda > 0$, then the joint distribution of $(\Theta, \mathcal{A}) = (\hat{\beta}_{\mathcal{A}}, S_{\mathcal{I}}, \mathcal{A})$ is given by*

$$\mathbb{P}_{\Theta,\mathcal{A}}(d\theta, A) = f_R\big(V_R^{\mathsf{T}} H(\theta, A; \mu_0, \lambda); \sigma^2\big) |\det T(A; \lambda)| d\theta \tag{1.9}$$

for (θ, A) satisfying (1.7) and (1.8), where $f_R(\bullet; \sigma^2)$ is the density of the distribution in (1.6) and $T(A; \lambda) = [V_R^{\mathsf{T}} \Psi_A | \lambda V_{IR}^{\mathsf{T}} W_{II} B(I)] \in \mathbb{R}^{n \times n}$.

Remark 1. The right side of (1.9) defines a joint density of $(\Theta, \mathcal{A})$ with respect to the parameterization (b_A, s_F) for θ. This density will be used to develop an MCMC algorithm for my conditional sampling step. It is a quite mild assumption that every n columns of X are linearly independent: If the entries of X are drawn from a continuous distribution over $\mathbb{R}^{n \times p}$, then this assumption will hold almost surely. This assumption also guarantees that the lasso solution is unique and $|\mathcal{A}| \leq n$ for every $y \in \mathbb{R}^n$ (Tibshirani, 2013). Hereafter, when conditioning on $\mathcal{A} = A$ I always assume that $|A| \leq n$.

1.2.2 Estimator augmentation in the group lasso

Given the group structure, the KKT condition for the group lasso estimator (1.1) is

$$\frac{1}{n}X^{\mathsf{T}}y = \Psi\hat{\beta} + \lambda WS, \tag{1.10}$$

where $W = \mathrm{diag}\{w_i\mathbf{I}_{p_i}\}_{i=1}^{J}$ and S is its subgradient:

$$\begin{cases} S_{(j)} = \hat{\beta}_{(j)}/\|\hat{\beta}_{(i)}\|_2 & \text{if } \hat{\beta}_{(j)} \neq 0, \\ \|S_{(j)}\|_2 \leq 1 & \text{otherwise.} \end{cases} \tag{1.11}$$

Or equivalently (1.10) can be rewritten as follows:

$$\frac{1}{n}X_{(j)}^{\mathsf{T}}y = \frac{1}{n}X_{(j)}^{\mathsf{T}}X\hat{\beta} + \lambda w_j S_{(j)} \quad \text{for } j \in \mathbb{N}_J. \tag{1.12}$$

Let $\mathcal{M} = \mathcal{G}(\hat{\beta}) \subset \mathbb{N}_J$ represent the active group indices and define $\hat{\gamma}_j = \|\hat{\beta}_{(j)}\|_2$. Rewriting the KKT condition gives

$$\frac{1}{n}X^{\mathsf{T}}\varepsilon = \sum_{j\in\mathcal{M}} \hat{\gamma}_j \Psi_{(j)} S_{(j)} + \lambda WS - \frac{1}{n}X^{\mathsf{T}}\mu_0 := H'(\hat{\gamma}_{\mathcal{M}}, S, \mathcal{M}; \mu_0, \lambda). \tag{1.13}$$

Note that, $\hat{\beta}_{(j)} = \hat{\gamma}_j S_{(j)}$ and thus $(\hat{\gamma}, S)$ is a equivalant representation of $(\hat{\beta}, S)$. Moving $X^{\mathsf{T}}/\sqrt{n}$ to the r.h.s to further define $\tilde{H}(\hat{\gamma}_{\mathcal{M}}, S)$,

$$E = \varepsilon/\sqrt{n} = \sqrt{n}(X^{\mathsf{T}})^{+}H'(\hat{\gamma}_{\mathcal{M}}, S, \mathcal{M}; \mu_0, \lambda) := \tilde{H}(\hat{\gamma}_{\mathcal{M}}, S, \mathcal{M}; \mu_0, \lambda). \tag{1.14}$$

The goal is to derive the distribution of $(\hat{\gamma}_{\mathcal{M}}, S, \mathcal{M})$ from that of E. Let f_E be the distribution of E which is $\mathcal{N}_n(0, \sigma^2\mathbf{I}_n/n)$ under the normal error assumption.

(1.10) forces $S \in \mathrm{row}(XW^{-1})$. Together with (1.11), S lies in $(n - |M|)$ dimensional manifold in $\mathbb{R}^p$,

$$\mathcal{S}_M = \left\{ v \in \mathrm{row}(XW^{-1}) : \|v_{(j)}\|_2 = 1 \ \forall j \in M \text{ and } \|v_{(j)}\|_2 \leq 1 \ \forall j \notin M \right\}. \tag{1.15}$$

Thereby, $s \in \mathcal{S}_M$ needs to satisfy

$$\begin{aligned} Q^{\mathsf{T}} s &= 0, \\ ||s_{(j)}||_2 &= 1 \quad \forall j \in M, \end{aligned} \tag{1.16}$$

where Q is an orthonormal basis of null(XW^{-1}). Differentiate (1.16) with respect to s gives,

$$\begin{aligned} Q^{\mathsf{T}} ds &= 0, \\ \langle s_{(j)}, ds_{(j)} \rangle &= 0 \quad \forall j \in M. \end{aligned} \tag{1.17}$$

With these constraints, s can be parameterized by $s_{F'} \in \mathbb{R}^{n-|M|}$, where $F' \subset \mathbb{N}_p$ and $|F'| = n - |M|$. In other words, there exists a mapping $T(s, M)$ such that $ds = T(s, M) ds_{F'}$.

Let $\Theta = (\hat{\gamma}_{\mathcal{M}}, S)$ and $\theta = (r_M, s)$ be the value of Θ. Analogous to Zhou (2014), Zhou and Min (2017a) show that under mild conditions $\tilde{H}$ is a bijection (Lemma 3.1 in Zhou and Min (2017b)) that maps $(\Theta, \mathcal{M})$ onto $\mathbb{R}^n$. Using the $(dr_M, ds_{F'})$ representation, the complete density of $(\Theta, \mathcal{M})$ is given as follows:

Theorem 2. *(Theorem 3.3 in Zhou and Min (2017a)) The distribution of* $(\Theta, \mathcal{M})$ *is given by*

$$\mathbb{P}_{\Theta,\mathcal{M}}(d\theta, M) = f_E\big(\tilde{H}(r_M, s; \mu_0, \lambda)\big) \big| \det J_M(r_M, s; \lambda) \big| d\theta,$$

$$\textit{where} \begin{cases} J_M(r_M, s; \lambda) = \Big[\sqrt{n}(X^{\mathsf{T}})^+ \big[(\Psi \circ s)_M | (r \circ \Psi + \lambda W) T(s, M)\big]\Big], \\ \Psi \circ s = \Big[\Psi_{(1)} s_{(1)} \Big| \cdots \Big| \Psi_{(J)} s_{(J)}\Big], \\ r \circ \Psi = \Big[r_1 \Psi(1) \Big| \cdots \Big| r_J \Psi_{(J)}\Big], \\ |M| \leq n. \end{cases}$$

1.2.3 Estimator augmentation in the scaled group lasso

The KKT condition of the scaled group lasso which corresponds to $(\hat{\beta}, \hat{\sigma})$ from (1.2) is

$$\frac{1}{n} X^{\mathsf{T}} y = \Psi \hat{\beta} + \lambda \hat{\sigma} W S, \tag{1.18}$$

$$\hat{\sigma} = \|y - X\hat{\beta}\|_2 / \sqrt{n} \tag{1.19}$$

where $W = \mathrm{diag}\{w_i \mathbf{I}_{p_i}\}_{i=1}^{J}$ and S is the same as in (1.11). With little algebra, from (1.18) and (1.19), one can see that S needs to satisfy $\lambda\sqrt{n}||(X^{\mathsf{T}})^{+} W S||_2 = 1$. Taking derivative with respect to s gives

$$s^{\mathsf{T}} W X^{\mathsf{T}} (X X^{\mathsf{T}})^{-2} X W ds = 0.$$

Due to this additional constraint, s now stays in $\mathbb{R}^{n-|M|-1}$ manifold in $\mathbb{R}^p$ and thus it can be parameterize by $s_{\tilde{F}}$ such that $ds = \tilde{T}(s, M) ds_{\tilde{F}}$, where $\tilde{T}(s, M) \in \mathbb{R}^{p \times (n-|M|-1)}$.

Let $\Theta = (\hat{\gamma}_{\mathcal{M}}, S, \hat{\sigma}^2)$ and $\theta = (r_M, s, \check{\sigma}^2)$ be the value of Θ. Using the $(dr_M, ds_{\tilde{F}})$ representation, the complete density of $(\Theta, \mathcal{M})$ can be shown as Theorem 3.

Theorem 3. *(Theorem 5.2 in Zhou and Min (2017a)) The distribution of $(\Theta, \mathcal{M})$ is given by*

$$\mathbb{P}_{\Theta, \mathcal{M}}(d\theta, M) = f_E\big(\tilde{H}(r_M, s; \beta_0, \lambda\check{\sigma})\big) \Big| \det J_M(\theta; \lambda) \Big| d\theta,$$

$$\textit{where} \begin{cases} J_M(r_M, s; \lambda) = \Big[\sqrt{n}(X^{\mathsf{T}})^{+} \big[(\Psi \circ s)_M \big| (r \circ \Psi + \lambda\check{\sigma} W)\tilde{T}(s, M) \big| \lambda W s\big]\Big], \\ \Psi \circ s = \Big[\Psi_{(1)} s_{(1)} \big| \cdots \big| \Psi_{(J)} s_{(J)}\Big], \\ r \circ \Psi = \Big[r_1 \Psi(1) \big| \cdots \big| r_J \Psi_{(J)}\Big], \\ |M| \le n - 1. \end{cases}$$

1.3 Review: The de-biased lasso

The debased lasso, or the desparified lasso, was developed by three groups (van de Geer et al., 2014; Zhang and Zhang, 2014; Javanmard and Montanari, 2014) around the same time. The key idea is to adjust the bias of the lasso estimator. Here, I review van de Geer

et al. (2014).

Assume that $y = X\beta_0 + \varepsilon$ and $\varepsilon \sim \mathcal{N}_n(0, \sigma^2 \mathbf{I}_n)$. Let $\hat{\Psi}$ be a relaxed inverse matrix of Ψ. From the KKT conditions of the lasso estimator (1.3) can be rewritten as

$$\beta_0 = \hat{\beta} + \lambda \hat{\Psi} W S - \frac{1}{n} \hat{\Psi} X^{\mathsf{T}} \varepsilon - \frac{\Delta}{\sqrt{n}},$$

where

$$\Delta = \sqrt{n}(\hat{\Psi}\Psi - \mathbf{I}_n)(\hat{\beta} - \beta_0). \tag{1.20}$$

By Theorem 2.2 in van de Geer et al. (2014), Δ is proven to be negligible which justifies the de-biased lasso estimator $\hat{b}$ to be defined as

$$\hat{b} := \hat{\beta} + \lambda \hat{\Psi} W S. \tag{1.21}$$

As a result, the asymptotic distribution of $\sqrt{n}(\hat{b} - \beta_0)$ becomes

$$\sqrt{n}(\hat{b} - \beta_0) \sim \mathcal{N}_p(0, \sigma^2 \hat{\Psi} \Psi \hat{\Psi}),$$

which is used to generate the confidence interval of $\beta_{0,j}$ for each $j \in \mathbb{N}_p$.

1.4 Review: Lee's method

Lee's method (Lee et al., 2016) generates confidence intervals of post-selection estimators when the selection event is given as $\mathcal{A} = A$ via lasso. Denoted by ν_j the j-th component of $X_A^+ \mu_0$. The goal is to find a confidence interval $\hat{I}_j$ of ν_j

$$\mathbb{P}(\nu_j \in \hat{I}_j \mid \mathcal{A} = A) \geq 1 - \alpha.$$

Assume $y \sim \mathcal{N}_n(\mu_0, \sigma^2 \mathbf{I}_n)$. Given active set A, decomposing the KKT condition of lasso

estimator (1.3) into an active part and an inactive part gives

$$\begin{aligned} X_A^\top (X_A \hat{\beta}_A - y) + n\lambda S_A &= 0, \\ X_I^\top (X_A \hat{\beta}_A - y) + n\lambda S_I &= 0, \end{aligned}$$

where $\|S_I\|_\infty < 1$. Looking $(\hat{\beta}_A, S_I)$ as random variables while fixing $\mathcal{A} = A$ and $S_A = s_A$, one can see the event $\{\mathcal{A} = A, \operatorname{sgn}(\hat{\beta}_A) = s_A\}$ is equivalent to imposing polyhedral constraints on y. Denoted a projection matrix onto X_A by P_A.

$$\{\mathcal{A} = A, S_A = s_A\} = \{Q_1(A, s_A) y \le Q_2(A, s_A)\}, \tag{1.22}$$

$$\begin{aligned} \text{where } Q_1(A, s_A) &= \begin{bmatrix} X_{-A}^\top (\mathbf{I}_n - P_A)/n\lambda \\ -X_{-A}^\top (\mathbf{I}_n - P_A)/n\lambda \\ -\operatorname{diag}(s_A)(X_A^\top X_A)^{-1} X_A^\top \end{bmatrix}, \\ \text{and } Q_2(A, s_A) &= \begin{bmatrix} \mathbf{1}_{p-|A|} - X_{-A}^\top (X_A^\top)^+ s_A \\ \mathbf{1}_{p-|A|} + X_{-A}^\top (X_A^\top)^+ s_A \\ -n\lambda \operatorname{diag}(s_A)(X_A^\top X_A)^{-1} s_A \end{bmatrix}. \end{aligned}$$

For $\eta \in \mathbb{R}^n$, let $c := \eta(\eta^\mathsf{T}\eta)^{-1}$ and $z := (\mathbf{I}_n - c\eta^\mathsf{T})y$. The polyhedral constraints on y (1.22) can be transform to derive the feasible interval of $\eta^\mathsf{T} y$.

$$\begin{aligned} \{Q_1(A, s_A) y < Q_2(A, s_A)\} &= \left\{ V_{s_A}^-(z) \le \eta^\top y \le V_{s_A}^+(z) \right\}, \\ \text{where } &\begin{cases} V_{s_A}^-(z) = \max_{j:(Q_1 c)_j < 0} \frac{Q_{2,j} - (Q_1 z)_j}{(Q_1 c)_j}, \\ V_{s_A}^-(z) = \min_{j:(Q_1 c)_j > 0} \frac{Q_{2,j} - (Q_1 z)_j}{(Q_1 c)_j}. \end{cases} \end{aligned}$$

Taking union over all the possible sign patterns of S_A gives

$$\{\mathcal{A} = A\} = \bigcup_{s_A \in \{-1,1\}^{|A|}} \{V_{s_A}^-(z) \le \eta^\top y \le V_{s_A}^+(z)\}. \tag{1.23}$$

With (1.23) and fix z with its observed value z^o, one can see

$$[\eta^\mathsf{T} y \mid \mathcal{A} = A, z = z^o] \sim \mathcal{TN}\left(\eta^\mathsf{T}\mu_0, \sigma^2\|\eta\|^2, \bigcup_{s_A}[V^-_{s_A}(z^o), V^+_{s_A}(z^o)]\right),$$

where $\mathcal{TN}(\mu, \sigma^2, \mathcal{V})$ is $\mathcal{N}(\mu, \sigma^2)$ truncated to the set $\mathcal{V}$. The condition $z = z^o$ can be removed from the independency between z and $\eta^\mathsf{T} y$. Fianlly, letting $\eta^\mathsf{T} = e_j^\mathsf{T} X_A^+$ for $j \in \mathbb{N}_{|A|}$ gives

$$[e_j X_A^+ y \mid \mathcal{A} = A] \sim \mathcal{TN}\left(e_j X_A^+ \mu_0, \sigma^2\|e_j X_A\|^2, \bigcup_{s_A}[V^-_{s_A}(z^o), V^+_{s_A}(z^o)]\right). \tag{1.24}$$

Confidence intervals of $e_j X_A^+ \mu_0$ for $j \in \mathbb{N}_{|A|}$ can be constructed with (1.24).

1.5 Outline

Chapter 2 introduces a new post-selection inference method which uses a simulation approach to draw samples from the conditional distribution of the response variable y. I introduce how to construct confidence sets and intervals of post-selection estimators and show the benefit of out methods over Lee's method. In Chapter 3, the focus moves to inference methods for data with a group structure and advocate the effectiveness of grouping by showing extensive numerical results. An R package which contains sampling and inference methods for lasso-type estimators is then introduced in Chapter 4. Lastly, the dissertation is concluded with summary and future directions in Chapter 5.

CHAPTER 2

Post-selection Inference

2.1 Introduction

Assuming that a random vector $y \in \mathbb{R}^n$ follows a multivariate Gaussian distribution,

$$y = \mu_0 + \varepsilon, \qquad \varepsilon \sim \mathcal{N}_n(0, \sigma^2 \mathbf{I}_n), \tag{2.1}$$

I wish to make inference on the unknown mean vector $\mu_0 \in \mathbb{R}^n$ after observing y. Given a set of p covariates $X = \left[X_1 | \cdots | X_p\right] \in \mathbb{R}^{n \times p}$, a common approach is to approximate μ_0 with a linear combination $X\beta$ for $\beta \in \mathbb{R}^p$. When the number of covariates is large, I often face the situation that only some of them can be included in the linear approximation. They may be selected manually or by a certain model selection method. Let $A \subset \{1, \cdots p\}$ be the set of selected covariates. After the selection step is done, my goal shifts to constructing a linear model that can best approximate μ_0 with only the selected covariates $X_A = (X_j, j \in A)$. Then the parameter of interest

$$\nu := X_A^+ \mu_0 = \underset{\beta \in \mathbb{R}^{|A|}}{\operatorname{argmin}} \|\mu_0 - X_A \beta\|_2^2 \tag{2.2}$$

is defined by the projection of μ_0 onto the space spanned by X_A. Inference on ν is called post-selection inference (Pötscher, 1991). In large-scale analysis, model selection is usually applied as an initial screening for important variables or features. In these applications, naive methods based on the standard t-statistic or interval will not provide valid inference for the selected variables due to selection bias in the screening step (Tibshirani et al., 2016; Liu et al., 2018). By conditioning on the model selection event, post-selection inference

provides reliable quantification of the significance of a selected variable, which is critical for follow-up investigations. Another appealing feature for post-selection inference is that it is valid without assuming a true linear model for y, only regarding the selected model as an approximation for μ_0 (Berk et al., 2013), which greatly relaxes its model assumptions.

When the selection step is done independently from y, for example by using another independent dataset or by pre-given information, inference on ν can be easily done with conventional methods. The distribution of the least-squares estimator $\hat{\nu} = X_A^+ y$ simply follows a Gaussian distribution. However, the problem becomes much more challenging when the selection step is data-driven and uses the same y. In such a case, conditioning on the selected active set A, the sampling distribution of y is restricted to a potentially irregular subset of $\mathbb{R}^n$. This problem is further complicated for high-dimensional data with $p > n$. Several lines of recent work have laid down the theoretical foundations and developed novel methods for post-selection inference on high-dimensional data. Tibshirani et al. (2016) develop a truncated Gaussian statistic to test the significance of an entering variable in each step of a sequential regression method, which generalizes the earlier work by Lockhart et al. (2014). Tibshirani et al. (2018) establish uniform convergence properties of this statistic, without normal assumption, as $n \to \infty$ and p stays fixed. Lee et al. (2016) build exact confidence intervals for individual components ν_j of ν in (2.2), where the set A is the support of the lasso (Tibshirani, 1996), which I reviewed in Section 1.4. The authors show that conditioning on the active set of the lasso is equivalent to imposing polyhedral constraints on y, a key idea used in Tibshirani et al. (2016) as well. Tian and Taylor (2017) have established asymptotic results for Lee's method without imposing Gaussian assumption. Taylor and Tibshirani (2018) further generalize Lee's method to generalized linear models, Cox's proportional hazards model and Gaussian graphical models. By conditioning on a smaller and more robust subset of the lasso active set, Liu et al. (2018) develop a more efficient method that produces shorter intervals.

In this chapter, I seek to make inference on ν (2.2) with the model selected by the lasso.

That is, the set A is the support of

$$\hat{\beta}(y) := \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \sum_{i=1}^{p} w_i |\beta_i|, \tag{2.3}$$

where $w_i > 0$ and are set to 1 by default. However, in contrast to Lee et al. (2016) and the other methods reviewed above, I aim at constructing not only confidence intervals for an individual ν_j, but also confidence sets for ν_B, where B contains an arbitrary subset of A. To the best of my knowledge, methods for constructing confidence sets after model selection have not been proposed in the literature. Although one might consider simultaneously covering all ν_j, $j \in B$ by controlling family-wise error rate, such an approach would be very stringent for large B, as verified numerically in comparison to my proposed method. On the other hand, the method of Lee et al. (2016) critically relies on the cumulative distribution function of a *univariate* Gaussian distribution truncated to the union of $2^{|A|}$ intervals. It seems highly intractable to generalize their technique for joint inference on a potentially large set of ν_j. Moreover, although Lee's method preserves the coverage rate at a desirable level, their confidence intervals are not always informative. In particular, their method sometimes produces infinite intervals with ∞ or $-\infty$ as the upper or lower bound, severely limiting its practical applications. Kivaranovic and Leeb (2018) show that the expected interval length of Lee's method can be infinity under certain condition, which is frequently satisfied in their simulation study.

A key difference between my method and the existing ones is that ours is built upon sampling of y^* that leads to the same active set of the lasso, i.e. $[y^* \mid \operatorname{supp}(\hat{\beta}(y^*)) = A]$, where $A = \operatorname{supp}(\hat{\beta}(y))$ is computed from the observed data (X, y). This sampling-based approach allows for the construction of confidence sets for joint inference on any subset of the parameter vector ν. It also offers great flexibility in choosing the statistic for inference, as the distribution of any function $T(y^*)$ can be readily approximated from a large sample of y^*. However, this conditional sampling is a challenging computational problem, since the event $\{\operatorname{supp}(\hat{\beta}(y^*)) = A\}$ is in general a rare event, especially when p is large. To complete this difficult task, I develop a novel conditional sampler via the method of estimator augmentation

which I reviewed in Chapter 1.2.1 (Zhou, 2014), given a point estimate $\tilde{\mu}$ of μ_0. To protect my method from a poor estimate $\tilde{\mu}$, I introduce a randomization step to draw a uniform sample of $\tilde{\mu}$ from a set $\widehat{C}$, which in conjunction with my conditional sampling of y^* produces an efficient and accurate tool for joint inference after lasso selection. The set $\widehat{C}$ can be seen as a way to incorporate the uncertainty in estimating μ_0 from y, prior to or unconditional on model selection, which allows for an adaptive and robust approximation of the distribution $[y^* \mid \text{supp}(\hat{\beta}(y^*)) = A]$. When used for inference on individual parameters, my method often builds much shorter confidence intervals than Lee's method, while achieving a comparable coverage rate. Furthermore, my method, by design, does not produce infinite intervals or sets. my post-selection inference method has been implemented in the R package **EAinference**, which will be introduced in Chapter 4.

The rest of this chapter is organized as follows. In Section 2.2, I introduce the key ingredients of my method: how to build confidence sets via conditional sampling and how to implement the randomization step. Section 2.3 develops a Markov chain Monte Carlo (MCMC) algorithm for the conditional sampling. Section 2.4 demonstrates empirically the effectiveness and accuracy of the confidence sets constructed by my method, including comparisons with Lee's method. I conclude the chapter with some remarks and discussion in Section 2.5. Proofs of technical results are provided in Section 2.6.

2.2 Post-selection inference

2.2.1 Basic idea

For the lasso estimate $\hat{\beta}(y)$, let $\mathcal{A}(y) = \text{supp}(\hat{\beta}(y))$ be the set of active variables. Given the active set $\mathcal{A}(y) = A$, the parameter of interest $\nu = X_A^+ \mu_0$ (2.2) is the coefficient vector for the projection of $\mu_0 = \mathbb{E}[y]$ onto $\text{span}(X_A)$. my goal is to construct a confidence set $\widehat{I}_B(\alpha)$ such that

$$\mathbb{P}\Big\{\nu_B \in \widehat{I}_B(\alpha)\Big|\mathcal{A}(y) = A\Big\} \geq 1 - \alpha \quad \text{for } B \subset \mathbb{N}_{|A|}, \tag{2.4}$$

where the probability is taken with respect to $y \sim \mathcal{N}_n(\mu_0, \sigma^2 \mathbf{I}_n)$. In particular, when $B = \{j\}$, $\widehat{I}_B(\alpha)$ is a confidence interval for ν_j, which I denote by $\widehat{I}_j(\alpha)$. A natural choice for the center of the confidence set is $\hat{\nu}_B := [X_A^+ y]_B$. This problem is, however, more complicated than it may look. Since I have selected variables using lasso, the distribution of $X_A^+ y$ given $\mathcal{A}(y) = A$ is no longer a Gaussian distribution, as the support of y is now only a proper subset of $\mathbb{R}^n$.

I will develop a simulation-based approach. Note that the conditioning event $\{\mathcal{A}(y) = A\}$ restricts my sampling to those y for which the lasso $\hat{\beta}(y)$ selects exactly the same variables in A, which is usually a rare event. Thus, it is almost impossible to use bootstrap to draw from $[y^* \mid \mathcal{A}(y^*) = A]$, where y^* denotes a sample drawn from an (estimated) distribution of y. However, estimator augmentation (Zhou, 2014) enables us to simulate from this conditional distribution, with a point estimate $\tilde{\mu}$ for μ_0, by an MCMC algorithm; see Section 2.3.2 for the details.

Suppose I have drawn a large sample of y^* by this Monte Carlo algorithm. One could use $[X_A^+(y^* - \tilde{\mu}) \mid \mathcal{A}(y^*) = A]$, which can be easily estimated from the samples of y^*, to approximate $[X_A^+(y - \mu_0) \mid \mathcal{A}(y) = A]$ and build a confidence set for ν_B. In practice, a poor choice of $\tilde{\mu}$ often results in poor coverage. To overcome this issue, I develop a robust method which randomizes the plug-in estimate $\tilde{\mu}$. As it will become clear, I do not attempt to estimate the conditional distribution $[X_A^+(y - \mu_0) \mid \mathcal{A}(y) = A]$, instead my approach is to bound the relevant quantiles of this distribution in order to perform conservative inference as stated in (2.4).

2.2.2 The randomization step

I will first develop my method for constructing confidence intervals for ν_j, which will be generalized in Section 2.2.3 to joint inference on ν_B. Let $q_{j,\gamma}(\mu)$ be the γ-quantile of the distribution

$$[\{X_A^+(y^* - \mu)\}_j \mid \mathcal{A}(y^*) = A], \tag{2.5}$$

where $y^* = \mu + \varepsilon^*$ and $\varepsilon^* \sim \mathcal{N}_n(0, \sigma^2 \mathbf{I}_n)$. For $\gamma \in (0,1)$, construct an interval

$$\xi_j(\mu, \gamma) := \Big[\hat{\nu}_j - q_{j,1-\gamma/2}(\mu), \hat{\nu}_j - q_{j,\gamma/2}(\mu)\Big].$$

By definition, the coverage rate of $\xi_j(\mu_0, \gamma)$ is $1-\gamma$. Call $\xi_j(\mu_0, \gamma)$ the oracle interval. Of course, μ_0 is unknown so I need an estimate $\tilde{\mu}$ in place of μ_0 to construct a practical interval $\xi_j(\tilde{\mu}, \gamma)$. One problem is that the conditional distribution in (2.5) depends on μ due to the selection event and $q_{j,\gamma}(\tilde{\mu})$ is not guaranteed to converge uniformly to $q_{j,\gamma}(\mu_0)$. To alleviate this issue, I propose a method to randomize the point estimate $\tilde{\mu}$, which is motivated by the following conservative construction.

Suppose I have a set $C \subset \mathbb{R}^n$ such that $\mu_0 \in C$. For $\gamma < 1/2$, define

$$q^*_{j,1-\gamma}(C) = \max_{\mu \in C} q_{j,1-\gamma}(\mu), \tag{2.6}$$

$$q^*_{j,\gamma}(C) = \min_{\mu \in C} q_{j,\gamma}(\mu). \tag{2.7}$$

Then it follows that the coverage rate of the interval $[\hat{\nu}_j - q^*_{j,1-\gamma/2}(C), \hat{\nu}_j - q^*_{j,\gamma/2}(C)]$ is at least $1-\gamma$. A possible choice for the set C is a confidence set $\widehat{C}$ for the mean μ_0, unconditional on the selected model. I have the following result about using such an interval for a conservative coverage. Recall $\nu = X_A^+ \mu_0$ and $\hat{\nu} = X_A^+ y$.

Proposition 4. *Suppose $y \sim \mathcal{N}_n(\mu_0, \sigma^2 \mathbf{I}_n)$ and $\widehat{C}$ is a $1-\alpha/2$ confidence set for μ_0, independent of y. Let $\xi^*_j(\widehat{C}) = [\hat{\nu}_j - q^*_{j,1-\alpha/4}(\widehat{C}), \hat{\nu}_j - q^*_{j,\alpha/4}(\widehat{C})]$. Then I have*

$$\mathbb{P}\left\{\nu_j \in \xi^*_j(\widehat{C}) \middle| \mathcal{A}(y) = A\right\} \geq 1 - \alpha. \tag{2.8}$$

Proposition 4 shows that I can construct a valid confidence interval for post-selection inference, provided a $(1-\alpha/2)$ confidence set $\widehat{C}$ for μ_0. Since its length is determined by the worst scenarios over all $\mu \in \widehat{C}$ as in (2.6) and (2.7), the confidence interval $\xi^*_j(\widehat{C})$ can be overly conservative, as shown by my numerical results in Section 2.4.1. Moreover, the assumption that $\widehat{C}$ is independent of y can be strong unless I use sample-splitting, which

does not align well with the purpose of post-selection inference. However, it provides good intuition for the use of the set $\widehat{C}$ in my proposed randomization step, as described in what follows.

Suppose that $\hat{\mu}$ is the center of $\widehat{C} = \widehat{C}(y)$ constructed from y. Let $\tilde{\mu}$ be uniformly distributed over $\widehat{C}$, i.e. $\tilde{\mu} \sim \mathcal{U}(\widehat{C})$, and $q_{j,\gamma}(\widehat{C})$ be the γ-quantile of the distribution

$$[\{X_A^+(y^* - \hat{\mu})\}_j \mid \mathcal{A}(y^*) = A], \tag{2.9}$$

where $y^* = \tilde{\mu} + \varepsilon^*$ and ε^* is independent of $\tilde{\mu}$. Construct an interval $\xi_j(\widehat{C})$ with $q_{j,\gamma}(\widehat{C})$ as

$$\xi_j(\widehat{C}) := [\hat{\nu}_j - q_{j,1-\alpha/4}(\widehat{C}), \hat{\nu}_j - q_{j,\alpha/4}(\widehat{C})]. \tag{2.10}$$

Note that the quantile $q_{j,\gamma}(\widehat{C})$ is calculated from a randomized plug-in estimate $\tilde{\mu}$ over the confidence set $\widehat{C}$, which takes into account the uncertainty in $\tilde{\mu}$. Thus, this interval incorporates more variation than using a fixed point estimate $\hat{\mu}$ as in $\xi_j(\hat{\mu}, \alpha)$.

Below, I present my main algorithm for constructing the confidence interval $\xi_j(\widehat{C})$. Let $\partial\widehat{C}$ denote the boundary of $\widehat{C}$.

Algorithm 1. Constructing interval $\xi_j(\widehat{C})$, $j \in \mathbb{N}_{|A|}$:

1. Draw $\tilde{\mu}^{(k)}$ uniformly from $\partial\widehat{C}$ for $k = 1, \ldots, K$.
2. For each k, draw $\{y^*_{ki}, i = 1, \ldots, N\}$ from $[y^* | \mathcal{A}(y^*) = A]$ where $y^* \sim \mathcal{N}_n(\tilde{\mu}^{(k)}, \sigma^2 \mathbf{I}_n)$.
3. Estimate $q_{j,\gamma}(\widehat{C})$ by the quantiles of $\{[X_A^+(y^*_{ki} - \hat{\mu})]_j, \forall k, i\}$ and construct $\xi_j(\widehat{C})$ (2.10) with the estimated quantiles.

Here, I draw $\tilde{\mu}^{(k)}$ from $\partial\widehat{C}$ for efficiency. Since $\widehat{C}$ is usually an n-dimensional ellipsoid, uniform points in $\widehat{C}$ will be close to its boundary when n is large.

My randomization of the plug-in estimate $\tilde{\mu}$ can be interpreted from a Bayesian perspective, regarding μ_0 as a random vector. As discussed above, a confidence interval can be constructed if I have a good approximation to the distribution $[y^* \mid \mathcal{A}(y^*) = A, \mu_0]$, where

$y^* \mid \mu_0 \sim \mathcal{N}_n(\mu_0, \sigma^2 \mathbf{I}_n)$ is a new response vector independent of y. From a Bayesian perspective, a good approximation that takes into account the uncertainty in μ_0 is the posterior predictive distribution

$$p\big(y^* \mid \mathcal{A}(y^*) = A, y\big) = \int p\big(y^* \mid \mathcal{A}(y^*) = A, \mu_0\big) p(\mu_0 \mid y) d\mu_0,$$

where $p(\mu_0 \mid y)$ is a posterior distribution for μ_0. Regarding $\mathcal{U}(\widehat{C})$, the uniform distribution over $\widehat{C}(y)$, as a posterior distribution for μ_0, steps 1 and 2 in Algorithm 1 can be interpreted as sampling from the above posterior predictive distribution. Drawing $\tilde{\mu}$ in step 1 is equivalent to drawing samples from $p(\mu_0 \mid y)$ and drawing y^* in step 2 is equivalent to sampling from $p(y^* \mid \mathcal{A}(y^*) = A, \tilde{\mu})$, which can be done by my Monte Carlo algorithm to be developed in the next section. In step 3, I find the quantiles of $[X_A^+(y^* - \hat{\mu}) \mid \mathcal{A}(y^*) = A, y]$, where $\hat{\mu}$, the center of $\widehat{C}$, is the posterior mean of μ_0.

2.2.3 Joint inference

Given the samples of y^* drawn by Algorithm 1, I can easily approximate the conditional distribution $[T(y^*) \mid \mathcal{A}(y^*) = A]$ for any function $T(\cdot)$ and carry out many inferential tasks. In particular, I extend my method to the construction of confidence sets for $\nu = X_A^+ \mu_0$.

Recall $\hat{\nu} = X_A^+ y$ and let $q = |A|$. Given a matrix $H \in \mathbb{R}^{m \times q}$ for some $m \leq q$, I wish to make inference on the parameter vector $H\nu \in \mathbb{R}^m$. Generalizing (2.9), let $q_{H,\gamma}(\widehat{C}; \ell_\delta)$ be the γ-quantile of the distribution

$$\Big[\big\| H(X_A^+ y^* - X_A^+ \hat{\mu}) \big\|_\delta \;\Big|\; \mathcal{A}(y^*) = A \Big], \tag{2.11}$$

where $\delta \in [1, \infty]$ specifies a particular ℓ_δ norm used in my construction. From the above Bayesian interpretation, (2.11) approximates $[\|H\hat{\nu} - H\nu\|_\delta \mid \mathcal{A}(y) = A]$ as its posterior predictive estimate. Then I construct a $1 - \alpha$ confidence set for $H\nu$ as an ℓ_δ ball

$$\xi_H(\widehat{C}; \ell_\delta) := \Big\{ \eta \in \mathbb{R}^m : \|\eta - H\hat{\nu}\|_\delta \leq q_{H,1-\alpha/2}(\widehat{C}; \ell_\delta) \Big\}, \tag{2.12}$$

where $\widehat{C}$, as in (2.10), is a $1-\alpha/2$ confidence set for the mean μ_0. For example, one can construct a confidence set for ν by letting $H = \mathbf{I}_q$. If one is interested in constructing a confidence set for the first two components in A, I can let $H = [e_1, e_2]^\mathsf{T}$, where e_j is the j-th standard basis vector in $\mathbb{R}^q$. In general, $\xi_H(\widehat{C}; \ell_\delta)$ is a confidence set for some linear transformation of ν.

Now the remaining question is how to build the $(1-\alpha/2)$ confidence set $\widehat{C}$ for μ_0, unconditional on the selected model. There are a few methods that may be used to construct such a confidence set for high-dimensional regression problems (Nickl and van de Geer, 2013; Zhou et al., 2019). I apply two different methods in this work. The first method is a two-step method, consisting of a projection and a shrinkage step (Zhou et al., 2019). This method builds an ellipsoid-shaped confidence set with different radii for strong and weak signals. The radius and center for weak signals are constructed using Stein's method. Denote by $\widehat{C}_S$ and $\hat{\mu}_S$ the confidence set and its center by this method. It is shown by Zhou et al. (2019) that $\widehat{C}_S$ is asymptotically honest,

$$\liminf_{n\to\infty} \inf_{\mu_0 \in \mathbb{R}^n} \mathbb{P}(\mu_0 \in \widehat{C}_S) \geq 1-\alpha/2,$$

where $\mathbb{P}$ is taken with respect to the distribution of y in (2.1). However, this method replies on sample-splitting in its construction, which adds another level of complexity in the application of my post-selection inference. Thus, I develop a second and simpler method, based on a given subset of covariates X_A. Let $A_0 = \mathrm{supp}(\beta_0)$ be the true support such that $\mu_0 = X_{A_0}\beta_{0A_0}$. If $A_0 \subset A$, then $X_A^+ y \sim \mathcal{N}_{|A|}(\beta_{0A}, \sigma^2(X_A^\mathsf{T} X_A)^{-1})$. From this fact, I build a confidence set $\widehat{D}$ for β_{0A} which defines a confidence set $\widehat{C} = X_A \widehat{D}$ for μ_0. The confidence set and its center built this way are denoted by $\widehat{C}_A$ and $\hat{\mu}_A$. A convenient choice of A would be $\mathcal{A}(y)$, the support of lasso, although under this choice $\widehat{C}_A$ is not guaranteed to achieve the nominal confidence level, unless $\mathbb{P}(\mathcal{A}(y) = A) \to 1$ for some $A \supset A_0$ as $n \to \infty$. However, I am only using $\widehat{C}_A$ as a mechanism to randomize the plug-in estimate $\tilde{\mu}$, and thus it does not have to be a valid confidence set. I will compare the performance of these two methods in Section 2.4.1 on simulated data. The comparison suggests that the second method usually

achieves comparable coverage as the first method, while being more coherent with my post-selection inference procedure in practice. Therefore, I use the second method by default for all the numerical results in this work.

Now I summarize the steps of my post-selection inference on $H\nu$.

Algorithm 2. Constructing $\xi_H(\widehat{C}; \ell_\delta)$:

1. Construct $(1-\alpha/2)$ confidence set $\widehat{C}_A$ centering at $\hat{\mu}_A$.
2. Apply Algorithm 1 with $\widehat{C} = \widehat{C}_A$ and $\hat{\mu} = \hat{\mu}_A$.
3. Estimate $q_{H,\gamma}(\widehat{C}; \ell_\delta)$ by the quantile of $\{\|HX_A^+(y^*_{ki} - \hat{\mu}_A)\|_\delta, \forall k, i\}$ and construct $\xi_H(\widehat{C}; \ell_\delta)$ in (2.12) with the estimated quantile.

Remark 2. I have implicitly assumed a fixed tuning parameter λ in (2.3) so far, but I observe that my method works well even using a λ chosen in a data-dependent way. This is very appealing in applications: One may simply use lasso, with a data-dependent λ, to identify potentially importance variables, followed by my inference tool to construct an interval for each. Although not the focus, for low-dimensional data ($p < n$), the confidence set $\widehat{C}$ for μ_0 can be constructed with $A = \mathbb{N}_p$ by the sampling distribution of the least-squares estimator, which is certainly valid with the nominal confidence level.

2.2.4 An illustration

In Section 2.2.2, I covered four different methods for constructing $\widehat{I}_j(\alpha)$. First, the oracle interval $\xi_j(\mu_0, \alpha)$ is constructed assuming the true mean μ_0 is known (the oracle). This is not a practical method and is used for illustration only. Second, $\xi_j(\hat{\mu}, \alpha)$ uses an estimate $\hat{\mu}$ in place of μ_0. My main proposal $\xi_j(\widehat{C})$, presented in Algorithm 1, randomizes the plug-in estimate of μ_0 by uniform sampling over the boundary of $\widehat{C}$. Lastly, the interval $\xi_j^*(\widehat{C})$ defined in Proposition 4 controls the worst case over $\widehat{C}$. A detailed comparison among the four methods will be conducted in Section 2.4.1. In general, the oracle interval $\xi_j(\mu_0, \alpha)$ reaches the nominal coverage rate with the shortest interval length, the coverage of $\xi_j(\hat{\mu}, \alpha)$

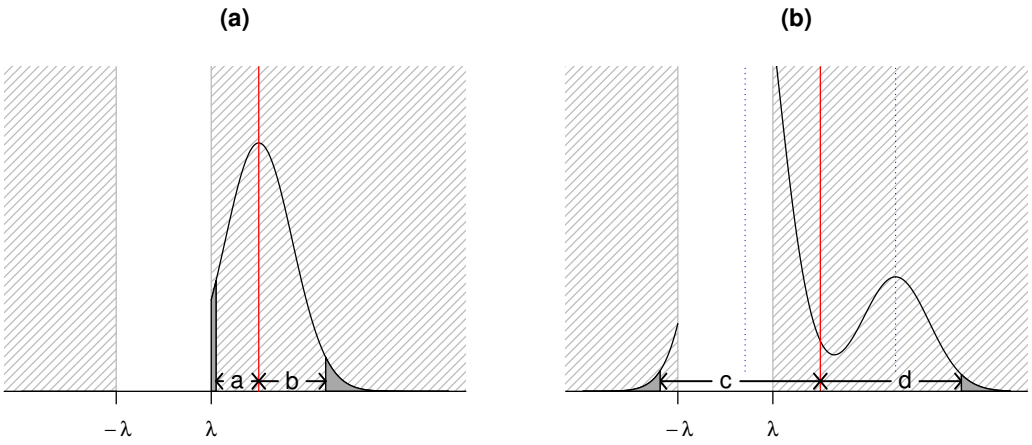

Figure 2.1: Illustration of the confidence intervals (a) $\xi(\hat{\mu})$ and (b) $\xi(\widehat{C})$ in one-dimension. The red lines represents $\hat{\nu} = \hat{\beta}^{\mathrm{LS}}$ and the two blue dotted lines indicate the boundaries of $\widehat{C}$.

tends to be lower than the desired level, while $\xi_j^*(\widehat{C})$ seems too conservative. The interval $\xi_j(\widehat{C})$ reaches a good compromise between coverage and interval length.

Here, I illustrate the difference between $\xi_j(\hat{\mu}, \alpha)$ and $\xi_j(\widehat{C})$ for $p = 1$, assuming $\|X_1\|^2 = n$. In this case, the lasso $\hat{\beta} = \mathrm{sgn}(\hat{\beta}^{\mathrm{LS}})(|\hat{\beta}^{\mathrm{LS}}| - \lambda)_+$, where $\hat{\beta}^{\mathrm{LS}} = X_1^{\mathsf{T}} y/n$ is the least-squares estimate. The distribution of $\hat{\beta}^{\mathrm{LS}}$ given $\mathcal{A}(y) = \{1\}$ is truncated to the intervals $(-\infty, -\lambda) \cup (\lambda, \infty) := T$ (shaded regions in Figure 2.1). Write the two confidence intervals as $\xi(\hat{\mu})$ and $\xi(\widehat{C})$, where $\hat{\mu} = X_1\hat{\beta}^{\mathrm{LS}}$ and $\widehat{C}$ projected to X_1 is an interval $(\hat{\beta}^{\mathrm{LS}} - \Delta, \hat{\beta}^{\mathrm{LS}} + \Delta)$, centered at $\hat{\beta}^{\mathrm{LS}}$ between the two blue dotted lines in panel (b). Both $\xi(\hat{\mu})$ and $\xi(\widehat{C})$ are centered at $\hat{\nu} = \hat{\beta}^{\mathrm{LS}}$, but with different end points. Let $\mathcal{TN}_d(\mu, \Sigma, \mathcal{V})$ denote $\mathcal{N}_d(\mu, \Sigma)$ truncated to the set $\mathcal{V}$. The interval $\xi(\hat{\mu}) = [\hat{\nu} - b, \hat{\nu} + a]$ is constructed based on the quantiles of

$$X_1^{\mathsf{T}} y^*/n \mid \mathcal{A}(y^*) = \{1\} \sim \mathcal{TN}(\hat{\beta}^{\mathrm{LS}}, \sigma^2/n, T),$$

indicated by the dark tail regions in Figure 2.1(a), where $y^* \sim \mathcal{N}_n(\hat{\mu}, \sigma^2 \mathbf{I}_n)$. On the contrary, as shown in Figure 2.1(b), the interval $\xi(\widehat{C}) = [\hat{\nu} - d, \hat{\nu} + c]$ is constructed from the quantiles of a mixture of two truncated normal distributions, i.e. $\mathcal{TN}(\hat{\beta}^{\mathrm{LS}} \pm \Delta, \sigma^2/n, T)$, each centered at a boundary of the interval $\widehat{C}$ (after being projected to X_1). If the true parameter $\beta_0 \in (-\lambda, \lambda)$ and close to zero, then $\xi(\widehat{C})$ is likely to cover β_0 while the other interval $\xi(\hat{\mu})$ will fail. In fact,

the difficulty in post-selection inference largely stems from such a situation in which some β_{0j} is very close to zero and consequently the conditional distribution of y given the selection event can change substantially with the mean μ_0. My method tackles this difficult problem by simulating from a mixture of such conditional distributions with mean $\tilde{\mu}$ randomizing over a suitable neighborhood of μ_0.

2.3 Conditional sampling

In this section, I develop an MCMC sampler to draw y^* such that $\mathcal{A}(y^*) = A$, which is the key conditional sampling step in my method. My sampler is based on the idea of estimator augmentation which was introduced in Chapter 1.2.1.

2.3.1 The target distribution

As an immediate consequence of Theorem 1, I can obtain a density for the conditional distribution $[\hat{\beta}_A, S_I \mid \mathcal{A} = A]$ for a fixed subset A, which is directly related to my target conditional sampling problem.

Corollary 5. *Under the same assumptions of Theorem 1, the conditional distribution* $[\hat{\beta}_A, S_I \mid \mathcal{A} = A]$ *is given by*

$$\mathbb{P}_{\Theta|\mathcal{A}}(d\theta|A) \propto f_R\big(V_R^{\mathsf{T}} H(\theta, A; \mu_0, \lambda); \sigma^2\big) d\theta := \pi(\theta \mid A; \mu_0, \sigma^2, \lambda) d\theta, \tag{2.13}$$

where $\theta = (b_A, s_I)$ *satisfies the constraints in* (1.7) *and* (1.8).

The conditional distribution $[\hat{\beta}_A, S_I \mid \mathcal{A} = A]$ has an especially simple density $\pi(\theta \mid A)$, which is just an n-variate density with respect to a fixed parameterization $(b_A, s_F) \in \mathbb{R}^n$ as the active set $\mathcal{A}$ is fixed to A and the set F only depends on A. See Corollary 1 in Zhou (2014) for a more detailed discussion on the truncated Gaussian nature of π.

Given the density in Corollary 5, I develop a Metropolis-Hastings (MH) sampler to draw $(\hat{\beta}_A, S_I)$ given the fixed active set, $\mathcal{A} = A$. This will achieve my goal of sampling $[y \mid \mathcal{A}(y) =$

A] because of the following result:

Theorem 6. *Suppose the assumptions of Theorem 1 hold and* $(\hat{\beta}_A^*, S_I^*)$ *follows the distribution* (2.13)*. Then I have*

$$[y \mid \mathcal{A}(y) = A] = \left[X_A \hat{\beta}_A^* + n\lambda (X^\mathsf{T})^+ \left\{W_A \operatorname{sgn}(\hat{\beta}_A^*) + W_I S_I^*\right\}\right], \tag{2.14}$$

and consequently,

$$[X_A^+ y \mid \mathcal{A}(y) = A] = \left[\hat{\beta}_A^* + n\lambda (X_A^\mathsf{T} X_A)^{-1} W_{AA} \operatorname{sgn}(\hat{\beta}_A^*)\right]. \tag{2.15}$$

As described in Algorithm 1, I wish to draw $[y^* \mid \mathcal{A}(y^*) = A]$ for $y^* \sim \mathcal{N}_n(\tilde{\mu}, \sigma^2 \mathbf{I}_n)$. Once I have drawn $(\hat{\beta}_A^*, S_I^*)$ from the density $\pi(\theta \mid A; \tilde{\mu}, \sigma^2, \lambda)$ (2.13), I can easily obtain a sample of y^* by (2.14), which follows the target conditional distribution. Note that only $X_A^+ y^*$ is needed in (2.9) and (2.11), which can be calculated directly with (2.15).

To provide an intuitive understanding of the conditional distributions in (2.13) and (2.14), let us consider a simple example with $n > p = 2$, $\Psi = \mathbf{I}_2$, $\mu_0 = X\beta_0$ and $A = \{1\}$. In this low-dimensional setting, $\operatorname{null}(X) = \{0\}$ and thus the constraint (1.7) is trivially satisfied for all $s \in \mathbb{R}^2$ (as $V_N = 0$). As shown in Figure 2.2(a), the sample space of $(\hat{\beta}_A, S_I) = (\hat{\beta}_1, S_2)$ is

$$(-\infty, 0) \times [-1, 1] \cup (0, \infty) \times [-1, 1] := \Omega_{-1} \cup \Omega_1,$$

which is an essentially connected set (i.e. having a connected closure). Since $\Psi = \mathbf{I}_2$, I may choose $V_R = \mathbf{I}_2$, whose columns form an orthonormal basis for $\operatorname{row}(X) = \mathbb{R}^2$, and under this choice f_R is the density of $\mathcal{N}_2(0, \sigma^2 \mathbf{I}_2/n)$. The contours of $[(\hat{\beta}_1, S_2) \mid \mathcal{A} = \{1\}]$ are shown in Figure 2.2(a). It is easier to understand this distribution if further conditioning on $\operatorname{sgn}(\hat{\beta}_1) = s_1$:

$$(\hat{\beta}_1, S_2) \mid (\mathcal{A} = \{1\}, \operatorname{sgn}(\hat{\beta}_1) = s_1) \sim \mathcal{TN}_2(\mu(s_1), \Sigma, \Omega_{s_1}), \qquad s_1 \in \{-1, 1\},$$

which is a bivariate normal distribution truncated to Ω_{s_1} for each s_1. The mean and covari-

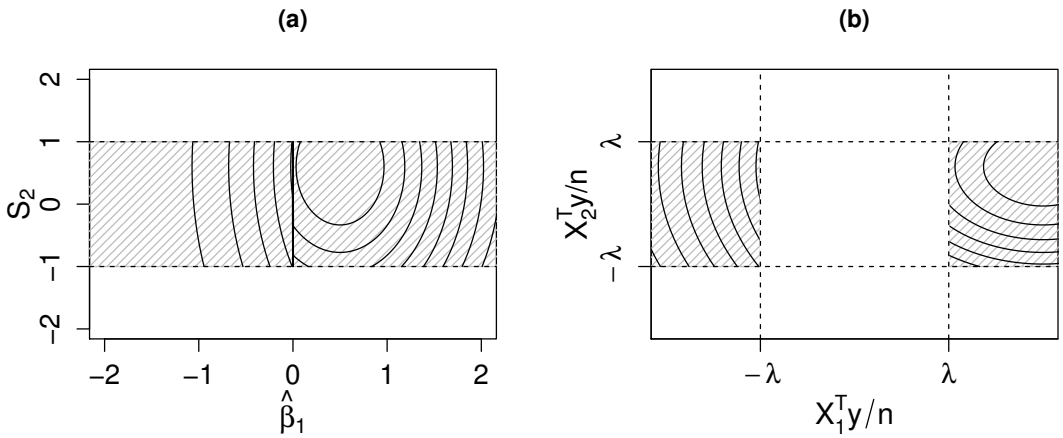

Figure 2.2: The conditional distributions of (a) $(\hat{\beta}_1, S_2)$ and (b) $X^{\mathsf{T}}y/n$ given $A = \{1\}$ for $p = 2$.

ance matrix are

$$\mu(s_1) = \begin{pmatrix} \beta_{01} - \lambda s_1 \\ \beta_{02}/\lambda \end{pmatrix}, \quad \Sigma = \frac{\sigma^2}{n} \begin{pmatrix} 1 & 0 \\ 0 & 1/\lambda^2 \end{pmatrix}.$$

The two sets of contours in panel (a), separated by the line segment $\{0\} \times [-1, 1]$, correspond to the two truncated normal distributions with different centers, $\mu(1)$ and $\mu(-1)$. Figure 2.2(b) plots the contours of the conditional distribution of $X^{\mathsf{T}}y/n$ given $\mathcal{A} = \{1\}$, which is a bivariate normal distribution $\mathcal{N}_2(\beta_0, \sigma^2\mathbf{I}_2/n)$ truncated to the union of two disconnected regions,

$$(-\infty, -\lambda) \times [-\lambda, \lambda] \cup (\lambda, \infty) \times [-\lambda, \lambda].$$

The contrast between the two sample spaces illustrates the potential advantage in designing Monte Carlo algorithms in the space of the augmented estimator $(\hat{\beta}_A, S_I)$ over the space of y.

2.3.2 A Metropolis-Hastings sampler

In what follows, I describe my MH sampler in detail. For notational brevity, write the target density as $\pi(\theta) \equiv \pi(\theta \mid A; \tilde{\mu}, \sigma^2, \lambda)$ hereafter, where the space of $\theta = (b_A, s_I)$ is defined by (1.7) and (1.8). These constraints must be satisfied in each step of the sampling process, which presents a technical challenge for my Monte Carlo algorithm. I adopt a coordinate-wise update of θ. Let $\theta^{(t)}$ be the value of θ at the t-th iteration. After proposing a new value $\theta_i^\dagger$ for its i-th component, the MH ratio is computed as

$$\zeta = \min\left\{1, \frac{\pi(\theta^\dagger)}{\pi(\theta^{(t)})} \frac{q(\theta^\dagger, \theta^{(t)})}{q(\theta^{(t)}, \theta^\dagger)}\right\},$$

where $q(\theta^{(t)}, \theta^\dagger)$ is the transition kernel of the proposal $\theta^\dagger$ given $\theta^{(t)}$. If $\theta^\dagger$ is accepted, let $\theta^{(t+1)} = \theta^\dagger$. Otherwise, I reuse the previous state, i.e. $\theta^{(t+1)} = \theta^{(t)}$.

I first derive explicit expressions for the feasible region of θ defined by (1.7) and (1.8). For the sake of notational simplicity, put

$$G = V_{IN}^\mathsf{T} W_{II} \in \mathbb{R}^{(p-n)\times|I|}, \qquad u = u(s_A) = -V_{AN}^\mathsf{T} W_{AA} \operatorname{sgn}(b_A) \in \mathbb{R}^{p-n},$$

where $s_A = \operatorname{sgn}(b_A)$, and rewrite (1.7) as $Gs_I = u$. Recall $|I| = p - |A|$ and $q = |A|$. Since $p - n$ constraints are imposed on s_I, there are only $n - q$ free coordinates in s_I. Partition s_I into free and dependent components and denote them by $s_F \in \mathbb{R}^{n-q}$ and $s_D \in \mathbb{R}^{p-n}$, respectively. Partition the columns of G accordingly. Then (1.7) can be rewritten

$$G_F s_F + G_D s_D = u \iff s_D = G_D^{-1}(u - G_F s_F), \tag{2.16}$$

which shows that s_D is a function of $(b_A, s_F) \in \mathbb{R}^n$. Now the feasible region for θ can be equivalently defined by

$$\|s_F\|_\infty \le 1, \qquad \|G_D^{-1}(u(s_A) - G_F s_F)\|_\infty \le 1. \tag{2.17}$$

Note that every time I update any component of (b_A, s_F), s_D needs to be updated accordingly

via (2.16). Below, I provide details about how to draw (b_A, s_F), the free coordinates of θ, given the current value $(b_A^{(t)}, s_F^{(t)})$. I assume that $(b_A^{(t)}, s_F^{(t)})$ is feasible and satisfies (2.17).

For the active coefficients b_A, a normal distribution is used as the proposal,

$$b_i^\dagger | b_i^{(t)} \sim \mathcal{N}(b_i^{(t)}, \tau_i^2), \qquad i \in A.$$

By using a symmetric proposal distribution, the MH ratio becomes the ratio of the target densities only,

$$\zeta = \min\left\{1, \frac{\pi(\theta^\dagger)}{\pi(\theta^{(t)})}\right\} = \min\left\{1, \frac{f_R\big(V_R^\mathsf{T} H(\theta^\dagger, A; \tilde{\mu}, \lambda); \sigma^2\big)}{f_R\big(V_R^\mathsf{T} H(\theta^{(t)}, A; \tilde{\mu}, \lambda); \sigma^2\big)}\right\}. \tag{2.18}$$

Under this proposal, $s_F^\dagger = s_F^{(t)}$ is unchanged. If $\mathrm{sgn}(b_i^\dagger) = \mathrm{sgn}(b_i^{(t)})$, then $s_A^\dagger = s_A^{(t)}$. Consequently, $\theta^\dagger$ satisfies the constraints in (2.17) and thus is feasible. If $\mathrm{sgn}(b_i^\dagger) \neq \mathrm{sgn}(b_i^{(t)})$, then $s_A^\dagger$ is different from $s_A^{(t)}$, with the i-th element replaced by $\mathrm{sgn}(b_i^\dagger)$. I need to verify the second inequality in (2.17). Let $u^\dagger = u(s_A^\dagger)$. If $\|G_D^{-1}(u^\dagger - G_F s_F^\dagger)\|_\infty \leq 1$, then $\theta^\dagger$ is feasible and I compute the MH ratio as in (2.18). Otherwise, I move to the next component in A.

When updating each component in s_F, denoted by $(s_F)_k$, it would be inefficient to use a naive proposal distribution, such as $\mathcal{U}(-1, 1)$, since it does not guarantee every component of $s_D^\dagger$ will stay in $[-1, 1]$. A better approach is to compute the feasible range of $(s_F)_k$. Holding s_A and $(s_F)_{-k}$ as constants, the second inequality in (2.17) defines $2(p-n)$ linear constraints on $(s_F)_k$,

$$-\mathbf{1}_{[p-n]} + G_D^{-1}U - (G_D^{-1}G_F)_{-k}(s_F)_{-k} \leq (G_D^{-1}G_F)_k(s_F)_k \leq \mathbf{1}_{[p-n]} + G_D^{-1}U - (G_D^{-1}G_F)_{-k}(s_F)_{-k},$$

from which the feasible range of $(s_F)_k$, $[LB_k, UB_k]$, can be computed,

$$LB_k = \max\left\{-1, M^{-1}\big(-\mathbf{1}_{[p-n]} + G_D^{-1}u - (G_D^{-1}G_F)_{-k}(s_F)_{-k}\big)\right\}, \tag{2.19}$$

$$UB_k = \min\left\{1, M^{-1}\big(\mathbf{1}_{[p-n]} + G_D^{-1}u - (G_D^{-1}G_F)_{-k}(s_F)_{-k}\big)\right\}, \tag{2.20}$$

Algorithm 3 $MH(\tilde{\mu}, \sigma, \lambda)$

Given $(b_A, s_F)^{(t)}$,
for $i \in A$ **do**
  draw $b_i^\dagger \sim \mathcal{N}(b_i^{(t)}, \tau_i^2)$.
  if $\mathrm{sgn}(b_i^\dagger) \neq \mathrm{sgn}(b_i^{(t)})$ and $\theta^\dagger$ is infeasible **then**
    continue.
  else
    $b_i^{(t+1)} \leftarrow b_i^\dagger$ with probability ζ; otherwise $b_i^{(t+1)} \leftarrow b_i^{(t)}$.
    $b_{A\setminus i}^{(t+1)} \leftarrow b_{A\setminus i}^{(t)}$, $s_F^{(t+1)} \leftarrow s_F^{(t)}$, $t \leftarrow t+1$.
  end if
end for
for $k \in \mathbb{N}_{|F|}$ **do**
  compute $LB_k^{(t)}$ and $UB_k^{(t)}$ by (2.19) and (2.20).
  draw $(s_F)_k^\dagger \sim \mathcal{U}(LB_k^{(t)}, UB_k^{(t)})$.
  $(s_F)_k^{(t+1)} \leftarrow (s_F)_k^\dagger$ with probability ζ; otherwise $(s_F)_k^{(t+1)} \leftarrow (s_F)_k^{(t)}$.
  $b_A^{(t+1)} \leftarrow b_A^{(t)}$, $(s_F)_{-k}^{(t+1)} \leftarrow (s_F)_{-k}^{(t)}$, $t \leftarrow t+1$.
end for

where $M = \mathrm{diag}\left((G_D^{-1}G_F)_k\right)$ is a $(p-n) \times (p-n)$ diagonal matrix having the k-th column of $G_D^{-1}G_F$ as its diagonal elements. Calculate $LB_k^{(t)}$ and $UB_k^{(t)}$ with $u^{(t)} = u(s_A^{(t)})$ and $(s_F^{(t)})_{-k}$. Note that $LB_k^{(t)} < UB_k^{(t)}$ since the current value $(s_F^{(t)})_k \in [LB_k^{(t)}, UB_k^{(t)}]$ by assumption. Propose $(s_F)_k^\dagger$ from $\mathcal{U}(LB_k^{(t)}, UB_k^{(t)})$ and compute $s_D^\dagger$ by plugging $s_F^\dagger$ and $u^\dagger = u^{(t)}$ in (2.16). Because $[LB_k, UB_k]$ does not depend on the current value of $(s_F)_k$, this proposal is symmetric, $q(\theta^\dagger, \theta^{(t)}) = q(\theta^{(t)}, \theta^\dagger)$. Therefore, the MH ratio again reduces to (2.18).

Putting the above pieces together I present the MH sampler in Algorithm 3. To highlight its dependency on $(\tilde{\mu}, \sigma, \lambda)$, I denote this algorithm by $MH(\tilde{\mu}, \sigma, \lambda)$.

2.3.3 Examples

Using a small dataset of size $(n, p) = (5, 10)$, I compared my MH sampler with parametric bootstrap which provided the ground truth here. I estimated μ_0 by $\tilde{\mu} = X\hat{\beta}$, where $\hat{\beta}$ is the lasso estimate. The active set chosen by the lasso was $A = \{6, 9\}$. In parametric bootstrap, I simulated $y^* \sim \mathcal{N}_n(\tilde{\mu}, \sigma^2 \mathbf{I}_n)$ and found the lasso solution $\hat{\beta}(y^*)$. If the support of $\hat{\beta}(y^*)$ was indeed $\{6, 9\}$, the sample $\hat{\beta}(y^*)$ would be accepted. I ran this bootstrap method until I accepted $10,000$ samples whose active set $\mathcal{A}(y^*) = A$. This is essentially a naive rejection

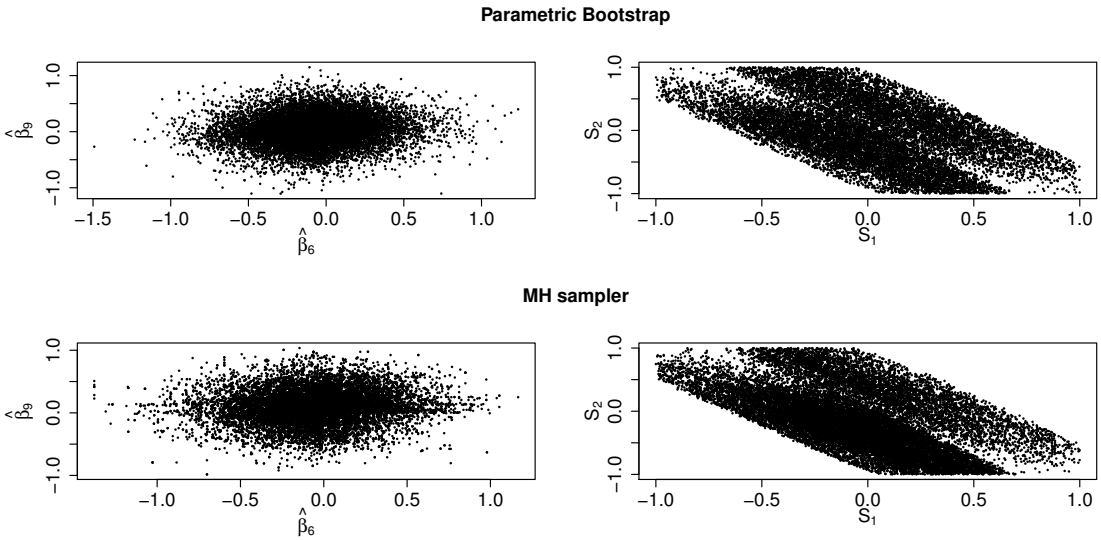

Figure 2.3: Comparison between bootstrap samples (top) and MH samples (bottom). The left column shows the scatter plot of $(\hat{\beta}_6, \hat{\beta}_9)$ while the right column shows the scatter plot of (S_1, S_2). These are the first two components in A and I, respectively.

sampling method. Note that the bootstrap is only applicable for this small dataset. Even for such a small dataset, the number of bootstrap samples simulated in order to obtain 10,000 samples was 1.5×10^5, i.e., the acceptance rate was only 6.67%. This demonstrates the necessity of my MH sampler for this conditional sampling problem. The MH sampler was then used to draw $20,000$ samples. See Figure 2.3 for a comparison between the samples generated by the two methods. It can be seen from the scatter plots that the results of my MH sampler were very close to the exact samples generated by the bootstrap, providing a numerical validation of my algorithm.

I present a quick visualization of the MH samples on a bigger dataset of size $(n, p) = (50, 100)$. See Figure 2.4 for summary plots of the samples for the first two active coefficients, $\hat{\beta}_1$ and $\hat{\beta}_4$. The autocorrelation plots and the sample path plots show that my MH sampler was quite efficient with a fast decay in autocorrelation.

2.4 Numerical results

In this section, I examine the performance of my method by providing simulation results under various settings. In Section 2.4.1, I show the effectiveness of the proposed randomization

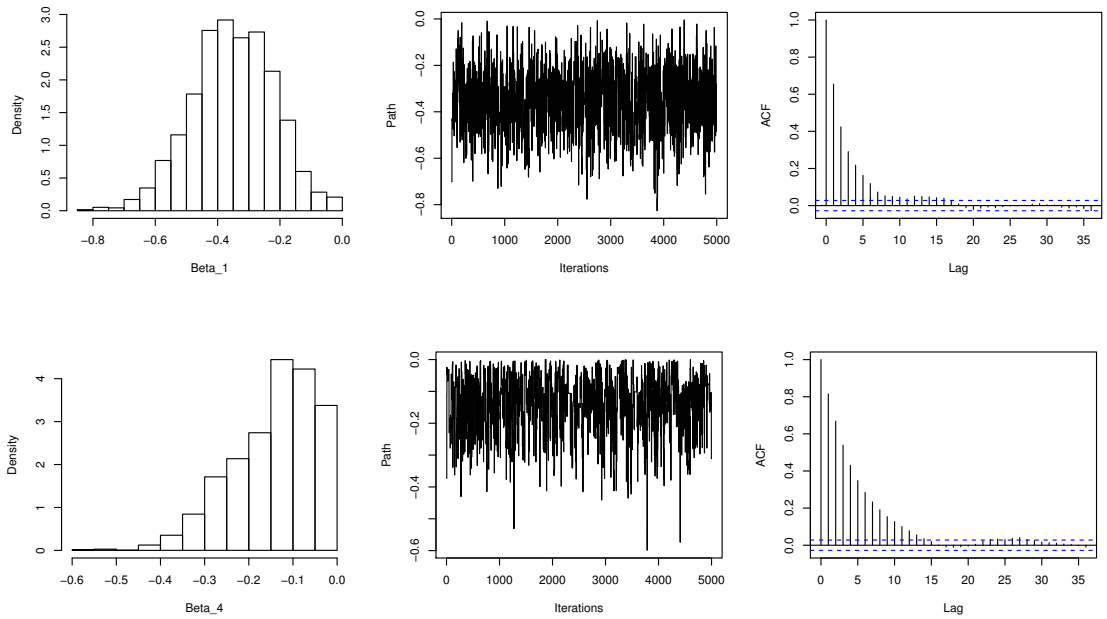

Figure 2.4: Summary plots for $\hat{\beta}_1$ and $\hat{\beta}_4$ from my MH sampler when $A = \{1, 4, 5\}$: histogram, sample trace, and autocorrelation function.

of the plug-in estimate. Section 2.4.2 examines the robustness of my method with regard to the lasso tuning parameter λ. In Section 2.4.3, my confidence intervals are compared with those built by Lee's method. Section 2.4.4 provides simulation results for the construction of confidence sets by my method. A detailed case study is presented in Section 2.4.5 to clarify the differences between my method and Lee's method.

2.4.1 The effect of randomization

To see the effect of my randomization step, I compare four different confidence intervals defined in Section 2.2.2:

(1) $\xi_j(\mu_0) = [\hat{\nu}_j - q_{j,1-\alpha/2}(\mu_0), \hat{\nu}_j - q_{j,\alpha/2}(\mu_0)]$ (oracle);

(2) $\xi_j(\hat{\mu}) = [\hat{\nu}_j - q_{j,1-\alpha/2}(\hat{\mu}), \hat{\nu}_j - q_{j,\alpha/2}(\hat{\mu})],\ \hat{\mu} \in \{\hat{\mu}_A, \hat{\mu}_S\}$;

(3) $\xi_j(\widehat{C}) = [\hat{\nu}_j - q_{j,1-\alpha/4}(\widehat{C}), \hat{\nu}_j - q_{j,\alpha/4}(\widehat{C})],\ \widehat{C} \in \{\widehat{C}_A, \widehat{C}_S\}$;

(4) $\xi^*_j(\widehat{C}) = [\hat{\nu}_j - q^*_{j,1-\alpha/4}(\widehat{C}), \hat{\nu}_j - q^*_{j,\alpha/4}(\widehat{C})],\ \widehat{C} \in \{\widehat{C}_A, \widehat{C}_S\}$.

Recall that $\hat{\nu} = X_A^+ y$ and see (2.5), (2.6), (2.7) and (2.9) for the definitions of $q_{j,\gamma}(\mu)$, $q_{j,\gamma}(\widehat{C})$

and $q^*_{j,\gamma}(\widehat{C})$. As I described in Section 2.2.2, there are two ways of constructing $\widehat{C}$, with center $\hat{\mu}$. The subscripts A and S are used to distinguish the two methods. Again, for intervals (3) and (4), $(\alpha/4, 1-\alpha/4)$-quantiles are used due to Bonferroni correction.

Algorithm 1 is used to construct interval (3). Likewise, for interval (4), I draw $\{\tilde{\mu}^{(i)}\}_{i=1}^{K}$ from $\partial\widehat{C}$ and estimate $q^*_{j,\alpha/4}(\widehat{C})$ and $q^*_{j,1-\alpha/4}(\widehat{C})$ by

$$q^*_{j,\alpha/4}(\widehat{C}) = \min_{1\le i\le K} q_{j,\alpha/4}(\tilde{\mu}^{(i)}), \qquad q^*_{j,1-\alpha/4}(\widehat{C}) = \max_{1\le i\le K} q_{j,1-\alpha/4}(\tilde{\mu}^{(i)}).$$

I set $K = 20$, and given each $\tilde{\mu}^{(i)}$, I sampled 500 y^*'s, i.e. the total number of samples used for intervals (3) and (4) was $20 \times 500 = 10,000$. For a fair comparison, I fixed the number of samples to be $10,000$ for (1) and (2). Note that my MH sampler was used in all the four methods to draw from $[y^* \mid \mathcal{A}(y^*) = A]$ and estimate the quantiles $q_{j,\gamma}(\mu)$. Twenty datasets with $(n, p, A_0) = (50, 100, \mathbb{N}_5)$ were simulated. The true coefficients β_{0A_0} were drawn from $\mathcal{U}(-1, 1)$. Each row of X was independently sampled from $\mathcal{N}_p(0, \Sigma)$. I considered three types of covariance matrix Σ in this comparison:

- Identity (I): $\Sigma = \mathbf{I}_p$,
- Toeplitz (T): $\Sigma_{ij} = 0.5^{|i-j|}$,
- Exponential Decay (ED): $\Sigma^{-1}_{ij} = 0.4^{|i-j|}$.

The significance level α was set to 0.05 and $\sigma^2 = 1$ was assumed to be known. For each dataset, λ was chosen by cross-validation with the one standard error rule.

The following metrics are used to compare the results. For a subset $E \subset \mathbb{N}_p$ and confidence intervals, $\widehat{I}_j$ for $j \in A$, I define power and coverage by averaging over the variables in the set E:

$$\text{Power} = \textstyle\sum_{j\in E} \mathbb{P}\Big(0 \notin \widehat{I}_j\Big)/|E|,$$

$$\text{Coverage} = \textstyle\sum_{j\in E} \mathbb{P}\Big(\nu_j \in \widehat{I}_j\Big)/|E|.$$

Table 2.1: Power and coverage rate for (1) $\xi(\mu_0)$ (oracle), (2) $\xi(\hat{\mu})$, (3) $\xi(\widehat{C})$ and (4) $\xi^*(\widehat{C})$.

Σ	Method	Power	Coverage		
		$A_0 \cap A$	A	$A_0 \cap A$	$A_0^c \cap A$
I	(1)	1.000	0.960(0.379)	0.960(0.471)	0.962(0.202)
	(2_A)	0.980	0.882(0.493)	0.880(0.555)	0.885(0.373)
	(3_A)	0.760	0.934(0.751)	0.900(0.857)	1.000(0.546)
	(4_A)	0.800	1.000(0.881)	1.000(0.874)	1.000(0.895)
	(2_S)	0.880	0.618(0.479)	0.600(0.520)	0.654(0.401)
	(3_S)	0.580	0.934(0.850)	0.900(0.973)	1.000(0.613)
	(4_S)	0.740	1.000(0.975)	1.000(0.980)	1.000(0.965)
T	(1)	0.978	0.955(0.426)	0.956(0.542)	0.954(0.191)
	(2_A)	0.978	0.821(0.562)	0.933(0.634)	0.591(0.416)
	(3_A)	0.711	0.970(0.852)	0.956(0.963)	1.000(0.624)
	(4_A)	0.800	0.985(0.945)	1.000(0.966)	0.954(0.902)
	(2_S)	0.800	0.642(0.537)	0.667(0.593)	0.591(0.421)
	(3_S)	0.489	0.970(0.962)	0.956(1.106)	1.000(0.668)
	(4_S)	0.689	0.985(1.044)	1.000(1.077)	0.954(0.979)
ED	(1)	0.967	0.957(0.361)	0.951(0.456)	0.969(0.180)
	(2_A)	0.934	0.914(0.451)	0.918(0.521)	0.906(0.318)
	(3_A)	0.754	0.957(0.678)	0.934(0.802)	1.000(0.441)
	(4_A)	0.721	1.000(0.816)	1.000(0.831)	1.000(0.785)
	(2_S)	0.869	0.774(0.427)	0.721(0.472)	0.875(0.340)
	(3_S)	0.639	0.957(0.752)	0.934(0.879)	1.000(0.511)
	(4_S)	0.688	1.000(0.901)	1.000(0.908)	1.000(0.889)

The subscripts A and S indicate two ways of estimating $\widehat{C}$ and its center $\hat{\mu}$. The average length of confidence intervals is reported in the parentheses.

A few informative choices for E are A, $A_0 \cap A$ and $A_0^c \cap A$. The set A includes all the variables selected by lasso, while the sets $A_0 \cap A$ and $A_0^c \cap A$ separate the true positive and the false positive variables. I report in Table 2.1 the average coverage rate over variables in each of the three sets and the power of detecting true positive variables $A_0 \cap A$ for each of the four methods. I omit the subscript j to simplify my notation and to indicate averaging over a subset of indices, such as $j \in A$.

The coverage rate of $\xi(\mu_0)$ was at the desired level while its average length was the shortest among all the methods. This is an obvious result, since the true parameter μ_0 is assumed to be known (the oracle). Using a single plug-in estimate, $\xi(\hat{\mu})$ produced shorter

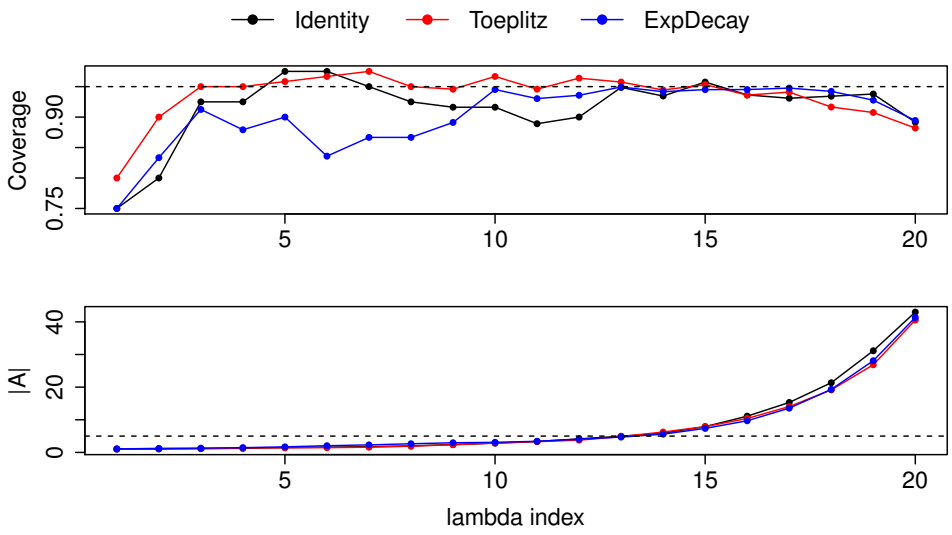

Figure 2.5: Sensitivity test for the choice of λ with the datasets of $(n, p, A_0) = (50, 100, \mathbb{N}_5)$. Each point is the average from 20 datasets.

confidence intervals (CIs) compared to $\xi(\widehat{C})$ and $\xi^*(\widehat{C})$. However, the coverage rate of $\xi(\hat{\mu})$ was much lower than the nominal level, especially for $j \in A_0^c \cap A$. On the contrary, with randomized $\tilde{\mu}$ drawn from the confidence set $\widehat{C}$, the CIs of $\xi(\widehat{C})$ achieved the desired coverage rate, which demonstrates the importance of my proposed randomization step. The intervals $\xi^*(\widehat{C})$ showed a similar effect as $\xi(\widehat{C})$, but they turned out to be the most conservative with overall coverage rates close to 1 and the longest interval lengths. In particular, for the set $A_0^c \cap A$ the average length of $\xi^*(\widehat{C}_A)$ was much longer than the length of $\xi(\widehat{C}_A)$.

Between the two ways of constructing $\widehat{C}$, $\xi(\widehat{C}_A)$ and $\xi(\widehat{C}_S)$ had the same coverage rates. However, I observe that the average length of $\xi(\widehat{C}_S)$ was longer than that of $\xi(\widehat{C}_A)$, which reduced its power.

There was a constructive suggestion regarding how to construct $\widehat{C}_S$. As described in Section 2.2.3, the original algorithm uses the same dataset to decompose strong and weak signals, and to compute confidence intervals. The suggested way is to use an independent dataset to construct $\widehat{C}_S$ so that it does not use any information of the target dataset. However, consistent to the result in Table 2.1, the methods involving $\widehat{C}_S$ did not outperform those of $\widehat{C}_A$. Therefore, in the following numerical results, I choose to use $\xi(\widehat{C}_A)$ only. See related

discussion in Section 2.2.3. Studying what causes the performance differences in rigorous way is not covered in this dissertation but it can be one of the interesting future works.

2.4.2 Sensitivity to λ

Using the same datasets from Section 2.4.1, I ran more tests to examine how sensitive the coverage of $\xi(\widehat{C}_A)$ is to λ, the tuning parameter of the lasso. I chose 20 λ values, equally spaced between $\|X^\mathsf{T} y\|_\infty/n$ and 0. Figure 2.5 plots the coverage rate and the size of the active set $q = |A|$ against the index i_λ of λ. Note that the λ sequence is in decreasing order so that q increases with i_λ. Each point in the plot is the average of 20 datasets. The coverage rate of $\xi(\widehat{C}_A)$ was preserved around the desired level, indicated by the dashed line in the top panel, when the lasso active set was not extremely small or large. In fact, the coverage rate was well maintained around 95% for $2 \leq q \leq 30$ ($7 \leq i_\lambda \leq 19$), while the size of the true active set $|A_0| = 5$ (the dashed line in the bottom panel). This shows that my method works well for a wide range of models selected by lasso. The coverage rate was a little more sensitive to the choice of λ for the exponential decay design than the other two designs. However, even for that case, when the size of active set $q \geq 3$ ($i_\lambda \geq 10$), the coverage rate stayed around the desired level.

One might worry about the low coverage rates for the first few and the last λ values. However, these λ's are either too small or too big to be chosen in practice. Recall that I choose λ by cross-validation with the one standard error rule, denoted by λ_{1se}. The 5% and 95% percentiles of $i_{\lambda_{1se}}$ were 10.21 and 17.35, between which the performance of my method is seen to be very stable (Figure 2.5). This analysis confirms the notion that my inference tool may be used in conjunction with a data-dependent turning of lasso to quantify the significance of a quite large set of selected features, as discussed in Remark 2.

2.4.3 Comparison on individual intervals

In this section, $\xi(\widehat{C}_A)$ is compared with Lee's method (Lee et al., 2016) implemented in the R package **selectiveInference**. Datasets were simulated in the same way as in Section

Table 2.2: Comparision between (a) $\xi(\widehat{C}_A)$ and (b) Lee's method.

(n,p)	A_0	Σ	Method	Power	Coverage			$\mathbb{P}_\infty$
				$A_0 \cap A$	A	$A_0 \cap A$	$A_0^c \cap A$	
$(100,200)$	$A_0^{(1)}$	I	(a)	0.870	0.968(0.555)	0.956(0.673)	0.975(0.486)	
			(b)	0.783	0.968(1.819)	1.000(0.998)	0.949(2.310)	1.6%
		T	(a)	0.836	0.953(0.590)	0.934(0.721)	0.978(0.416)	
			(b)	0.820	0.944(1.562)	0.951(0.882)	0.935(2.617)	9.3%
		ED	(a)	0.870	0.931(0.501)	0.896(0.606)	0.981(0.349)	
			(b)	0.818	0.946(1.251)	0.922(0.824)	0.981(1.887)	2.3%
		EC	(a)	0.643	0.846(0.612)	0.768(0.760)	0.892(0.523)	
			(b)	0.500	0.973(5.178)	0.964(3.513)	0.978(6.288)	9.4%
	$A_0^{(2)}$	T	(a)	0.911	0.914(0.539)	0.889(0.657)	0.928(0.476)	
			(b)	0.822	0.969(1.618)	0.956(0.994)	0.976(1.984)	4.7%
		ED	(a)	0.889	0.948(0.460)	0.933(0.574)	0.956(0.403)	
			(b)	0.822	0.993(2.733)	0.978(1.301)	1.000(3.457)	3.7%
$(200,400)$	$A_0^{(1)}$	I	(a)	0.923	0.954(0.433)	0.949(0.489)	0.967(0.286)	
			(b)	0.872	0.972(0.728)	0.962(0.376)	1.000(1.645)	0.0%
		T	(a)	0.897	0.933(0.451)	0.926(0.539)	0.946(0.288)	
			(b)	0.868	0.933(1.154)	0.912(0.787)	0.973(1.887)	2.9%
		ED	(a)	0.909	0.928(0.362)	0.896(0.468)	0.968(0.230)	
			(b)	0.779	0.978(1.443)	0.987(0.956)	0.968(2.127)	6.5%
		EC	(a)	0.841	0.926(0.492)	0.921(0.671)	0.929(0.379)	
			(b)	0.714	0.975(3.481)	0.984(2.078)	0.970(4.422)	6.2%
	$A_0^{(2)}$	T	(a)	1.000	0.991(0.441)	1.000(0.512)	0.987(0.407)	
			(b)	0.917	0.928(0.929)	0.972(0.731)	0.907(1.043)	13.5%
		ED	(a)	0.971	0.927(0.383)	0.914(0.441)	0.932(0.355)	
			(b)	0.857	0.972(0.945)	0.943(0.510)	0.986(1.162)	3.7%

The numbers in the parentheses are the average length of the confidence intervals. For Lee's method, only finite intervals are used to compute the average length and $\mathbb{P}_\infty$ is the proportion of excluded infinite intervals.

2.4.1 but with two larger sizes, $(n,p) \in \{(100,200),(200,400)\}$, and one more type of design matrix

- Equicorrelation (EC): $\Sigma_{ii} = 1$ and $\Sigma_{ij} = .7$ $(i \neq j)$.

Note that the correlation among predictors was the highest under this design. I also considered two different ways of placing true active coefficients. In the first case, the true active coefficients were placed together, i.e. $A_0^{(1)} = \{1, \cdots, 5\}$. In the second case, they were evenly spaced out, i.e. $A_0^{(2)} = \{1, p/5+1, \cdots, 4p/5+1\}$. The true active covariates were highly

correlated with each other in the first case, while they were more correlated with other inactive covariates in the second case. See Table 2.2 for the comparison results. Note that the designs Identity and Equicorrelation were considered only with $A_0^{(1)}$, since the two ways of assigning A_0 are equivalent for these two designs.

The coverage rate of Lee's method was well-preserved at the nominal level, $1-\alpha$, in most cases. However, their method sometimes generated very wide or even infinite intervals with ∞ or $-\infty$ as the upper or lower bound. This happens when the conditional distribution $[(X_A^+ y)_j \mid \mathcal{A}(y), (X_A^+ y)_{-j}]$ is truncated to a union of bounded intervals and the observed value of $(X_A^+ y)_j$ is close to one of its boundaries (Kivaranovic and Leeb, 2018). See Section 2.4.5 for a case study that exemplifies this issue. The last column in Table 2.2 reports the proportion of infinite intervals estimated by Lee's method. For example, when $(n, p) = (200, 400)$ and $A_0 = A_0^{(2)}$ with the Toeplitz design, the proportion of infinite confidence intervals was 13.5%. The chance of encountering such an issue was already quite high but it would be even higher if I increased the confidence level.

On the other hand, for most settings, my confidence intervals $\xi(\widehat{C}_A)$ succeed to stay at the desired level while having much shorter average length than the intervals by Lee's method. For every setting except the case of $(n, p, A_0, \Sigma) = (100, 200, A_0^{(1)}, \mathrm{EC})$, my coverage rates averaging over all $j \in A$ were higher than 0.9 and very close to 0.95. The average length of my intervals was uniformly shorter than that of Lee's method (after excluding infinite ones). The difference in the interval length was especially significant for the coefficients in $A \cap A_0^c$ and for the equicorrelation designs. For example, in Table 2.2 when $(n, p) = (200, 400)$ and $A_0 = A_0^{(1)}$ with the equicorrelation design, the average length of $\xi(\widehat{C}_A)$ was 0.492, while the average length from Lee's method was 3.481. This is extremely long considering the fact that I drew β_{0j} from $\mathcal{U}(-1, 1)$ for $j \in A_0$. These long intervals failed to detect significant coefficients and thus resulted in a low power.

In Figures 2.6 and 2.7, I show box-plots of the interval lengths from the 20 datasets in each design for a closer view. Each box-plot reports the interval lengths for all variables in A, $A \cap A_0$ or $A \cap A_0^c$. Consistent with the results from Table 2.2, the interval lengths of Lee's method were much larger than those of my method. In particular, under the equicorrelation

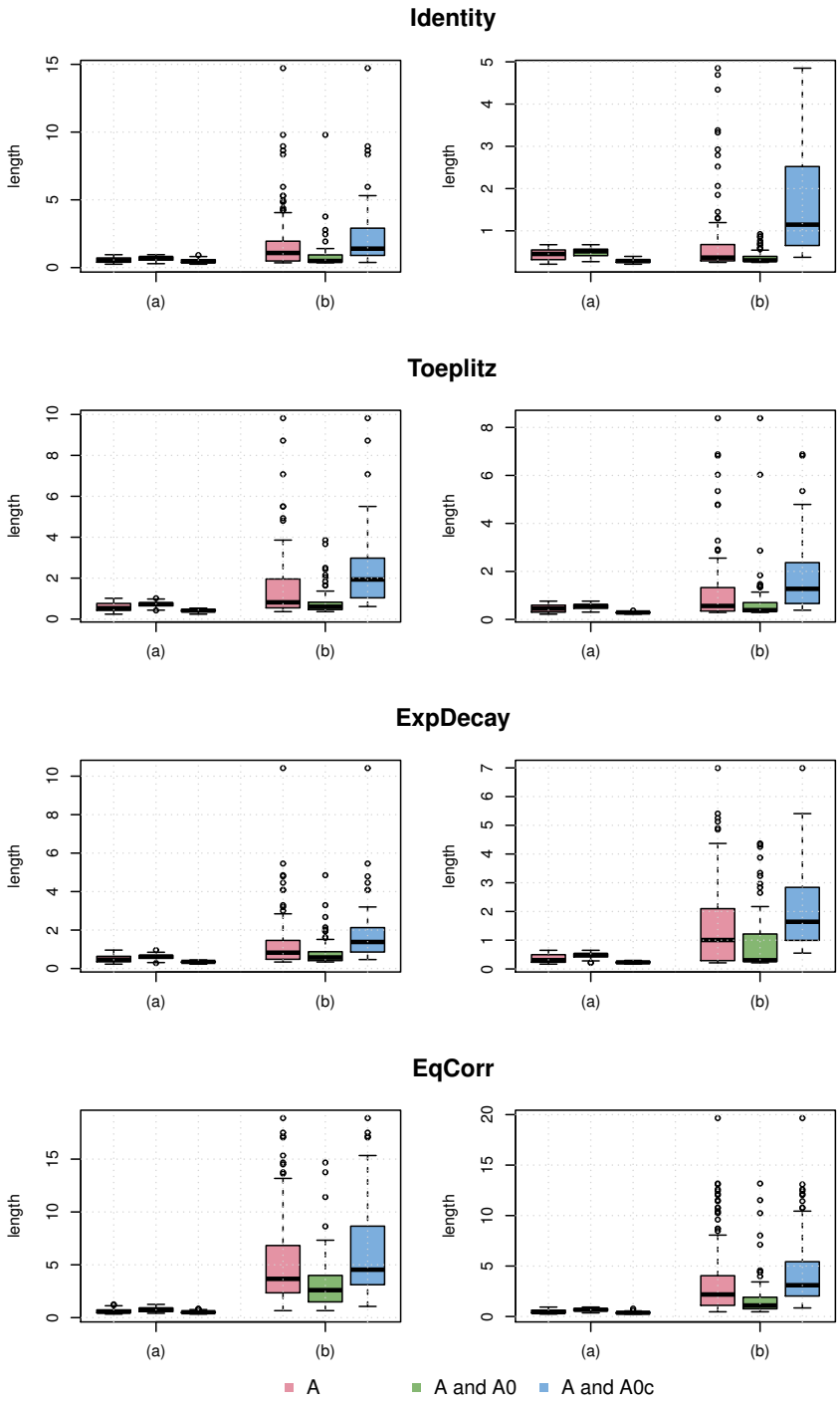

Figure 2.6: Comparison between (a) $\xi(\widehat{C}_A)$ and (b) Lee's method when $A_0 = \{1, \cdots, 5\}$. The left and right columns are for $(n, p) = (100, 200)$ and $(200, 400)$, respectively.

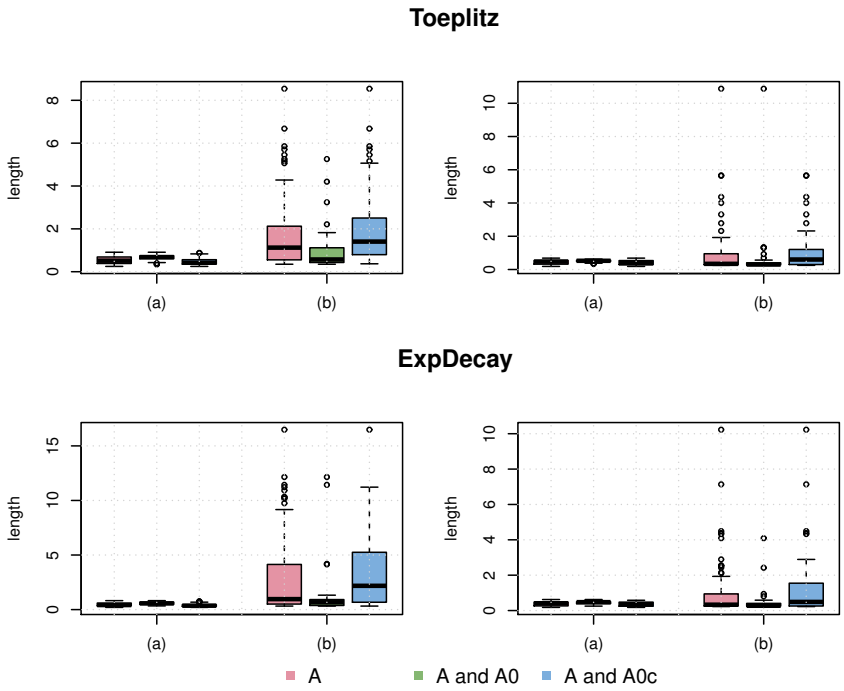

Figure 2.7: The same as Figure 2.6 but for $A_0 = \{1, p/5+1, \cdots, 4p/5+1\}$.

design, the maximum length of my intervals was even smaller than the first-quartile length of Lee's method for all three sets of variables. The length of my intervals is also much less variable than that of Lee's method for every case in the two figures, which shows that my method is more consistent across different datasets. One can easily see that Lee's method produced a number of lengthy intervals, represented as isolated dots or outliers in a box-plot. These intervals are not informative at all. Lastly, the difference between the two methods is most drastic for the set $A \cap A_0^c$, where the intervals from Lee's method can be 10 times longer than ours. Note that by removing all the infinite intervals output by Lee's method from these plots, this comparison favors Lee's method.

2.4.4 Comparison on joint confidence sets

I conducted further experiments to examine the performance of my method in constructing confidence sets for ν. I generated results under three types of design, $\Sigma \in \{\mathrm{T}, \mathrm{ED}, \mathrm{EC}\}$, and two data sizes, $(n, p) \in \{(100, 200), (200, 400)\}$, with A_0 fixed to $\{1, \ldots, 5\}$. Under each of

Table 2.3: Coverage, power, and size of pairwise confidence sets

	Method	$(n,p)=(100,200)$			$(n,p)=(200,400)$			Overall
		T	ED	EC	T	ED	EC	
Coverage	$\xi(\widehat{C};\ell_2)$	0.941	0.971	0.965	0.912	0.912	0.937	0.940
	$\xi(\widehat{C};\ell_\infty)$	0.972	0.984	0.970	0.950	0.962	0.944	0.964
	Lee	0.924	0.973	0.997	0.972	0.989	0.992	0.975
Power	$\xi(\widehat{C};\ell_2)$	1.000	0.983	0.831	1.000	1.000	0.986	0.967
	$\xi(\widehat{C};\ell_\infty)$	0.986	0.975	0.769	1.000	0.992	0.986	0.951
	Lee	0.718	0.558	0.138	0.736	0.558	0.568	0.546
Diameter	$\xi(\widehat{C};\ell_2)$	0.885	0.734	1.045	0.644	0.488	0.792	0.765
	$\xi(\widehat{C};\ell_\infty)$	1.173	0.972	1.384	0.850	0.653	1.065	1.016
	Lee	2.854	2.301	10.221	1.935	3.493	6.251	4.509
Volume	$\xi(\widehat{C};\ell_2)$	0.627	0.431	0.883	0.331	0.193	0.521	0.498
	$\xi(\widehat{C};\ell_\infty)$	0.699	0.481	0.986	0.366	0.220	0.601	0.559
	Lee	4.181	2.822	50.299	1.861	5.818	20.843	14.304
$\mathbb{P}_\infty$	Lee	0.201	0.058	0.229	0.117	0.171	0.147	0.154

Note: For Lee's method, only finite sets are used to compute the average diameter and volume, and $\mathbb{P}_\infty$ reports the proportion of excluded infinite sets.

these six settings, the same 20 datasets as in Section 2.4.3 were used. First, I constructed confidence sets $\xi_{[e_i,e_j]^\intercal}(\widehat{C};\ell_\delta)$ (2.12) for each pair (ν_i,ν_j), $i\neq j$, with ℓ_2 norm and ℓ_∞ norm, i.e. $\delta\in\{2,\infty\}$. Then, I moved to confidence sets $\xi_{\mathbf{I}_q}(\widehat{C};\ell_\delta)$, $q=|A|$, for joint inference on ν, again using the two norms, ℓ_2 and ℓ_∞. Consequently, the confidence sets were either a sphere or a hypercube in $\mathbb{R}^q$. As I am not aware of any method specifically designed for joint inference after lasso selection, I compared my results with Lee's method using multiple test adjustment. To build a $1-\alpha$ confidence set for ν_B, $|B|=d$, individual intervals $\widehat{I}_k$, $k\in B$, were constructed by Lee's method with an adjusted confidence level $1-\alpha/d$, and then a confidence set was constructed as the Cartesian product of $\widehat{I}_k, k\in B$.

The following metrics are used to evaluate constructed confidence sets. Recall A is the set of selected variables and $q=|A|$. For a positive integer m, let $\mathcal{B}(m)=\{B: B\subset\mathbb{N}_q,\ |B|=m\}$ index all size-m subsets of A. I define coverage and power by averaging over sets of variables in $\mathcal{B}(m)$:

$$\text{Coverage}=\textstyle\sum_{B\in\mathcal{B}(m)}\mathbb{P}\Big(\nu_B\in\xi_{e_B^\intercal}(\widehat{C})\Big)/|\mathcal{B}(m)|,$$

Table 2.4: Coverage, power and size of joint confidence sets

	Method	$(n,p)=(100,200)$			$(n,p)=(200,400)$			Overall
		T	ED	EC	T	ED	EC	
Coverage	$\xi_{\mathbf{I}_q}(\widehat{C};\ell_2)$	0.900	1.000	0.950	0.850	0.900	0.950	0.925
	$\xi_{\mathbf{I}_q}(\widehat{C};\ell_\infty)$	1.000	1.000	0.950	1.000	1.000	1.000	0.992
	Lee	0.950	0.950	0.950	0.800	1.000	1.000	0.942
Power	$\xi_{\mathbf{I}_q}(\widehat{C};\ell_2)$	1.000	1.000	0.900	1.000	1.000	1.000	0.983
	$\xi_{\mathbf{I}_q}(\widehat{C};\ell_\infty)$	1.000	1.000	0.650	1.000	1.000	1.000	0.942
	Lee	0.250	0.150	0.050	0.250	0.450	0.050	0.200
Diameter	$\xi_{\mathbf{I}_q}(\widehat{C};\ell_2)$	1.192	1.093	1.509	0.891	0.785	1.324	1.132
	$\xi_{\mathbf{I}_q}(\widehat{C};\ell_\infty)$	2.179	2.080	3.101	1.576	1.500	2.831	2.211
	Lee	3.362	3.602	8.321	2.262	1.188	6.471	4.201
Volume*	$\xi_{\mathbf{I}_q}(\widehat{C};\ell_2)$	0.826	0.719	0.946	0.627	0.530	0.812	0.743
	$\xi_{\mathbf{I}_q}(\widehat{C};\ell_\infty)$	0.958	0.838	1.148	0.707	0.613	1.011	0.879
	Lee	1.013	1.084	3.201	0.699	0.474	1.865	1.389
$\mathbb{P}_\infty$	Lee	0.300	0.150	0.450	0.200	0.300	0.400	0.300

Note: Volume* is the normalized volume. For Lee's method, only finite sets are used to compute the average diameter and volume, and $\mathbb{P}_\infty$ reports the proportion of excluded datasets due to infinite volumes or infinite diameters.

$$\text{Power} = \textstyle\sum_{B\in\mathcal{B}(m)} \mathbb{P}\Big(0 \notin \xi_{e_B^\intercal}(\widehat{C})\Big)/|\mathcal{B}(m)|.$$

For example, when considering $\xi_{\mathbf{I}_q}(\widehat{C})$, $m = q$ and $|\mathcal{B}(q)| = 1$. For pairwise confidence sets, $m = 2$ and $|\mathcal{B}(2)| = q(q-1)/2$. The volume and the diameter of a confidence set were recorded for comparison as well, where the diameter is defined as the maximum distance between any two points in the set. In short, a confidence set has a better performance if it has a higher power and a smaller volume or diameter, while achieving the nominal coverage rate.

Table 2.3 reports the comparison on pairwise confidence sets. The last column reports the overall averages across all simulation settings. While the coverage rates for all confident sets were close to the desirable level, the volume and the diameter of Lee's method were often much larger than my confidence sets. For example, when compared to $\xi(\widehat{C};\ell_2)$ under $(n,p,\Sigma) = (100,200,\text{EC})$, the average diameter and the average volume of Lee's method were 10 and 55 times larger, respectively. To compare the power, I restricted to those pairs (i,j) for which both variables X_i and X_j were in the true support A_0, i.e. $i,j \in A_0 \cap A$. As

my method built smaller confidence sets, it is not surprising to see that my confidence sets showed a much higher power for all data settings.

Table 2.4 reports the results of joint confidence sets for ν, i.e. $H = \mathbf{I}_q$ in (2.12). Since an average volume can be highly influenced by a few datasets with large active sets, i.e. large $|A|$, I normalized each volume by the size of A to calculate Volume$^* =$ Volume$^{1/|A|}$ before averaging over datasets. As seen from the table, while the coverage rates of $\xi_{\mathbf{I}_q}(\widehat{C}; \ell_2)$, $\xi_{\mathbf{I}_q}(\widehat{C}; \ell_\infty)$ and Lee's method all stayed around the desirable level, the average diameter and the average volume of Lee's method were larger than ours. In particular, for the equicorrelation designs (EC), the average diameter and normalized volume of Lee's method were, respectively, more than 4 and 2 times larger than those of $\xi_{\mathbf{I}_q}(\widehat{C}; \ell_2)$. When $(n, p, \Sigma) = (200, 400, \text{ED})$, I observe that the average volume of Lee's method was smaller than that of ours. This is because I only used datasets for which Lee's method did not produce any infinite intervals when computing the diameters and volumes for their method, which clearly underestimated the average size of their confidence sets. More extreme differences are seen when comparing the power of a confidence set. While the average power of my method was close to one for most cases, the average power of Lee's method was only around 0.20, which demonstrates the advantage of my method. The issue of producing infinite confidence sets by Lee's method was even more severe for joint inference, as expected. For $(n, p, \Sigma) = (100, 200, \text{EC})$, their method generated infinite intervals for almost half of the datasets, which would be problematic in practical applications.

Overall, my confidence sets were able to achieve the nominal coverage rate with a high power and a small diameter. my current implementation constructs either a sphere or a hypercube centered at $H\hat{\nu}$ as a confidence set. One may propose alternative ways to build a confidence set of other shapes using the samples of y^* generated by my MCMC algorithm. One option is to approximate the contours of $[HX_A^+ y^* \mid \mathcal{A}(y^*) = A]$ (cf. (2.11)) to build a confidence set, in the similar spirit of a highest posterior density region. I leave this appealing possibility to future work.

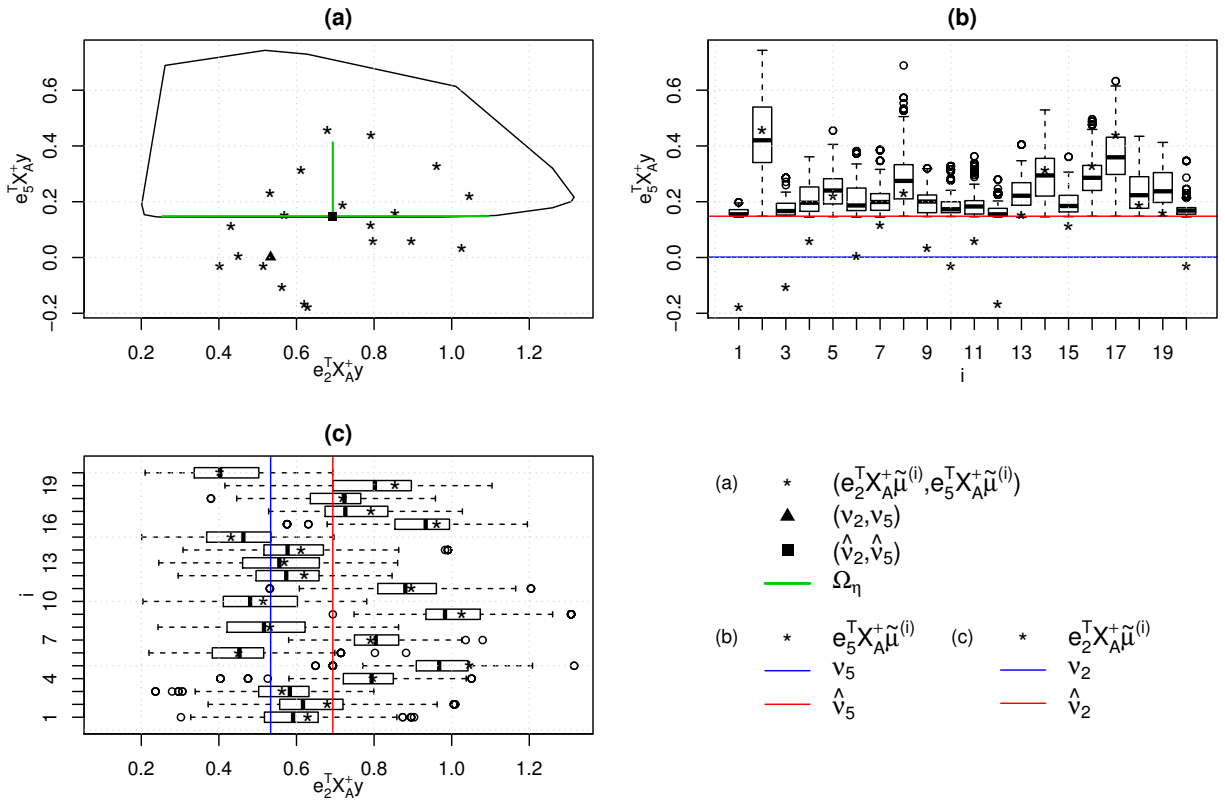

Figure 2.8: The difference between my and Lee's methods. (a) The feasible regions, (b) box-plots of $e_5^\mathsf{T} X_A^+ y^*$, and (c) box-plots of $e_2^\mathsf{T} X_A^+ y^*$.

2.4.5 A case study

Both my method and Lee's method are based on the truncated Gaussian distribution of y given $\mathcal{A}(y) = A$, but for some data Lee's intervals turned out to be much wider in the above comparisons. To clarify the key differences between the two methods at a conceptual level, I took a closer look at one dataset from the simulation setting $(n, p, \Sigma) = (100, 200, \mathrm{T})$ in Table 2.2, for which the lasso support included seven variables, i.e. $|A| = 7$. Here, I focus on making inference about (ν_2, ν_5), whose true value was $(0.533, 0.001)$. The corresponding observed value was $(\hat{\nu}_2, \hat{\nu}_5) = (\eta_2^\mathsf{T} y, \eta_5^\mathsf{T} y) = (0.694, 0.148)$, where $\eta_j = (X_A^+)^\mathsf{T} e_j$. My intervals for the two parameters were $\xi_2(\widehat{C}) = [0.266, 1.126]$ and $\xi_5(\widehat{C}) = [-0.263, 0.150]$, respectively, while Lee's intervals $\widehat{I}_2 = [0.469, 0.918]$ and $\widehat{I}_5 = [-11.398, 0.413]$. While all four intervals cover the true parameters, $\widehat{I}_5$ is extremely wide compared to $\xi_5(\widehat{C})$.

Let us walk through my procedure to construct $\xi_j(\widehat{C})$ in this example. Given an uncon-

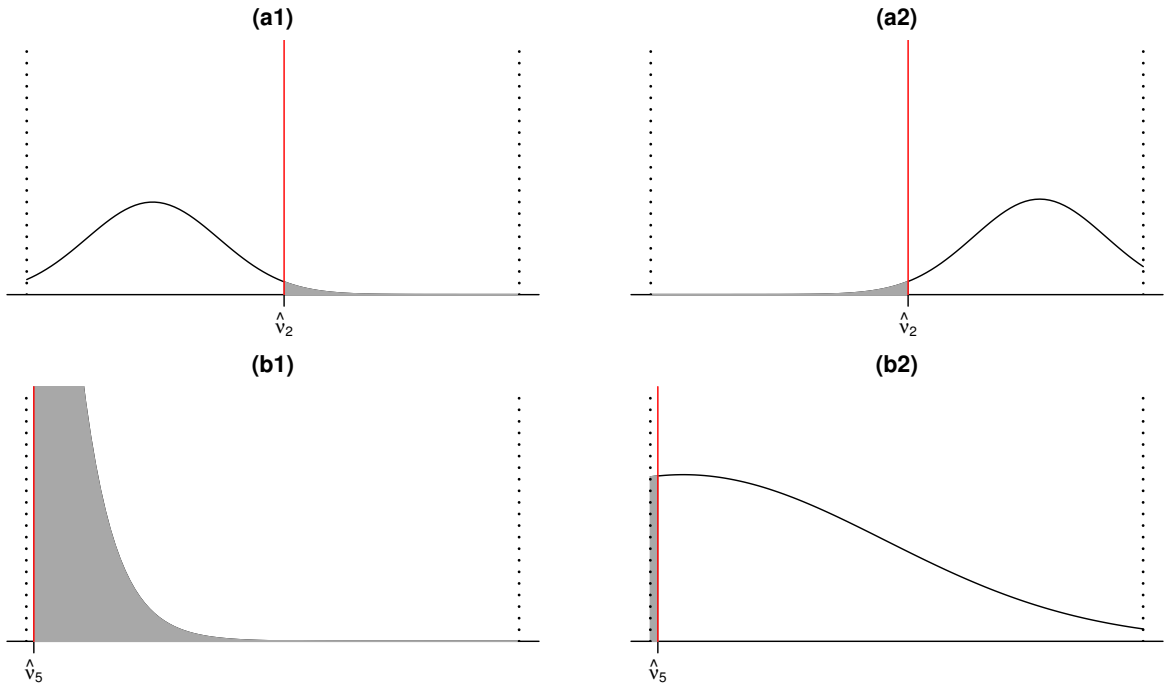

Figure 2.9: Illustration of deriving confidence intervals using Lee's method. The dotted lines in panel (a) and (b) represent $\Omega_\eta(2) = [0.254, 1.095]$ and $\Omega_\eta(5) = [0.144, 0.413]$, respectively.

ditional confidence set $\widehat{C}$, I first draw $\tilde{\mu}^{(i)}$ uniformly over $\mathcal{U}(\widehat{C})$, which are shown as the gray dots in Figure 2.8(a) after being projected to η_2 and η_5. For each $\tilde{\mu}^{(i)}$, I simulate a sample of y^* from the conditional distribution $[y^* \mid \mathcal{A}(y^*) = A]$, i.e. $\mathcal{N}_n(\tilde{\mu}^{(i)}, \sigma^2\mathbf{I}_n)$ truncated to the union of many polyhedra, say $\mathcal{D}$, whose projection $(\eta_2^\mathsf{T}\mathcal{D}, \eta_5^\mathsf{T}\mathcal{D})$ is illustrated by the solid-line polygon in Figure 2.8(a). (The exact polygon may differ slightly as I am just plotting an approximate one for easy illustration.) The histograms of the simulated $\eta_j^\mathsf{T} y^*$ are shown as box-plots in Figure 2.8(b) and (c) for $j = 2, 5$. Each box-plot corresponds to the distribution of $\eta_j^\mathsf{T} y^*$ given one $\tilde{\mu}^{(i)}$ for $i = 1, \ldots, K$. I then construct confidence intervals or sets from the aggregation of these samples across all i.

To construct the interval $\widehat{I}_j$ for ν_j, Lee et al. (2016) decompose y into $\eta_j^\mathsf{T} y$ and its orthogonal component $z_{-j} := (\mathbf{I}_n - P_{\eta_j})y$, the residual after projecting to η_j. Their inference is then based on the conditional distribution

$$\eta_j^\mathsf{T} y \mid \{\mathcal{A}(y) = A, z_{-j}\} \sim \mathcal{TN}(\nu_j, \sigma^2\|\eta_j\|^2, \Omega_\eta(j)),$$

where the truncation interval $\Omega_\eta(j)$ depends on the observed value z^{o}_{-j} of z_{-j}. The green line segments in Figure 2.8(a) show $\Omega_\eta(j)$, the intersection between $\mathcal{D}$ and the line $\{y : z_{-j} = z^{\text{o}}_{-j}\}$, which is a bounded one-dimensional interval for each j. They are much more restrictive than the feasible set of my samples y^*, projected to these two dimensions, shown as the solid-line polygon. This is the first key difference between the two methods. Then Lee's method finds all possible values of $\nu_j \in \mathbb{R}$ that makes the observed statistic $\hat{\nu}_j$ within $(\alpha/2, 1-\alpha/2)$-quantiles to define $\widehat{I}_j$. When $\hat{\nu}_j$ is close to the boundary of $\Omega_\eta(j)$, which is the case for $\hat{\nu}_5$, their method tends to generate very wide intervals, such as $\widehat{I}_5 = [-11.398, 0.413]$. See Figure 2.9 for the illustration. The lower bound of $\widehat{I}_2$, 0.469, is the center of the truncated normal distribution in panel (a1) which makes the dark area to be $1-\alpha/2$ while the upper bound of $\widehat{I}_2$, 0.918, is the center of the truncated normal distribution in panel (a2). Same logic is applied to panel (b1) and (b2) and that is how I got $\widehat{I}_5 = [-11.398, 0.413]$. I deliberately set the center of the distribution in panel (b1) to -0.5 in order to make the curve visible. The actual curve is hard to visualize since the density is too away from its center.

My method gets around this issue with truncated Gaussian inference by considering a smaller range of ν_j represented by the samples of $e_j^\mathsf{T} X_A^+ \tilde{\mu}$, i.e. the star dots ($\star$) in panel (a), which is the second key difference. This greatly shortens the constructed intervals, such as $\xi_5(\widehat{C}) = [-0.263, 0.150]$. See Figure 2.8(b) for illustration. On the other hand, when $\hat{\nu}_j$ is far away from the boundaries of $\Omega_\eta(j)$, as for $j = 2$ in this example, Lee's method builds efficient intervals, while ours can be slightly wider due to the additional randomness in $\tilde{\mu}$ (Figure 2.8c).

2.5 Discussion

I have proposed a new method for post-selection inference, based on estimator augmentation and a conditional MCMC sampler. Estimator augmentation is applied to derive a closed-form density for the conditional distribution $[\hat{\beta}_A, S_I \mid \mathcal{A}(y) = A]$, which is then used as the target distribution in my MCMC sampler. I randomize the estimate of the mean μ_0 by uniform sampling over a confidence set, which incorporates the uncertainty in using a

plug-in estimate of μ_0 in my sampling procedure. I have shown with numerical comparisons that my method constructs much shorter confidence intervals than Lee's method (Lee et al., 2016), while achieving a comparable coverage rate. Moreover, unlike their method, my method never produces any infinite confidence intervals. With its great flexibility, I further demonstrated that my method can perform joint inference by constructing confidence sets for any set of parameters of interest after lasso selection, which is a unique contribution of this work.

While I have focused on the lasso active set in this work, conditioning on more general events is possible under my framework for post-selection inference. Recall that I parameterize the augmented estimator $(\hat{\beta}, S)$ by the triplet $(\hat{\beta}_{\mathcal{A}}, S_{\mathcal{I}}, \mathcal{A})$. Suppose the selected model is defined by the event $F(\hat{\beta}_{\mathcal{A}}, S_{\mathcal{I}}, \mathcal{A}) \in \mathcal{E}$, where F is a mapping. Similar to Corollary 5, I may obtain the density for the conditional distribution of the augmented estimator given the event $F(\hat{\beta}_{\mathcal{A}}, S_{\mathcal{I}}, \mathcal{A}) \in \mathcal{E}$ based on Theorem 1. Let $I_{\mathcal{E}}(v)$ be the indicator function for $\{v \in \mathcal{E}\}$.

Corollary 7. *Under the same assumptions of Theorem 1, the conditional distribution* $[\hat{\beta}_{\mathcal{A}}, S_{\mathcal{I}}, \mathcal{A} \mid F(\hat{\beta}_{\mathcal{A}}, S_{\mathcal{I}}, \mathcal{A}) \in \mathcal{E}]$ *is given by*

$$\mathbb{P}_{\Theta, \mathcal{A} \mid F \in \mathcal{E}}(d\theta, A) \propto f_R\big(V_R^{\mathsf{T}} H(\theta, A; \mu_0, \lambda); \sigma^2\big) |\det T(A; \lambda)| I_{\mathcal{E}}(F(\theta, A)) d\theta, \tag{2.21}$$

where $(\theta, A) = (b_A, s_I, A)$ *satisfying the constraints in* (1.7) *and* (1.8).

Compared to the joint distribution (1.9), the only difference is the inclusion of the indicator function, which essentially imposes more constraints on (θ, A) in addition to (1.7) and (1.8). A key step in the development of Monte Carlo algorithms for the above more general conditional distribution is to design efficient proposals that move (θ, A) in its feasible region satisfying all the imposed constraints. This is a challenging and interesting future direction. As a concrete example, suppose I select variables by thresholding lasso $\hat{\beta}_j$ for $j \in \mathbb{N}_p$, that is, the selected model is $M = \{j : |\hat{\beta}_j| \geq \tau\}$. Then the conditioning event in this example

can be written as

$$\left\{|\hat{\beta}_j| \geq \tau, \forall\, j \in M\right\} \bigcap \left\{|\hat{\beta}_j| < \tau, \forall\, j \notin M\right\}.$$

Note that the active set $\mathcal{A}$ is no longer fixed on this event, although it must satisfy $\mathcal{A} \supset M$. As a result, the target distribution (2.21) is a joint distribution for Θ and $\mathcal{A}$, which will incur additional computational cost to my MH sampler. In particular, the dimension of Θ will change when the size of $\mathcal{A}$ changes. A normal distribution centered at $b_j^{(t)}$ and truncated to $(-\infty, -\tau] \cap [\tau, \infty)$ can be used as a proposal for $b_j^{\dagger}$, $j \in M$ at the t-th iteration. For $j \notin M$, I need to consider some of the active $b_j^{(t)} \in (-\tau, \tau)$ turning into zero and vice versa, which would change the active set $\mathcal{A}$.

Post-selection inference on data with group structure using estimator augmentation for group lasso can be another interesting future topic. Due to the complex sample space of the augmented group lasso estimator (Zhou and Min, 2017a), developing an MCMC sampler is more complicated than the one for lasso. However, this potential generalization will be important for applications with pre-grouped variables, categorical variables, or highly correlated predictors.

2.6 Proofs

Proof of Proposition 4. From (2.6) and (2.7), I have $\xi_j(\mu_0, \alpha/2) \subset \xi_j^*(\widehat{C})$ if $\mu_0 \in \widehat{C}$. Then,

$$\begin{aligned}
& \mathbb{P}\left\{\nu_j \in \xi_j^*(\widehat{C}) \middle| \mathcal{A}(y) = A\right\} \\
\geq\ & \mathbb{P}\left\{\nu_j \in \xi_j^*(\widehat{C}) \middle| \mathcal{A}(y) = A, \mu_0 \in \widehat{C}\right\} \mathbb{P}\big(\mu_0 \in \widehat{C} \mid \mathcal{A}(y) = A\big) \\
\geq\ & \mathbb{P}\left\{\nu_j \in \xi_j(\mu_0, \alpha/2) \middle| \mathcal{A}(y) = A, \mu_0 \in \widehat{C}\right\} \mathbb{P}\big(\mu_0 \in \widehat{C} \mid \mathcal{A}(y) = A\big).
\end{aligned}$$

Due to the independence between $\widehat{C}$ and y,

$$\mathbb{P}\left\{\nu_j \in \xi_j(\mu_0, \alpha/2)\middle|\mathcal{A}(y) = A, \mu_0 \in \widehat{C}\right\} = \mathbb{P}\left\{\nu_j \in \xi_j(\mu_0, \alpha/2)\middle|\mathcal{A}(y) = A\right\} = 1 - \alpha/2,$$

and $\mathbb{P}\big(\mu_0 \in \widehat{C} \mid \mathcal{A}(y) = A\big) = \mathbb{P}(\mu_0 \in \widehat{C}) = 1 - \alpha/2$, which imply (2.8). □

Proof of Theorem 6. Under the assumptions of Theorem 1, for every $y \in \mathbb{R}^n$ there is a unique $(\hat{\beta}, S)$ (Lemma 1 in Zhou (2014)). Therefore, the KKT condition (1.3) establishes a bijection between y and the augmented estimator $(\hat{\beta}, S)$. Consequently, y can be uniquely represented by

$$\begin{aligned} y &= (X^\mathsf{T})^+(X^\mathsf{T}X\hat{\beta} + n\lambda WS) \\ &= X_A\hat{\beta}_A + n\lambda(X^\mathsf{T})^+\big\{W_A \operatorname{sgn}(\hat{\beta}_A) + W_I S_I\big\}, \end{aligned} \tag{2.22}$$

where $(X^\mathsf{T})^+ = (XX^\mathsf{T})^{-1}X$ because X has full row rank. Since $|A| \le n$ (Remark 1) and every n columns of X are linearly independent, I have $X_A^+ = (X_A^\mathsf{T}X_A)^{-1}X_A^\mathsf{T}$. From (1.3),

$$X_A^\mathsf{T}y = X_A^\mathsf{T}X_A\hat{\beta}_A + n\lambda W_{AA}\operatorname{sgn}(\hat{\beta}_A).$$

Multiplying both sides by $(X_A^\mathsf{T}X_A)^{-1}$, I get

$$X_A^+y = \hat{\beta}_A + n\lambda(X_A^\mathsf{T}X_A)^{-1}W_{AA}\operatorname{sgn}(\hat{\beta}_A). \tag{2.23}$$

Then the conclusions (2.14) and (2.15) follow from (2.22) and (2.23) due to the assumption that $[\hat{\beta}_A^*, S_I^*] = [\hat{\beta}_A, S_I \mid \mathcal{A} = A]$. □

CHAPTER 3

Group Inference

3.1 Introduction

When sparse estimation is the main interest, it has been shown that grouping variables may improve the performance in many aspects. First, when individual variable signals are not strong enough to detect, grouping them may lead them to be identifiable (Zhou and Min, 2017b). Second, it can be more relevant when dealing with highly correlated variables (Huang and Zhang, 2010; Meinshausen, 2015). When some of the variables are highly correlated, it is not easy to study the estimator since individual variables may not be detected as significant even if they are. However, by grouping correlated variables together, it can alleviate such an issue. Lastly, there exists some data which are more relevant to apply group-level analysis. For example, for the gene data, genes can be categorized into certain groups by using gene pathways. For categorical variables, when it is expressed through a set of indicator variables, it makes more sense to group them together and test the significance of the group not individual ones. In such cases, exploiting the group structure will perform better than performing ordinary Lasso since additional information is used. This motivates us to study group inference.

In this chapter, I first present some numerical results that support the advantages of grouping effect and then introduce some applications in high-dimensional inference using estimator augmentation in group lasso which was introduced in Section 1.2.2.

3.2 Uncertainty quantification under group sparsity

3.2.1 Methods and simulated data

Suppose the underlying model is

$$y = X\beta_0 + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_n),$$

with the group structure $\mathcal{G} = \{\mathcal{G}_j\}_{1:J}$. Zhou and Min (2017b) propose a parametric bootstrap sampler for a group lasso given $\tilde{\beta}$ and $\hat{\sigma}$, point estimates of β_0 and σ. Let $B(y; \lambda)$ denote the set of minimizers of the loss in (1.1) so that $\hat{\beta} \in B(y; \lambda)$. The parametric bootstrap for the group lasso contains two steps. Denote the parametric bootstrap by $PB(\tilde{\beta}, \hat{\sigma}^2, \lambda)$ in order to emphasize the dependency on $(\tilde{\beta}, \hat{\sigma}^2, \lambda)$:

Algorithm 4 ($PB(\tilde{\beta}, \hat{\sigma}^2, \lambda)$)**.** Given $(\tilde{\beta}, \hat{\sigma}^2, \lambda)$,

(1) draw $\varepsilon^* \sim \mathcal{N}_n(0, \hat{\sigma}^2 \mathbf{I}_n)$ and set $y^* = X\tilde{\beta} + \varepsilon^*$;

(2) solve (1.1) with y^* in place of y to obtain $\hat{\beta}^* \in B(y^*; \lambda)$.

After drawing a large sample of $\hat{\beta}^*$ values with above procedures, inference for each group $j \in \mathbb{N}_J$ can be made. By default, choose the function

$$f_j(\hat{\beta}^*_{(j)} - \tilde{\beta}_{(j)}) = \|X_{(j)}(\hat{\beta}^*_{(j)} - \tilde{\beta}_{(j)})\|^2 \tag{3.1}$$

to build a confidence region and carry out a significance test for $\beta_{0(j)}$. From the bootstrap sample of $\hat{\beta}^*$, estimate the $(1-\gamma)$-quantile $f_{j,(1-\gamma)}$ such that

$$\mathbb{P}\left\{ f_j(\hat{\beta}^*_{(j)} - \tilde{\beta}_{(j)}) > f_{j,(1-\gamma)} \middle| X, \tilde{\beta}, \hat{\sigma} \right\} = \gamma.$$

Then the $(1-\gamma)$ confidence region for $\beta_{0(j)}$ becomes

$$R_j(\gamma) = \left\{ \theta \in \mathbb{R}^{p_j} : f_j(\hat{\beta}_{(j)} - \theta) \leq f_{j,(1-\gamma)} \right\}. \tag{3.2}$$

One may also test the hypothesis $H_{0,j} : \beta_{0(j)} = 0$, which will be rejected at level γ if $0 \notin R_j(\gamma)$, that is, if $\|X_{(j)}\hat{\beta}_{(j)}\|^2 > f_{j,(1-\gamma)}$. This approach can make inference about $\beta_{0(j)}$ simultaneously for $j \in \mathbb{N}_J$.

The remaining question is how to construct $\tilde{\beta}$ and $\hat{\sigma}^2$. One possible way to construct $\tilde{\beta}$ is to threshold the group lasso $\hat{\beta}$,

$$\tilde{\beta}_{(j)} = \hat{\beta}_{(j)} I(\|\hat{\beta}_{(j)}\| > b_{\text{th}}), \quad j \in \mathbb{N}_J, \tag{3.3}$$

where $b_{\text{th}} > 0$ is a cutoff value. Useful practical guidance is to choose the cutoff so that all small coefficient groups will be thresholded to zero. Let $M = G(\tilde{\beta})$ be the active groups of $\tilde{\beta}$ and $A = \mathcal{G}_M$ be the set of active coefficients of $\tilde{\beta}$. Then I perform a least-squares regression of y on X_A to re-calculate the nonzero coefficients of $\tilde{\beta}$, which reduces their bias, and to estimate the error variance $\hat{\sigma}^2$, provided that $|A| < n$.

To evaluate its finite-sample performance, the parametric bootstrap method is applied on simulated and real data sets. For each data set, a solution path of the group lasso is obtained using the `R` package `grpreg` (Breheny and Huang, 2015) and the tuning parameter λ is chosen by cross-validation. Let $\hat{\beta} \in B(y; \hat{\lambda})$ denote the solution for the chosen $\hat{\lambda}$ and $s = |G(\hat{\beta})|$ be the number of active groups of $\hat{\beta}$. In light of equation (3.15) in Zhou and Min (2017b), set the threshold value $b_{\text{th}} = \frac{1}{2}\hat{\lambda}(sp_{\max})^{1/2}$ to obtain $\tilde{\beta}$. When $\tilde{\beta}$ has n or more nonzero components, only its largest $\lfloor n/p_{\max} \rfloor - 1$ groups in terms of ℓ_2 norm are kept. Then the active coefficients of $\tilde{\beta}$ are re-computed via least-squares, and the noise variance is estimated by the residual. Given $(\tilde{\beta}, \hat{\sigma})$, $N = 300$ bootstrap samples of $\hat{\beta}^* \in B(y^*; \hat{\lambda})$ are drawn to make inference.

My bootstrap method is compared with competitors, including two methods of the de-biased lasso approach (Javanmard and Montanari, 2014; van de Geer et al., 2014) which was reviewed in Section 1.3, and the group-bound method (Meinshausen, 2015), implemented in the `R` package `hdi` (Dezeure et al., 2015) and `R` function `SSlasso`. To distinguish from the de-biased lasso method of Javanmard and Montanari (2014), I will call the method of van de Geer et al. (2014) the de-sparsified lasso. The group-bound method generates one

sided confidence intervals for ℓ_1 norm of each group. It starts from constructing a convex set $\mathcal{C}_\alpha$ such that $\mathbb{P}(\varepsilon \in \mathcal{C}_\alpha) \geq 1 - \alpha$. From there, the lower bound of one-sided confidence interval for each j-th group is computed. If the lower bound is positive, the group-bound identifies the corresponding group to be significant.

The `hdi` package allows the user to input an estimate of the noise variance for the de-sparsified lasso, for which I use the same estimate in my approach to make results more comparable. Other tuning parameters are chosen via the default methods in their respective implementation.

The rows of X are independently drawn from $\mathcal{N}_p(0, \Sigma)$, with Σ chosen from the following two designs: (i) Toeplitz, $\Sigma_{jk} = 0.5^{|j-k|}$; (ii) Exponential decay, $(\Sigma^{-1})_{jk} = 0.4^{|j-k|}$. Recall that q_0 denotes the number of active coefficients. I adopt two distinct ways to assign active coefficients: (1) Set the first q_0 coefficients, β_{0k}, $k = 1, \ldots, q_0$, to be nonzero; (2) the active coefficients are evenly spaced in $\mathbb{N}_p$. Since neighboring X_j's are highly correlated in both designs, the two different ways of assigning active coefficients lead to distinct correlation patterns among the true predictors and between the true and false predictors. Index the two designs by $d \in \{\text{i, ii}\}$ and the two ways of assigning active coefficients by $a \in \{1, 2\}$. Given X and β_0, the response y is simulated from $\mathcal{N}_n(X\beta_0, \mathbf{I}_n)$. I fixed $q_0 = 10$ and drew β_{0k} from $\mathcal{U}(-1, 1)$ for $k \in A_0$. I chose $(n, p) \in \{(100, 200), (100, 400)\}$. The combination of above choices (a, d, n, p) created eight different data generation settings. In each setting, I generated $K = 20$ data sets, i.e. K independent realizations of (y, X, β_0).

3.2.2 Group inference

Let $M_0 = G(\beta_0)$, $s_0 = |M_0|$ and $q_0 = |\mathcal{G}_{M_0}|$ so that s_0 is the number of active groups and q_0 the number of active coefficients of β_0. I first examine the performance in group inference. The predictors were partitioned into groups of size $p_j = 10$ by two different methods. In the first method, I group the 10 active coefficients into one group and the other zero coefficients into the remaining groups, in which case there is only one active group. In the second way of grouping, there are two active groups, each containing five nonzero coefficients and five

zero coefficients. I will denote these two ways of grouping by $\mathcal{P}_1$ and $\mathcal{P}_2$. Clearly, the signal strength of the active groups in $\mathcal{P}_2$ is weaker.

My bootstrap method, the de-sparsified lasso, and the group-bound method were used to test the hypothesis $H_{0,j} : \beta_{0(j)} = 0$ for each group. The de-sparsified lasso method outputs a p-value ψ_k for each individual test $\beta_{0k} = 0$ for $k \in \mathbb{N}_p$. If $k \in \mathcal{G}_j$, I adjust the p-value by Bonferroni correction with the group size to obtain $\psi_{\text{adj},k} = \psi_k p_j$. Then the hypothesis $H_{0,j}$ will be rejected at level α if $\min_{k\in\mathcal{G}_j} \psi_{\text{adj},k} \leq \alpha$. For each $j \in \mathbb{N}_J$, the group-bound method constructs a lower bound for $\|\beta_{0(j)}\|_1$ to test the hypothesis $H_{0,j}$ at level γ. I chose $\alpha = 0.05$ and recorded the numbers of rejections among the active and the zero groups, denoted by m_M and m_{M^c}, respectively. Then for each method, I calculated the power PWR $= m_M/s_0$ and the type-I error rate, i.e. false positive rate, FPR $= m_{M^c}/(J - s_0)$. My method can build a confidence region for each group (3.5), and I recorded the coverage rate r_M for the active groups. Note that the coverage rate for the zero groups $r_{M^c} = 1 - \text{FPR}$. The other two competing methods do not construct confidence regions for a group of coefficients. The average result over the K data sets in each data generation setting is reported in Table 3.1.

The big picture of this comparison is very clear. my bootstrap method shows a very satisfactory control of type-I errors, around the nominal level of 5% for all cases, while its coverage rate for active groups is > 0.9 with power > 0.8 for a strong majority. In contrast, the de-sparsified lasso method is too optimistic with very high type-I error rates, ranging between 40% and 70%, and the group-bound approach is extremely conservative, resulting in no false rejections but having little power at all. Although the bias term Δ (1.20) can be far from negligible when n is finite, the de-sparsified lasso method totally ignores this term, and as a result its confidence intervals are often too narrow and its p-values become severely underestimated. On the contrary, my approach takes care of the bias in the group lasso via simulation instead of asymptotic approximation, which turns out to be very important for finite samples as suggested by the comparison. This is one of the reasons for the observed better performance of my method in Table 3.1. Another reason is my explicit use of the group lasso so that group structures are utilized in both the estimation of $\tilde{\beta}$ and the bootstrap simulation. These points will be further confirmed in my subsequent

Table 3.1: Coverage, power and false positive rate (%) in group inference

Data Setting			bootstrap			de-sparsified		group-bound	
(n,p)	(a,d)	$\mathcal{G}$	r_M	PWR	FPR	PWR	FPR	PWR	FPR
$(100,200)$	(1, i)	$\mathcal{P}_1$	95.0	95.0	5.5	100.0	44.2	0.0	0.0
		$\mathcal{P}_2$	92.5	67.5	5.3	97.5	48.9	0.0	0.0
	(1, ii)	$\mathcal{P}_1$	95.0	100.0	5.0	100.0	50.3	0.0	0.0
		$\mathcal{P}_2$	90.0	97.5	3.6	100.0	54.4	0.0	0.0
	(2, i)	$\mathcal{P}_1$	100.0	100.0	5.3	100.0	53.4	0.0	0.0
		$\mathcal{P}_2$	90.0	85.0	4.7	100.0	52.5	0.0	0.0
	(2, ii)	$\mathcal{P}_1$	100.0	100.0	4.2	100.0	61.8	0.0	0.0
		$\mathcal{P}_2$	100.0	95.0	4.7	100.0	61.7	0.0	0.0
$(100,400)$	(1, i)	$\mathcal{P}_1$	95.0	100.0	4.9	100.0	42.6	0.0	0.0
		$\mathcal{P}_2$	85.0	75.0	3.2	100.0	45.4	0.0	0.0
	(1, ii)	$\mathcal{P}_1$	90.0	100.0	4.4	100.0	64.2	0.0	0.0
		$\mathcal{P}_2$	85.0	87.5	6.2	97.5	61.4	0.0	0.0
	(2, i)	$\mathcal{P}_1$	90.0	100.0	4.0	100.0	65.8	0.0	0.0
		$\mathcal{P}_2$	85.0	67.5	4.3	100.0	64.7	0.0	0.0
	(2, ii)	$\mathcal{P}_1$	100.0	100.0	4.0	100.0	74.6	0.0	0.0
		$\mathcal{P}_2$	82.5	90.0	3.0	100.0	66.6	0.0	0.0

r_M: coverage rate of active groups; PWR: power; FPR: false positive rate.

comparison on inference for individual coefficients.

The group-bound method is by nature a conservative approach, testing the null hypothesis $\beta_{0G} = 0$ for $G \subset \mathbb{N}_p$ with a lower-bound of the ℓ_1 norm $\|\beta_{0G}\|_1$. By design, the type-I error is controlled simultaneously for all groups $G \subset \mathbb{N}_p$ at the significance level γ even if one specifies a particular group, like what I did in this comparison. It suffers from low power, especially when the group G does not include all the covariates that are highly correlated with the true variables. To verify this observation, I did more test on the data sets of size $(n,p) = (100,200)$. For the Toeplitz design with the first 10 coefficients being active, its power stayed close to zero until I included the first 100 variables in the group G and increased to 0.86 when $G = \{1,\ldots,p\}$ including all variables. I then increased the signal strength by simulating active coefficients $\beta_{0k} \sim \mathcal{U}(-3,3)$. In this case, the power of the group-bound method increased to 0.45 for $G = \{1,\ldots,10\}$ and to 0.52 for $G = \{1,\ldots,50\}$, which are still substantially lower than the power of my bootstrap method under weaker signals; see the first row in Table 3.1. This numerical comparison demonstrates the advantage of my

Table 3.2: Power and confidence intervals for inference on individual coefficients

Data Setting (n,p)	(a,d)	Method	PWR	r_M	L_M	r_{M^c}	L_{M^c}	r	L
$(100,200)$	(1, i)	bootstrap	54.0	43.5	0.456	96.7	0.136	94.0	0.152
		de-sparsified	67.5	76.5	0.540	89.3	0.542	88.6	0.542
		de-biased	60.0	81.0	0.566	98.8	0.568	97.9	0.568
	(1, ii)	bootstrap	61.0	56.5	0.414	96.4	0.110	94.4	0.127
		de-sparsified	74.5	74.5	0.434	86.1	0.433	85.6	0.433
		de-biased	68.0	83.0	0.443	98.8	0.441	98.1	0.441
	(2, i)	bootstrap	60.5	54.5	0.426	96.2	0.131	94.1	0.146
		de-sparsified	70.0	79.0	0.498	85.1	0.503	84.8	0.503
		de-biased	60.0	95.0	0.519	98.9	0.525	98.7	0.525
	(2, ii)	bootstrap	72.0	60.0	0.408	96.2	0.138	94.4	0.152
		de-sparsified	79.5	85.0	0.422	82.8	0.420	82.9	0.420
		de-biased	72.0	96.0	0.443	99.1	0.441	99.0	0.442
$(100,400)$	(1, i)	bootstrap	47.0	35.5	0.335	96.2	0.065	94.7	0.071
		de-sparsified	65.0	74.0	0.537	85.0	0.539	84.8	0.539
		de-biased	53.0	73.5	0.523	99.1	0.526	98.5	0.526
	(1, ii)	bootstrap	59.5	41.5	0.359	96.9	0.077	95.5	0.084
		de-sparsified	70.5	69.0	0.485	88.4	0.483	88.0	0.483
		de-biased	64.0	68.5	0.429	99.2	0.428	98.4	0.428
	(2, i)	bootstrap	62.0	46.5	0.416	96.5	0.095	95.3	0.103
		de-sparsified	72.0	85.5	0.550	83.3	0.554	83.4	0.554
		de-biased	65.5	89.0	0.506	99.3	0.510	99.1	0.510
	(2, ii)	bootstrap	71.0	40.5	0.374	96.8	0.099	95.4	0.106
		de-sparsified	78.5	83.5	0.501	81.2	0.499	81.2	0.499
		de-biased	73.0	85.0	0.445	99.2	0.444	98.8	0.444

PWR: power; r: coverage rate; L interval length.

method in presence of between-group correlations, while the group-bound method might be more appropriate when groups are defined by clustering highly correlated variables together. Since its target application is different, I exclude the group-bound method from the following comparisons.

3.2.3 Individual inference

Since the de-sparsified lasso was designed without considering variable grouping, I conducted another set of comparisons on inference about individual coefficients. I included the de-biased lasso in these comparisons as well. For my bootstrap method, I completely ignore any group

structure and set $p_j = 1$ for all j throughout all the steps in my implementation. Under this setting, the confidence region $R_j(\alpha)$ in (3.5) reduces to an interval. I applied the three methods on the same data sets used in the previous comparison to construct 95% confidence intervals for individual coefficients and to test $H_k : \beta_{0k} = 0$ for $k \in \mathbb{N}_p$.

Let r_M, r_{M^c} and r denote the coverage rates for active, zero, and all coefficients, respectively, and let L_M, L_{M^c} and L be the corresponding average interval lengths. Note that $1 - r_{M^c}$ reports the type-I error rate of the test and the power can be calculated by checking whether or not an estimated interval for an active coefficient covers zero. Reported in Table 3.2 are the average results for each of the eight data generation settings in the Section 3.2.2 simulation study.

Largely consistent with the results in Table 3.1, I see that my bootstrap method shows a good control of the type-I error, implied by the observation that r_{M^c}, the coverage rate for zero coefficients, is slightly greater than but very close to 95%. The r_{M^c} for the de-sparsified lasso method is uniformly lower than 0.9, dropping to 0.8 in some cases, implying that its type-I error rate is again substantially higher than the desired level of 5%. Although this might lead to some moderate degree of increase in power, I argue that a strict control of false discoveries is critical for large-scale screening when p is large and $p \gg q_0$. For instance, the type-I error rate of the de-sparsified lasso is around 15% for $n = 100, p = 400$. This means that it brought about $0.1p = 40$ more than expected false positives, which was much larger than the number of true positives $q_0 = 10$. This would be a severe disadvantage of the de-sparsified lasso method in the typical high-dimensional and sparse setting, $p \gg q_0$, under which the method was developed. The de-biased lasso, on the contrary, reached very high coverage of zero coefficients, close to or above 99% for all the cases. Its coverage of nonzero coefficients is also seen to be higher than the other two methods. However, the coverage rates for active coefficients of all three methods can be substantially lower than the desired level in many cases. For my method, this was caused by the inaccuracy in detecting active coefficients by thresholding a lasso estimate, which kept only 30% to 40% of them for the case $(n, p) = (100, 400)$. Even including the largest 20 components of the lasso would still identify $< 75\%$ of the true active variables. Furthermore, without grouping the signal

Table 3.3: Average runtimes (in seconds) of de-sparsified lasso/bootstrap

$n \setminus p$	100	200	400	800
50	12.98/11.03	29.29/22.17	73.61/46.55	167.53/98.37
100		41.19/20.72	108.62/42.54	233.26/83.80
200			223.30/43.86	496.69/91.54
400				1242.78/102.33

strength became small which violates Assumption 2 in Zhou and Min (2017b), as β_{0k} was uniformly distributed over $(-1, 1)$. I observe that the interval lengths between active and zero coefficients are very different for my method but are almost identical for the two de-biased lasso methods. For $j \notin A_0$, the variance of the lasso $\hat{\beta}_j$ is in general smaller and $\hat{\beta}_j$ can be exactly zero. My method makes use of such sparsity to improve the efficiency of interval estimation for zero coefficients. The de-biased lasso $\hat{b}$ (1.21) de-sparsifies all components of the lasso, in some sense averaging the uncertainty over all coefficients.

The effect of grouping variables can be seen by comparing the results for my parametric bootstrap with those in Table 3.1: At almost the same level of type-I error rate, its power and coverage rate of active coefficients can be boosted substantially through either way of grouping, which numerically confirms my motivation to group coefficients for a more sensitive detection of signals.

Finally, I compared the running time between my method applied to the lasso and the de-sparsified lasso method. The time complexity of my method is in proportion to the bootstrap sample size N, which has been fixed to $N = 300$ above. I ran both methods without parallelization to make inference about all p coefficients. Reported in Table 3.3 are the average running times per data set of the two methods for 10 different combinations of (n, p). One sees that my method is uniformly faster across all scenarios and can be significantly faster when the data size is large. For example, my bootstrap method is 12 times faster than the de-sparsified lasso when $n = 400$ and $p = 800$, which corresponds to one order of magnitude improvement in terms of speed. Bootstrapping the group lasso is even faster, since the group lasso in general runs faster than the lasso.

Table 3.4: Coverage rate, power and false positive rate in presence of weak and dense signals

		$\epsilon = 0.02$				$\epsilon = 0.2$			
	(a, d)	(1, i)	(1, ii)	(2, i)	(2, ii)	(1, i)	(1, ii)	(2, i)	(2, ii)
bootstrap	r_S	86.0	86.0	83.0	84.0	86.0	87.0	82.0	87.0
	r_W	42.0	60.0	52.0	62.0	46.0	12.0	14.0	22.0
	PWR_S	77.0	94.0	75.0	94.0	66.0	94.0	68.0	87.0
	PWR_W	8.0	6.0	4.0	6.0	84.0	54.0	70.0	78.0
	FPR	4.9	4.2	5.0	4.1	2.6	3.6	3.2	2.2
de-sparsified (Bonferroni)	PWR_S	99.0	99.0	100.0	100.0	97.0	99.0	98.0	97.0
	PWR_W	18.0	32.0	8.0	20.0	84.0	48.0	70.0	78.0
	FPR	20.8	23.9	16.2	18.6	18.2	27.5	16.9	19.1
de-sparsified (Wald)	PWR_S	71.0	88.0	83.0	91.0	58.0	85.0	70.0	81.0
	PWR_W	0.0	0.0	0.0	0.0	30.0	0.0	0.0	0.2
	FPR	0.0	0.2	0.1	0.1	0.0	0.0	0.0	0.3

r_S, r_W: coverage rate for strong and weak signal groups, respectively; PWR_S, PWR_W: power for strong and weak signal groups, respectively; FPR: false positive rate.

3.2.4 Weak and dense signals

In the second simulation study, I added a third active group of 10 coefficients to the setting of $(n, p) = (100, 400)$ with grouping $\mathcal{P}_2$. I set the coefficients $\beta_{0k} \in \{\pm\epsilon\}$ in the third group and chose $\epsilon \in \{0.02, 0.2\}$. The signal of this group, in particular when $\epsilon = 0.02$, was much weaker than that of the first two groups. The vector β_0 also became denser with $s_0 = 3$ and $q_0 = 20$. Both aspects make the data sets more challenging for an inferential method. I note that neither the sparsity condition nor the signal strength condition in Assumption 2 from Zhou and Min (2017b) is satisfied. The results here thus can indicate how the bootstrap method works when key assumptions of my asymptotic theory are violated. To obtain accurate estimation of coverage rates in presence of such small signals, I increased the number of data sets generated in each setting to $K = 50$.

I applied my bootstrap method and the de-sparsified lasso with Bonferroni adjustment to perform group inference on these data sets, as I did in Section 3.2.2. Moreover, I conducted a Wald test with the de-sparsified lasso $\hat{b}$ as follows. The asymptotic distribution of $\hat{b}$ (1.21) implies that, for a fixed group G of size m, $T_G = n(\hat{b}_G - \beta_{0G})^\mathsf{T} V_{GG}^{-1} (\hat{b}_G - \beta_{0G})$ follows a χ^2 distribution with m degrees of freedom as $n \to \infty$, where V is the covariance of the Gaussian

random vector W. Thus, one may use the statistic T_G to test whether a group of coefficients is zero such as in $H_{0,j} : \beta_{0(j)} = 0$. The results of the three methods are reported in Table 3.4. To highlight the expected difference in performance, I separately report the coverage rate and power for the strong groups, r_S and PWR_S, and those for the weak group, r_W and PWR_W.

Again my bootstrap method achieved a good control of type-I error rate, all around or below 5%. The coverage rate and power for the strong groups are comparable to those in Table 3.1, indicating that they were not affected by the inclusion of a weak group. The power for detecting the third group is low when $\epsilon = 0.02$, which is fully expected given the low signal strength, and becomes much higher when ϵ is increased to 0.2. As discussed above, the data simulated here do not satisfy Assumption 2 in Zhou and Min (2017b). As a consequence, the bootstrap samples might not provide a good approximation to the sampling distribution, which could be a reason for the low coverage rate of the weak group. I observe that r_W was in general higher when $\epsilon = 0.02$ than when $\epsilon = 0.2$. This is because $\sup_{S_2} \|\beta_{0(j)}\|$ with $\epsilon = 0.02$ is closer to the requirement in Assumption 2 in Zhou and Min (2017b) for small coefficients. The de-sparsified lasso with Bonferroni correction failed to control the type-I error rate at the desired level. While it shows a higher detection power for the weak group when $\epsilon = 0.02$, its power is largely comparable to my method when $\epsilon = 0.2$. The gain in power could simply be the result of high false positive rates. Compared to Table 3.1, I see a decrease in the false positive rates. This is because the error variance σ^2 was overestimated for these data sets, which alleviated the underestimate of p-values by the de-sparsified lasso. On the contrary, the Wald test seems too conservative, almost never rejecting any zero group. Its power of detecting the weak group is close to zero for most of the cases. These results are the consequence of ignoring the term Δ in (1.20), which introduces systematic bias in the Wald test statistic for finite samples. This numerical comparison demonstrates the advantage of my bootstrap method in the existence of weak coefficient groups under a relatively dense setting.

3.2.5 Real data designs

I further tested my method on design matrices drawn from a gene expression data set (Ivanova et al., 2006), which contains expression profiles for about $40,000$ mouse genes across $n = 70$ samples. The expression profile of each gene was transformed to a standard normal distribution via quantile transformation. The following procedure was used to generate data sets for my comparison. First, randomly pick p genes and denote their expression profiles by $X_j \in \mathbb{R}^n$ for $j = 1, \ldots, p$. I calculate the correlation coefficients $(r_{ij})_{p\times p}$ among X_j's and the total absolute correlation $r_{i\bullet} = \sum_j |r_{ij}|$ for each gene $i \in \mathbb{N}_p$. For the gene with the highest $r_{i\bullet}$, I identify the $(m-1)$ genes that have the highest absolute correlation with this gene. These m genes are put into one group of size m. Then I remove them from the gene set and repeat this grouping process until I partition all p genes into $J = p/m$ groups. This grouping mechanism results in high correlation among covariates in the same group. Next, fixing the first s_0 groups to be active, I draw their coefficients $\beta_{0k} \sim \mathcal{U}(-b, b)$. The parameters in the above procedure were chosen as $p \in \{500, 1000\}$, $b \in \{1, 3, 5\}$, $m = 10$, and $s_0 = 3$. For each combination of (p, b), I obtained $K = 100$ independent realizations of (X, β_0). Given each realization, a range of the noise variance $\sigma^2 \in \{0.1, 0.5, 1\}$ was then used to simulate the response $y \sim \mathcal{N}_n(X\beta_0, \sigma^2\mathbf{I}_n)$. Compared to the data generation settings in Section 3.2.1, data sets in this subsection have a smaller sample size $n = 70$ but a higher dimension p, and the correlation among the covariates is much higher. These put great challenges on an inferential method.

I applied both the bootstrap and the de-sparsified lasso methods to perform group inference as I did in Section 3.2.2. As reported in Table 3.5, my bootstrap method gives a good and slightly conservative control over type-I errors, with false positive rates all close to but below 5%, the desired level. Its power in general increases as the signal-to-noise ratio increases and is seen to be around 0.5 when the signal-to-noise ratio is reasonably high. On the contrary, the type-I error rate of the de-sparsified lasso method, not reported in the table, was even > 0.9 for most of the cases, showing that it failed to provide an acceptable p-value approximation for these data sets. This might be caused by the facts that this method is not

Table 3.5: Comparison of power and false positive rate on gene expression data

Data Setting			Group inference bootstrap		Individual inference ($p_j = 1$) bootstrap		de-sparsified lasso		
p	β_0	σ^2	PWR	FPR	PWR	FPR	PWR	FPR	PWR*
500	$(-1, 1)$	0.1	50.3	1.6	14.2	3.1	41.9	19.3	12.9
		0.5	24.7	1.8	13.1	3.1	36.1	15.0	12.0
		1	24.7	2.4	12.0	3.0	31.8	12.7	13.0
	$(-3, 3)$	0.1	61.3	1.0	15.7	3.3	51.0	27.1	13.2
		0.5	54.7	1.1	15.0	3.2	46.2	21.0	10.7
		1	48.7	1.2	15.1	3.4	47.3	22.4	11.2
	$(-5, 5)$	0.1	58.7	1.4	14.9	3.1	48.5	24.4	14.3
		0.5	57.7	1.2	14.9	3.1	43.1	20.0	10.8
		1	55.7	0.9	14.2	3.1	44.4	19.2	14.3
1000	$(-1, 1)$	0.1	41.0	1.1	12.4	2.0	36.3	14.0	12.0
		0.5	29.3	2.0	11.5	2.0	29.8	10.6	15.3
		1	30.3	1.1	10.4	1.8	29.4	10.5	13.2
	$(-3, 3)$	0.1	41.0	0.8	13.9	2.1	34.8	13.9	14.2
		0.5	33.7	0.7	12.2	2.1	37.4	15.5	13.8
		1	46.7	1.3	12.8	2.1	39.5	17.7	13.4
	$(-5, 5)$	0.1	50.0	1.0	12.4	2.1	33.6	12.3	13.9
		0.5	47.7	0.9	12.8	2.1	34.3	14.3	10.9
		1	45.7	0.8	11.8	2.1	36.9	15.9	11.9

FPR: false positive rate; PWR: power; PWR*: power of the de-sparsified lasso after matching false positive rate to 5%.

designed for group inference and that $n = 70$ is too small for asymptotic approximation. To conduct a complete comparison, I then used both methods to make inference about individual coefficients as in Section 3.2.3, in which the group structures were totally ignored in my method by setting all $p_j = 1$. My method again controlled the type-I error to a level slightly lower than 5%, but showed a decrease in power, as expected, without utilizing grouping. The false positive rate of the de-sparsified lasso became smaller for individual inference, ranging between 15% and 30%, but still far from the desired level of 5%. This makes it difficult to compare power between the two methods, as the observed higher power of one method could simply come at the cost of more false positives. To resolve this issue, I sorted the p-values for all the zero coefficients output from the de-sparsified lasso method, and chose a cutoff p^* such that 5% of them would be rejected. In this way, the false positive rate by definition is

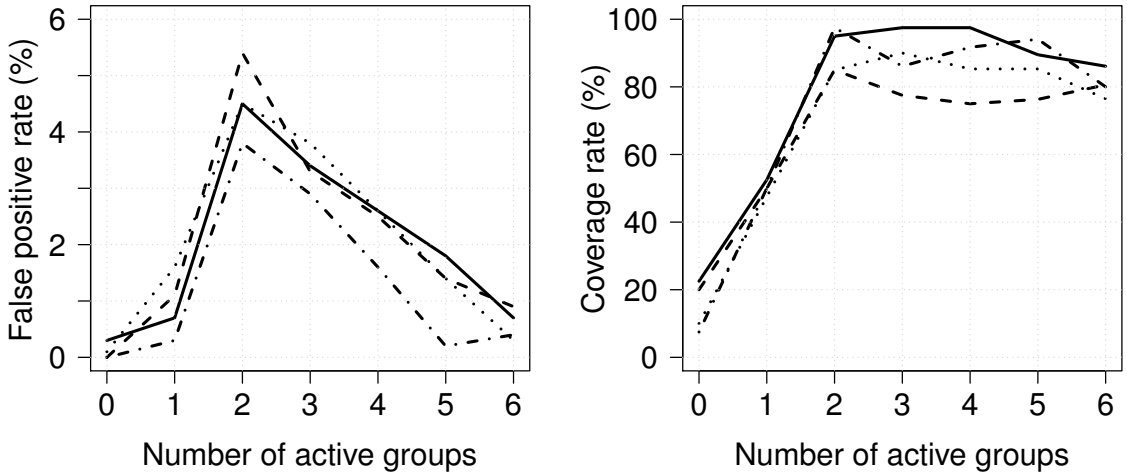

Figure 3.1: Sensitivity analysis on thresholding: (left) false positive rate and (right) coverage rate of active groups against the number of active groups of thresholded group lasso for the four settings of $(a, d) = (1, \text{i})$ (solid), $(1, \text{ii})$ (dash), $(2, \text{i})$ (dot), and $(2, \text{ii})$ (dot-dash).

always 5%, slightly higher than that of my bootstrap method, while the corresponding power becomes largely comparable. I also noticed that the overestimate of the significance level by the de-sparsified lasso method was severe for these data sets: To achieve the target type-I error rate of 5%, the cutoff for its p-values turned out to be < 0.002 for all the settings and was even much smaller for many of them.

This comparison shows that my parametric bootstrap method can achieve a desired level of false positive control in presence of high correlation among a large number of predictors. It again confirms that grouping variables can lead to substantial power gain.

3.2.6 Sensitivity to thresholding

With $\hat{\lambda}$ chosen by cross-validation, the only parameter that requires user input in my method is the threshold value b_{th}. The following experiment has been conducted to examine how sensitive my method is to this parameter. Given the group lasso solution $\hat{\beta}$, I reorder its groups so that $\|\hat{\beta}_{(1)}\| \geq \ldots \geq \|\hat{\beta}_{(J)}\|$. Then I choose a range of threshold values, $b_{\text{th}} = \|\hat{\beta}_{(k+1)}\|$, such that the thresholded $\tilde{\beta}$ has k active groups for $k = 0, \ldots, K$, say $K = 6$. I applied this procedure on the simulated data sets generated from the four settings with

$n = 100, p = 400$ and $\mathcal{G} = \mathcal{P}_2$ in Table 3.1, which have $s_0 = 2$ active groups. These were the most difficult settings for which my bootstrap method had the lowest power and coverage rate r_M. Figure 3.1 plots the curve of the false positive rate and the curve of the active group coverage rate r_M against k, the number of active groups after thresholding, in each of the four settings.

The false positive rates are well-controlled at the desired level of 5% for all the threshold values. They are around 0.05 when b_{th} is well-chosen so that $\tilde{\beta}$ has $k = s_0 = 2$ active groups, and become smaller when b_{th} deviates from the optimal value. This suggests that my method is not sensitive to the threshold value in terms of type-I error control. The coverage of the active groups stays at a high level when $\tilde{\beta}$ contains two or more active groups, but can be substantially lower if one or both of the true active groups are missing. Thus, including a few zero groups in the active set of $\tilde{\beta}$ will not hurt the performance of my method that much, since via refitted least-squares, the estimated coefficients of these groups tend to be small. Similar patterns were observed for inference on individual coefficients, when b_{th} was chosen for $\tilde{\beta}$ to have up to 30 nonzero coefficients while the true active set contained only 10 variables.

3.3 Estimator augmentation in group lasso

In this section, I introduce some applications which make inference about β_0 by utilizing the joint density of the augmented block lasso estimator (Zhou and Min, 2017a) which was reviewed in Section 1.2.2. Consider the group lasso estimator with ℓ_2 group regularization. The goal is to test the hypothesis $H_{0,G} : \beta_{0G} = 0$ or to construct confidence regions for β_{0G} for some $G \subset \mathbb{N}_p$. Without loss of generality, assume that $G = \mathcal{G}_j$ for some $j \in \mathbb{N}_J$ so that my goal is to infer $\beta_{0(j)}$. Denote the null hypotheses by $H_{0,j} : \beta_{0(j)} = 0$ for $j \in \mathbb{N}_J$.

3.3.1 De-biased parametric bootstrap

Consider inference with an estimator in the form of $\hat{b} = \hat{b}(\hat{\beta}, S) \in \mathbb{R}^p$, a mapping of the augmented estimator $(\hat{\beta}, S)$. One such approach that has drawn recent attention is the

de-biased lasso and its generalization to the de-biased group lasso. Given a $p \times p$ matrix $\hat{\Theta} = \hat{\Theta}(X)$, a form of the de-biased estimator may be expressed as

$$\hat{b}(\hat{\beta}, S) = \hat{\beta} + \hat{\Theta} X^{\mathsf{T}}(y - X\hat{\beta})/n = \hat{\beta} + \lambda \hat{\Theta} W S, \tag{3.4}$$

where $(\hat{\beta}, S)$ is either the augmented lasso or the augmented group lasso. Different de-biased estimators have been constructed with different $\hat{\Theta}$, which is often some version of a relaxed inverse of the Gram matrix Ψ. It is usually impossible to obtain the exact distribution of $(\hat{b} - \beta_0)$ for a finite sample. Thus, bootstrap methods have been developed (Zhang and Cheng, 2017; Dezeure et al., 2017) with improved performance compared to asymptotic approximations for the de-biased methods.

Assuming the error distribution is $\mathcal{N}_n(0, \sigma^2 \mathbf{I}_n)$ with a known σ^2 for now, I use the parametric bootstrap $PB(\tilde{\beta}, \sigma^2, \lambda)$ in Algorithm 4. This time, for every y^* I generate, I also compute S^* via (1.3). Let $(\hat{\beta}^*, S^*)$ be the augmented estimators drawn in such way. Let $\hat{b}^* = \hat{b}(\hat{\beta}^*, S^*)$. Choosing a function $h_j : \mathbb{R}^{p_j} \to [0, \infty)$, I estimate its $(1-\gamma)$-quantile $h_{j,(1-\gamma)}$ from a large bootstrap sample such that

$$\mathbb{P}\left\{ h_j(\hat{b}^*_{(j)} - \tilde{\beta}_{(j)}) > h_{j,(1-\gamma)} \middle| \tilde{\beta} \right\} = \gamma.$$

Then, a $(1-\gamma)$ confidence region for $\beta_{0(j)}$ can be constructed in the form of

$$R_j(\gamma) = \left\{ \theta \in \mathbb{R}^{p_j} : h_j(\hat{b}_{(j)} - \theta) \le h_{j,(1-\gamma)} \right\}. \tag{3.5}$$

By duality the p-value for testing $H_{0,j}$ is approximated by the tail probability

$$\mathbb{P}\left\{ h_j(\hat{b}^*_{(j)} - \tilde{\beta}_{(j)}) \ge h_j(\hat{b}_{(j)}) \middle| \tilde{\beta} \right\}. \tag{3.6}$$

Common choices of h_j include, for example, various norms and $h_j(\theta) = \|X_{(j)}\theta\|$. Although out of the scope of this chapter, the asymptotic validity of (3.5) and (3.6) comes from the fact that $(\hat{b}_{(j)} - \beta_{0(j)})$ is an asymptotic pivot with a careful choice of $\hat{\Theta}$ (Mitra and Zhang,

2016; van de Geer et al., 2014).

An interesting and key observation is that the joint density of $[\hat{\beta}^*, S^* \mid \tilde{\beta}]$ is explicitly given in Theorem 2, with $\tilde{\beta}$ in place of β_0, through its equivalent representation. Denote this density by $f_M(r_M, s; \tilde{\beta}, \sigma^2, \lambda)$ to emphasize its dependence on $(\tilde{\beta}, \sigma^2, \lambda)$, i.e.

$$f_M(r_M, s; \tilde{\beta}, \sigma^2, \lambda) = f_E\big(\tilde{H}(r_M, s; \mu_0, \lambda)\big)\Big|J_M(r_M, s; \lambda)\Big|. \tag{3.7}$$

In principle, I can use Monte Carlo methods, such as importance sampling and MCMC, to draw $(\hat{\beta}^*, S^*)$ and obtain a sample of $\hat{b}^* = \hat{b}(\hat{\beta}^*, S^*)$, which serve as alternatives to the above bootstrap sampling. Monte Carlo methods may bring computational efficiency and flexibility compared to parametric bootstrap. In the following, I will demonstrate the efficiency of importance sampling in calculating tail probabilities as in (3.6), which is a prominent difficulty for the bootstrap. Monte Carlo methods for other applications, including those with an estimated error distribution, are discussed in Section 3.3.5.

3.3.2 Importance sampling

Recall that $\mathcal{M} = \mathcal{G}(\hat{\beta})$ is a random variable and M is the value of $\mathcal{M}$. Given M, let $\Omega_M = (\mathbb{R}^+)^{|M|} \times \mathcal{S}_M$ be the sample space of $(\hat{\gamma}_M, S)$ given M where $\mathcal{S}_M$ is defined as in (1.15). Let Ω be the sample space for the augmented estimator $(\hat{\gamma}_{\mathcal{M}}, S, \mathcal{M})$:

$$\Omega = \bigcup_{|M| \le n} \Omega_M \times \{M\}.$$

The following simple fact about the parameterization of $\mathcal{S}_M$ is useful for designing proposal distributions in importance sampling.

Lemma 8. *Let $\alpha \in (1, \infty)$. Define $\mathscr{M} = \{M \subset \mathbb{N}_J : |M| \le n\}$. For each $M \in \mathscr{M}$, the manifold $\mathcal{S}_M$, except for a set of measure zero, can be parameterized by s_F such that the index set $F = F(M)$ only depends on M.*

A consequence of Lemma 8 is that I may use the same volume element $d\theta = dr_M ds_F$

almost everywhere in the subspace Ω_M, which eases my development of a Monte Carlo algorithm. Suppose that $q_M(r_M, s)$ is the density of a distribution over Ω with respect to $d\theta$ such that $\sum_M \int_{\Omega_M} q_M(r_M, s)d\theta = 1$. As long as the support of q_M is Ω_M for all $M \in \mathscr{M}$, it can be used as a proposal distribution in importance sampling. With a little abuse of notation, put $\theta = (r_M, s) \in \Omega_M$ so that (θ, M) represents a point in the sample space Ω at which the volume element is $d\theta$. Suppose I want to estimate the expectation of a function $h(\hat{\beta}, S) = h(\hat{\gamma}, S, \mathcal{M})$ with respect to f_M, using $(\hat{\beta}, S)$ and $(\hat{\gamma}, S, \mathcal{M})$ interchangeably. Importance sampling can be readily implemented given the densities f_M and q_M. Draw (θ_t, M_t) from the proposal $q_M(\theta)$ for $t = 1, \ldots, N$ and calculate importance weights $\omega_t = f_{M_t}(\theta_t)/q_{M_t}(\theta_t)$. Then by the law of large numbers, the weighted sample mean

$$\hat{h} = \frac{\sum_{t=1}^N \omega_t h(\theta_t, M_t)}{\sum_{t=1}^N \omega_t} \xrightarrow{a.s.} \mathbb{E}[h(\hat{\gamma}, S, \mathcal{M})]$$

provides the desired estimate. To estimate the probability in (3.6), h is taken to be the indicator function of the event of interest. When the true $\beta_{0(j)} \neq 0$, the p-value (3.6) can be tiny, and bootstrap may fail to provide a meaningful estimate of the significance level. In such cases, it is much more efficient to use importance sampling with a proposal distribution that has a higher chance to reach the tail of the bootstrap distribution $f_M(r_M, s; \tilde{\beta}, \sigma^2, \lambda)$.

I design two types of proposal distributions. The first type of proposals draw $(\hat{\beta}^*, S^*)$ by the bootstrap algorithm $PB(\beta^\dagger, c\sigma^2, \lambda^\dagger)$ with a proper choice of $(\beta^\dagger, c, \lambda^\dagger)$, where $c > 0$ is a constant. The proposal distribution has density $f_M(r_M, s; \beta^\dagger, c\sigma^2, \lambda^\dagger)$, again by Theorem 2. By increasing the error variance with $c > 1$, choosing $\beta^\dagger \neq \tilde{\beta}$, and possibly with a different $\lambda^\dagger$, I can propose samples in the region of interest in (3.6) which has a small probability with respect to the target distribution. The Jacobian term $J_M(r_M, s; \lambda)$ in Theorem 2 is the time-consuming part in evaluating the densities for calculating importance weights. If I choose $\lambda^\dagger = \lambda$, however, this term will cancel out and the importance weight is simply the ratio of two normal densities, whose calculation is almost costless. My empirical study shows that this choice gives comparable estimation accuracy and thus I always let $\lambda^\dagger = \lambda$. Denote by $IS(\beta^\dagger, c)$ the importance sampling with the first type of proposals. My second design uses

a mixture of two proposal distributions with different $\beta^\dagger$ and c, which has more flexibility in shifting samples to multiple regions of interest. Again the Jacobian term cancels out in the importance weight (3.8). My importance sampling with a mixture proposal is detailed in the following algorithm. For brevity, write

$$\tilde{H}(\hat{\beta}, S; \beta_0) = \sqrt{n}(X^\mathsf{T})^+(\Psi\hat{\beta} + \lambda W S - \Psi\beta_0),$$

which is identical to the $\tilde{H}$ in (1.14).

Algorithm 5 ($IS(a_1, \beta_1^\dagger, c_1; a_2, \beta_2^\dagger, c_2)$)**.** Given $a_1 + a_2 = 1$, $\beta_1^\dagger, \beta_2^\dagger \in \mathbb{R}^p$ and $c_1, c_2 > 0$,

(1) draw Z from $\{1, 2\}$ with probabilities $\{a_1, a_2\}$ and $(\hat{\beta}^*, S^*)$ from $PB(\beta_Z^\dagger, c_Z\sigma^2, \lambda)$;

(2) calculate importance weight

$$\omega^* = \frac{\phi_n\left(\tilde{H}(\hat{\beta}^*, S^*; \tilde{\beta}); \sigma^2/n\right)}{\sum_{k=1}^2 a_k\, \phi_n\left(\tilde{H}(\hat{\beta}^*, S^*; \beta_k^\dagger); c_k\sigma^2/n\right)}. \tag{3.8}$$

Remark 3. The first algorithm $IS(\beta^\dagger, c)$ can be regarded as a special case of Algorithm 5 with $a_1 = 1$, $\beta_1^\dagger = \beta^\dagger$ and $c_1 = c$. One can easily generalize Algorithm 5 to a mixture proposal with $K \geq 3$ component distributions. For other error distributions, I simply replace ϕ_n in (3.8) by f_E, the density of $\varepsilon/\sqrt{n}$.

In my numerical results, the efficiency of an importance sampling estimate is measured by its coefficient of variation (cv) across multiple independent runs and compared with direct bootstrap outlined in Algorithm 4.

3.3.3 Group lasso

I begin with a simpler application to test the complete null hypothesis $H_0 : \beta_0 = 0$ using the statistic $T = h(\hat{\beta}) = \sum_j \|\hat{\beta}_{(j)}\|$, where $\hat{\beta}$ is the group lasso for a particular λ. In this case, my target density $f_M(r_M, s; \beta_0 = 0, \sigma^2, \lambda)$ determines the exact distribution of T under H_0.

I set the group size $p_j = 10$ for all groups and fixed $\sigma^2 = 1$. Each row of X was drawn

Table 3.6: Simulated datasets for testing complete null hypothesis

Dataset	(n, p)	(ρ_1, ρ_2)	λ	T
1-10	$(30, 100)$	$(0, 0)$	$(0.396, 0.796)$	$(0.017, 0.337)$
11-20	$(30, 100)$	$(0, 0)$	$(0.554, 1.613)$	$(0.460, 1.964)$
21-30	$(30, 100)$	$(0.5, 0)$	$(0.956, 2.650)$	$(0.045, 2.186)$

from $\mathcal{N}_p(0, \Sigma)$, where the diagonal elements of Σ are all 1. The off-diagonal elements $\Sigma_{ij} = \rho_1$ if i, j are in the same group and $\Sigma_{ij} = \rho_2$ otherwise. I simulated 30 datasets with parameters (n, p, ρ_1, ρ_2) reported in Table 3.6. Put $v = (1, 1, 1, 1, -1, -1, -1, -1, 0, 0)$. For the first 10 datasets, I chose $\beta_0 = 0$ so that H_0 is true. For the other 20 datasets, the first two groups of β_0 were active, with $\beta_{0(1)} = \beta_{0(2)} = v/2$ for datasets 11 to 20 and $\beta_{0(1)} = \beta_{0(2)} = v$ for datasets 21 to 30. For each dataset, λ was chosen to be the smallest value such that the group lasso solution had two active groups. The range of λ and that of the statistic T across the simulated datasets are reported in Table 3.6 as well.

I applied the algorithm $IS(0, 5)$ to generate $N = 100,000$ samples. Denote the samples by $\hat{\beta}_t^*$, with importance weight ω_t, for $t = 1, \ldots, N$. The p-value for the observed statistic T was then estimated by

$$\hat{q}^{(IS)} = \frac{\sum_{t=1}^N \omega_t I(h(\hat{\beta}_t^*) \geq T)}{\sum_{t=1}^N \omega_t}. \tag{3.9}$$

This procedure was repeated 20 times independently for each dataset to calculate the mean $\bar{q}$ and the standard deviation of $\hat{q}^{(IS)}$, from which I calculated $\text{cv}(\hat{q}^{(IS)})$. If I had used the bootstrap algorithm $PB(0, \sigma^2, \lambda)$ for the same N to estimate the p-value, denoted by $\hat{q}^{(PB)}$, its cv would have been close to $\sqrt{(1-\bar{q})/(N\bar{q})}$. Figure 3.2 plots $\log_{10}(\bar{q})$, $\text{cv}(\hat{q}^{(IS)})$ and $\log_{10}\{\text{cv}(\hat{q}^{(PB)})/\text{cv}(\hat{q}^{(IS)})\}$ for the 30 datasets. I observe from the ratios of cv's in panel (c) that, for datasets 11 to 30, the importance sampling estimates are much more accurate, while the estimated p-values, as shown in panel (a), are very small. For many of these 20 datasets, the improvement of importance sampling over bootstrap can be five or more orders of magnitude. The p-values are insignificant for the first 10 datasets, in which the null hypothesis is true. In a majority of these cases, the importance sampling estimates are slightly less accurate than the bootstrap estimates, which is fully expected.

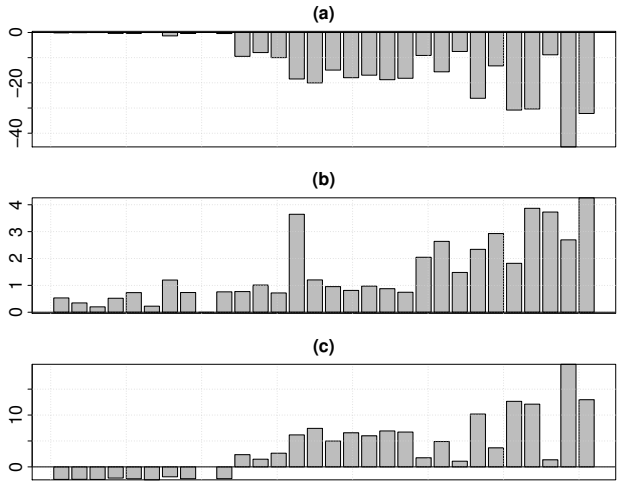

Figure 3.2: Estimation of p-values for testing H_0 with the group lasso. (a) $\log_{10} \bar{q}$, (b) $\text{cv}(\hat{q}^{(IS)})$ and (c) $\log_{10}\{\text{cv}(\hat{q}^{(PB)})/\text{cv}(\hat{q}^{(IS)})\}$. The result for a dataset is reported by a vertical bar in each plot.

3.3.4 A de-biased approach

The second application concerns a de-biased group lasso in the form of (3.4). Since my method applies to any choice of $\hat{\Theta}$, to simplify the discussion I set $\hat{\Theta} = \Sigma^{-1}$ instead of using a particular estimate, where Σ is the population covariance of X. The test statistic is chosen as $h_j(\hat{b}_{(j)}) = \|X_{(j)}\hat{b}_{(j)}\| := T_j$ in (3.6).

I simulated 20 datasets independently under the same settings as those for datasets 11 to 30 in Table 3.6. The tuning parameter λ was chosen by the same method as in Section 3.3.3 to calculate the group lasso $\hat{\beta}$ and the de-biased estimate $\hat{b}$ (3.4) for each dataset. Figure 3.3 plots these two estimates for one dataset, in which $\beta_{0(1)} = \beta_{0(2)} = v$ and $\beta_{0(j)} = 0$ for $j > 2$. I see that the de-biased group lasso $\hat{b}$ is not sparse, $\hat{b}_{(j)} \neq 0$ for all j, and its first two groups are closer to the active groups of β_0 than the group lasso. This largely removed the shrinkage in the active coefficients of the group lasso solution and substantially reduced its bias. My goal here is to test $H_{0,1} : \beta_{0(1)} = 0$ by estimating the probability (3.6) for $T_1 = \|X_{(1)}\hat{b}_{(1)}\|$ with a plug-in point estimate $\tilde{\beta}$. The observed value of the test statistic T_1 ranges from 4.4 to 21.2 across the 20 datasets. Due to the asymptotic unbiasedness of $\hat{b}_{(j)}$, the bootstrap distribution $[\hat{b}^*_{(j)} - \tilde{\beta}_{(j)} \mid \tilde{\beta}]$ is not sensitive to the choice of $\tilde{\beta}$ as long as it is sparse. Thus, I choose $\tilde{\beta} = \hat{\beta}$. See Dezeure et al. (2017) for related discussions.

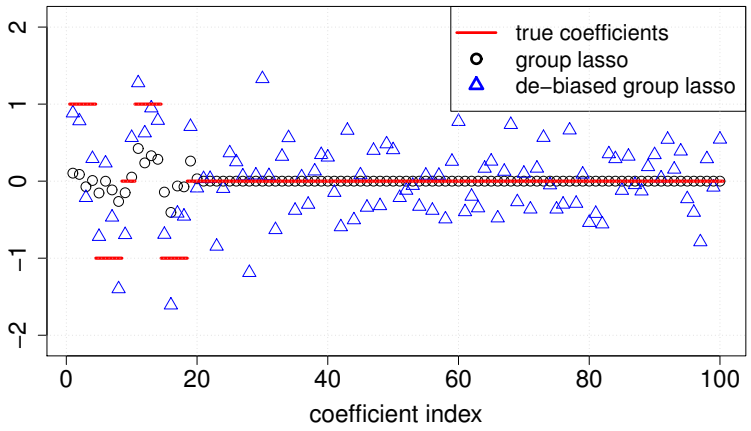

Figure 3.3: The group lasso and de-biased group lasso solutions for one dataset with $p = 100$, where the size of each group is 10.

I designed the following mixture proposal for Algorithm 5 to approximate the p-value (3.6) by importance sampling:

$$a_1 = a_2 = 1/2; c_1 = 2, c_2 = 4; \beta_1^\dagger = \hat{\beta}, \beta_{2(1)}^\dagger = \hat{\beta}_{(1)}/2, \beta_{2(-1)}^\dagger = \hat{\beta}_{(-1)}.$$

Note that $\beta_{2(1)}^\dagger$ is the middle point between $\hat{\beta}_{(1)}$ and 0, serving as a bridge between the target distribution and the null hypothesis $H_{0,1}$. To achieve a wider coverage of the sample space, the error variances of both component distributions were chosen to be greater than σ^2. I applied Algorithm 5 to generate $N = 100,000$ weighted samples $(\hat{\beta}_t^*, S_t^*)$, with weights ω_t, for each dataset. Similar to (3.9), the p-value for T_1 was estimated as

$$\hat{q}^{(IS)} = \frac{\sum_{t=1}^N \omega_t I(\|X_{(1)}(\hat{b}_{t(1)}^* - \hat{\beta}_{(1)})\| \geq T_1)}{\sum_{t=1}^N \omega_t}, \tag{3.10}$$

where $\hat{b}_t^* = \hat{b}(\hat{\beta}_t^*, S_t^*)$ as in (3.4). I replicated this procedure 20 times independently to calculate the cv of $\hat{q}^{(IS)}$ as I did in the previous example. The same comparisons were conducted and the results are reported in Figure 3.4. Strong majority of the p-values were estimated to be significant, since $\beta_{0(1)} \neq 0$ for all 20 datasets. The cv's of the importance sampling estimates are seen to be quite small, which is especially satisfactory for those tiny tail probabilities on the order of 10^{-10} or smaller. As shown in Figure 3.4(c), my importance

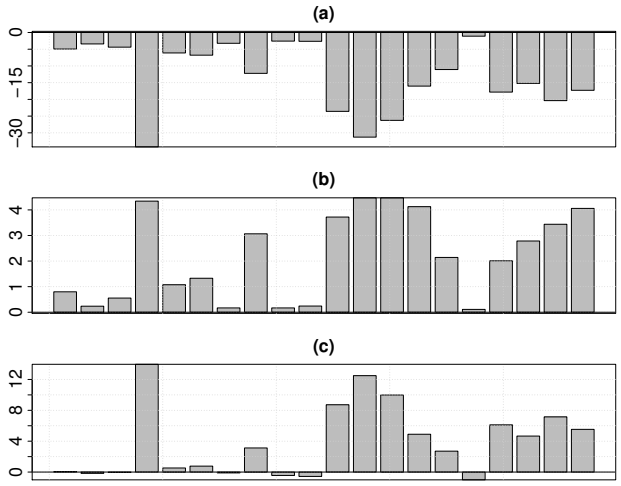

Figure 3.4: Estimation of p-values for testing $H_{0,1}$ with a de-biased group lasso. Plots are in the same format as those in Figure 3.2.

sampling estimation is more efficient than parametric bootstrap for at least 13 out of the 20 datasets, many showing orders of magnitude of improvement. For most of the other datasets, the importance sampling results are very comparable to the results from bootstrap.

Compared to the parametric bootstrap in Algorithm 4, the only additional step in my importance sampling algorithms is to evaluate importance weights, such as (3.8), of which the computing time is negligible relative to computing group lasso solutions. As a result, the total running time of the importance sampling is almost identical to that of the bootstrap sampling. The above two applications thus exemplify the huge gain in estimation accuracy by importance sampling via estimator augmentation at almost identical computing cost. It is worth mentioning that accurate estimation of small p-values is crucial for ranking the importance of predictors and controlling false discoveries in large-scale screening.

3.3.5 Other applications

Given the joint density $f_M(r_M, s; \tilde{\beta}, \sigma^2, \lambda)$, one may design MCMC algorithms to draw samples $(\hat{\beta}^*, S^*)$ from this distribution, which is identical to the distribution of a bootstrap sample generated by $PB(\tilde{\beta}, \sigma^2, \lambda)$ in Algorithm 4. The advantage of an MCMC algorithm is that it does not need to solve a convex optimization program in any of its steps. But

evaluating the Jacobian term in f_M could be time-consuming. Another potential application is conditional sampling from $[\hat{\beta}^*, S^* \mid \hat{\beta}^* \in B]$, which will be useful in post-selection inference. For example, conditioning on the model selected by $\hat{\beta}$, i.e. $G(\hat{\beta}^*) = G(\hat{\beta})$, I may wish to sample from an estimator $\hat{b}^*$ with a nice asymptotic distribution for inference. For this problem, bootstrap may be impractical since the conditioning event is often a rare event. However, from the joint density one can easily obtain the conditional density $\propto f_G(r_G, s)$, where $G = G(\hat{\beta})$, and implement an MCMC algorithm to draw from this conditional distribution. In the case of the lasso, Zhou (2014) implemented an Metropolis-Hastings sampler for such conditional sampling. The more general case for a block lasso will be considered in the future.

Under a Gaussian error assumption, it is a common practice to plug an estimated variance $\hat{\sigma}^2$ in the bootstrap $PB(\tilde{\beta}, \hat{\sigma}^2, \lambda)$. As long as $\hat{\sigma}^2$ is consistent with a certain rate, inference will be valid asymptotically (Dezeure et al., 2017; Zhou and Min, 2017b). Therefore, I can use my importance sampling algorithms with $f_M(r_M, s; \tilde{\beta}, \hat{\sigma}^2, \lambda)$ as the target density. Note that the density f_M (3.7) depends on the error distribution only through the density f_E of $\varepsilon/\sqrt{n}$. Under a general i.i.d. error assumption, estimating f_E reduces to estimating the density of an univariate distribution, which can be done quite accurately even when n is moderate by either a parametric or a nonparametric method. Given an estimate $\hat{f}_E$, my target density is readily obtained with f_E replaced by $\hat{f}_E$. An appealing alternative is to de-bias a scaled group lasso, which estimates σ^2 in a coherent way, for inference as in Mitra and Zhang (2016). As in Section 1.2.3, estimator augmentation can be applied to derive the joint density of an augmented scaled group lasso, including its variance estimator $\hat{\sigma}^2$. Given the density, one can follow the same importance sampling algorithms for tail probability approximation.

CHAPTER 4

R-package : EAinference

4.1 Introduction

As reviewed in the earlier chapters, high-dimensional inference is now getting a lot of attention. However, not many R packages that support high-dimensional inference methods are available. In this chapter, the R package will be introduced `EAinference` which supports many applications using estimator augmentation, including a parametric bootstrap sampler, computing importance weight, and constructing a post-selection confidence sets/intervals. The package is now available on CRAN.

In my best knowledge, there are three R packages that support high-dimensional inference. First, the `hdi` package (Dezeure et al., 2015) implemented the sample-splitting method, the de-biased methods for lasso and ridge regression, and the group bound method. The `selectiveInference` package (Tibshirani et al., 2017) implemented post-selection inference methods for lasso, lars and the forward stepwise regression which was used in Chapter 2 to compare the proposed method with Lee's method (Lee et al., 2016). The `c060` package (Martin Sill and Zucknick, 2014) implemented stability selection (Meinshausen and Bühlmann, 2010) for lasso and the elastic net.

The unique aspects of `EAinference` package are that (1) it supports sampling y from its conditional density, in particular, drawing y such that $\mathcal{A}(y) = A$ for a given active set A, (2) it can construct confidence sets on multiple post-selection estimators which could not be done by any other packages, (3) since it uses simulation approaches, great flexibility can be obtained. Since `EAinference` can generate samples, users can easily come up with their own hypothesis.

Section 4.2 introduces parametric bootstrap sampler, high-dimensional importance sampler and Markov chain Monte Carlo sampler. Section 4.3 covers two inference methods; (1) post-selection confidence sets which was introduced in Chapter 2, and (2) de-biased confidence intervals. The chapter is conclude by presenting demonstration of my package in Section 4.4.

4.1.1 Common arguments and sample data

Common arguments shared by multiple functions in `EAinference` are listed here. `X` and `Y` are a design matrix $X \in \mathbb{R}^{n \times p}$ and a response vector $y \in \mathbb{R}^n$, respectively. `type` determines the type of lasso estimators which can be chosen among {`"lasso"`, `"grlasso"`, `"slasso"`, `"sgrlasso"`} which represent lasso, group lasso, scaled lasso and scaled group lasso estimator, respectively. `lbd` and `weights` are λ and $\{w_j\}_{1:p}$ in (1.1) and (1.2) and each is set to `rep(1,p)` and `1:p` by default, respectively. Whenever `type` is set to either `"grlasso"` or `"sgrlasso"`, group structure needs to be specified through `group` argument. By default, `group` is set to $\mathbb{N}_p$ so that each variable forms its own group. For example, if $p = 3$, the default group structure will be $(1, 2, 3)$ which makes separate groups for each variable. If one wants to group the first and the second variables together, the group structure needs to be specified as $(1, 1, 2)$. Whenever a function supports the parallel computing, users can set `parallel = TRUE` and specify the number of cores to use with `ncores`.

Every time a new function is introduced, a small data set with $(n, p) = (5, 10)$ will be used to ease understanding. Each column of a design matrix X is independently sampled from $\mathcal{N}_n(0, \mathbf{I}_n)$. The response y is drawn from $\mathcal{N}(X\beta_0, \mathbf{I}_n)$, where $\beta_0 = (1, 1, 0, \ldots, 0)$.

```
R> n <- 5
R> p <- 10
R> beta0 <- c(1, 1, rep(0, 8))
R> X <- matrix(rnorm(n*p), n)
R> Y <- X%*%beta0 + rnorm(n)
```

4.2 Sampling methods

To exploit the fact that the density of lasso, group lasso, scaled lasso and scaled group lasso are known, three sampling methods are developed.

4.2.1 Lasso type estimator

`lassoFit` provides four different penalized estimator along with their subgradients (and the error variance estimate $\hat{\sigma}^2$ for the scaled group lasso) using `gglasso` function in `gglasso` package; (a)lasso and group lasso, (b)scaled lasso and scaled group lasso. Note that if all the group sizes are 1, i.e. $p_j = 1$ for $\forall j$, group lasso and scaled group lasso reduce to lasso and scaled lasso, respectively. Each of the estimator is

$$\begin{aligned} \hat{\beta}^{(a)} &\in \underset{\beta\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \frac{1}{2}\|y - X\beta\|_2^2 + n\lambda \sum_{j=1}^{J} w_j \|\beta_{(j)}\|_2 \right\}, \\ (\hat{\beta}^{(b)}, \hat{\sigma}) &\in \underset{\beta\in\mathbb{R}^p,\ \sigma\in\mathbb{R}^+}{\operatorname{argmin}} \left\{ \frac{1}{2\sigma}\|y - X\beta\|_2^2 + \frac{\sigma}{2} + n\lambda \sum_{j=1}^{J} w_j \|\beta_{(j)}\|_2 \right\}. \end{aligned} \tag{4.1}$$

The corresponding subgradients are

$$\begin{aligned} S^{(a)} &= \frac{1}{n\lambda} W^{-1} \Big\{ X^\mathsf{T} y - X^\mathsf{T} X \hat{\beta}^{(a)} \Big\}, \\ S^{(b)} &= \frac{1}{n\lambda\hat{\sigma}} W^{-1} \Big\{ X^\mathsf{T} y - X^\mathsf{T} X \hat{\beta}^{(b)} \Big\}, \end{aligned} \tag{4.2}$$

where $W = \operatorname{diag}\{w_1 \mathbf{I}_{p_1}, \dots w_J \mathbf{I}_{p_J}\}$. The function can be called as follows:

```
R> lassoFit(X, Y, type, lbd, group = 1:ncol(X), weights = rep(1, max(group)),
+     verbose = FALSE, ...)
```

`lbd` can be a positive number as described earlier or it can be selected via cross validation; `lbd = "cv.min"` finds λ which gives the minimum cross-validation mean-squared error and `lbd = "cv.1se"` uses one standard error rule from cross-validation.

Below is the example code with `lbd = 0.4`. `group` and `weights` are not specified thus the default values are used. The output includes the lasso estimate(`B0`), the corresponding subgradient(`S0`), λ of the choice(`lbd`), weights $\{w_j\}_{1:p}$(`weights`), and the group structure(`group`).

```
R> lassoEst <- lassoFit(X = X, Y = Y, type = "lasso", lbd = .4)
R> lassoEst
$B0
 [1] 1.02808174 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
 [7] 0.00000000 0.01433208 0.00000000 0.90947626

$S0
 [1]  0.999999997  0.648516454  0.031990556 -0.006260897  0.095389758
 [6]  0.792181175  0.766627854  0.999999974  0.472402860  1.000000000

$lbd
[1] 0.4

$weights
 [1] 1 1 1 1 1 1 1 1 1 1

$group
 [1]  1  2  3  4  5  6  7  8  9 10
```

4.2.2 Parameteric bootstrap sampler

`PBsampler` provides bootstrap samples of four types of lasso estimators with their subgradients. Given point estimates $\tilde{\mu}$ and $\hat{\sigma}^2$, it first generates $y^* = \tilde{\mu} + \varepsilon^*$, where $\varepsilon^* \sim \mathcal{N}_n(0, \hat{\sigma}^2 \mathbf{I}_n)$. Then given λ, $(\hat{\beta}, S)$ is computed by solving (4.1) and (4.2) with y^* in place of y. Note that $(\tilde{\mu}, \hat{\sigma}^2, \lambda)$ determines the sampling distribution of y^* and thus it is denoted by $\pi(\bullet; \tilde{\mu}, \hat{\sigma}^2, \lambda)$.

`PBsampler` can be called as follows:

```
R> PBsampler(X, PE_1, sig2_1, lbd_1, PE_2, sig2_2, lbd_2,
+     weights = rep(1, max(group)), group = 1:ncol(X),
+     niter = 2000, type, PEtype = "coeff", Btype = "gaussian", Y = NULL,
+     parallel = FALSE, ncores = 2L, verbose = FALSE)
```

`X, weights, group, type` arguments work the same as in `lassoFit`.

Up to two sets of $(\tilde{\mu}, \hat{\sigma}^2, \lambda)$ can be supplied with (`PE_1`, `sig2_1`, `lbd_1`) and (`PE_2`, `sig2_2`, `lbd_2`). If (`PE_2`, `sig2_2` and `lbd_2`) is supplied, the sampling distribution becomes

$$\frac{1}{2}\pi(\bullet; \tilde{\mu}_1, \hat{\sigma}_1^2, \lambda_1) + \frac{1}{2}\pi(\bullet; \tilde{\mu}_2, \hat{\sigma}_2^2, \lambda_2),$$

which can be useful for my importance sampler `hdIS` which will be introduced in 4.2.3.

$\tilde{\mu}$ can be supplied in two different ways: (1) Set `PEtype="mu"` and supply $\tilde{\mu}$ to `PE`. (2) Set `PEtype="coeff"` and supply $\tilde{\beta}$ to `PE`; In such a case, $\tilde{\mu}$ is computed by $X\tilde{\beta}$. By default, `PEtype` is set to `"coeff"`.

`Btype` determines the type of bootstrap of which I support two types; `Btype="gaussian"` for Gaussian bootstrap and `Btype="wild"` for the wild bootstrap (Mammen, 1993). Gaussian bootstrap draws ε^* from $\mathcal{N}_n(0, \hat{\sigma}^2\mathbf{I}_n)$. The wild bootstrap generates ε^* with centered residuals and normal random variables; $\varepsilon_i^* = \tilde{e}_i w_i$, where w_i is drawn from $\mathcal{N}(0, \hat{\sigma}^2)$ and $\tilde{e}_i$ is a centered residual which corresponds to y_i. Note that if `Btype = "wild"`, `Y` needs to be supplied for the wild bootstrap in order to compute residuals.

Given residuals, ε^* from the wild bootstrap follows the normal distribution. Therefore, the method of estimator augmentation can be applied to derive the closed form density of $(\Theta, \mathcal{A}) = (\hat{\gamma}_{\mathcal{M}}, S, \mathcal{M})$.

Corollary 9. *When the wild bootstrap is used, f_E from Theorem 2 and 3 is $\mathcal{N}_n(0, \sigma^2 E/n)$, where $E = diag(\tilde{e}_i)_{1:n}$.*

Below is the example code to generate samples from the target distribution $\pi(\bullet; \tilde{\beta} = (0.5, 0.5, 0 \ldots, 0), \hat{\sigma}^2 = 2, \lambda = 0.4)$. It prints out last 10 samples of lasso estimators and corresponding subgradients.

```
R> PBsample <- PBsampler(X = X, PE_1 = c(.5,.5, rep(0,p-2)), sig2_1 = 2,
+      lbd_1 = .5, type = "lasso", niter = 10)
R> PBsample
Last 10 steps of beta's:
        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7] [,8]   [,9]
 [1,] 0.9144 0.0000 0.0000 0.1669 1.8154 0.0000 0.1576    0 0.0000
```

```
 [2,] 0.7449 0.9939  0.0000  0.1066 -1.0372 0.0000 0.0000    0 0.0000
 [3,] 0.1855 0.0000  0.0000  0.5056  0.4892 0.0000 0.0000    0 0.6095
 [4,] 0.1450 0.0000  0.0000  0.2007  0.0000 0.0000 0.0000    0 0.0000
 [5,] 0.0000 0.1592 -0.2010  0.0081  0.0000 0.0000 0.0000    0 0.8482
 [6,] 1.2659 0.0000  0.0000  0.0000  0.1583 0.0000 0.3505    0 0.0000
 [7,] 0.7767 0.0000  0.0000  0.0000 -0.8691 0.0000 0.0000    0 0.1380
 [8,] 0.0978 0.0000  0.0000  0.0000 -0.1025 0.0000 0.4532    0 0.0000
 [9,] 0.5927 0.0000  0.0000 -0.5236  0.0000 0.1841 0.0000    0 0.0000
[10,] 0.0000 0.9762 -0.1808  0.0000  0.0000 0.0000 0.0000    0 0.0000
      [,10]
 [1,]     0
 [2,]     0
 [3,]     0
 [4,]     0
 [5,]     0
 [6,]     0
 [7,]     0
 [8,]     0
 [9,]     0
[10,]     0


last 10 steps of subgradients:
        [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]
 [1,] 1.0000  0.8419  0.4540  1.0000  1.0000 -0.0868  1.0000 -0.3854
 [2,] 1.0000  1.0000 -0.3118  1.0000 -1.0000 -0.6319  0.7750 -0.2039
 [3,] 1.0000  0.3496  0.0405  1.0000  1.0000  0.1074 -0.5712  0.1212
 [4,] 1.0000 -0.2613  0.5411  1.0000 -0.1239  0.0443 -0.1413 -0.0368
 [5,] 0.2318  1.0000 -1.0000  1.0000 -0.4045 -0.8820 -0.2991 -0.0433
 [6,] 1.0000  0.9370  0.2816 -0.0186  1.0000  0.4421  1.0000 -0.0550
 [7,] 1.0000  0.9726 -0.7245  0.3258 -1.0000 -0.2413  0.0775  0.2515
 [8,] 1.0000  0.7716 -0.0393  0.3557 -1.0000 -0.1969  1.0000 -0.1005
```

```
 [9,] 1.0000  0.7156 -0.5455 -1.0000  0.6658  1.0000 -0.5762  0.8030
[10,] 0.0542  1.0000 -1.0000 -0.2744  0.0078 -0.1724 -0.1814  0.2557
         [,9]   [,10]
 [1,]  0.1181 -0.7456
 [2,]  0.6764 -0.6724
 [3,]  1.0000 -0.1663
 [4,]  0.6415 -0.0501
 [5,]  1.0000 -0.3451
 [6,] -0.0642 -0.6742
 [7,]  1.0000 -0.4092
 [8,]  0.2869 -0.5951
 [9,]  0.7757 -0.0717
[10,]  0.4986 -0.2489

Call:
PBsampler(X = X, PE_1 = c(0.5, 0.5, rep(0, p - 2)), sig2_1 = 2,
    lbd_1 = 0.5, niter = 10, type = "lasso")
```

Note that all the argument are stored in the output object.

```
R> str(PBsample)
List of 14
 $ beta   : num [1:10, 1:10] 0.914 0.745 0.185 0.145 0 ...
 $ subgrad: num [1:10, 1:10] 1 1 1 1 0.232 ...
 $ X      : num [1:5, 1:10] -2.1022 -0.0422 -0.4048 -0.1128 1.7971 ...
 $ PE     : num [1:10] 0.5 0.5 0 0 0 0 0 0 0 0
 $ sig2   : num 2
 $ lbd    : num 0.5
 $ weights: num [1:10] 1 1 1 1 1 1 1 1 1 1
 $ group  : int [1:10] 1 2 3 4 5 6 7 8 9 10
 $ type   : chr "lasso"
 $ PEtype : chr "coeff"
```

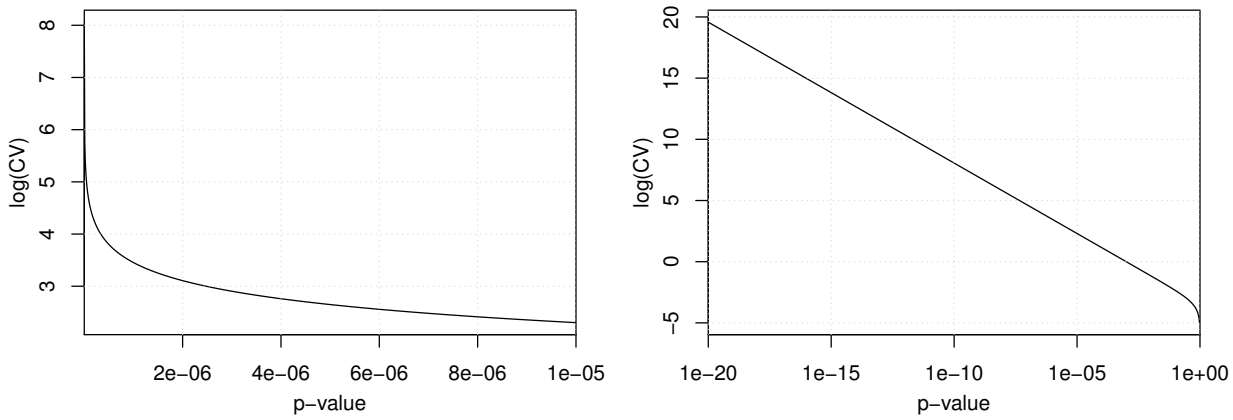

Figure 4.1: log(CV) of the p-value estimator, when the samples were directly drawn from the target distribution. The sample size, N, is set to $1k$. The x-axis represents the true p-value while the y-axis shows the log(CV) of the $\hat{p}$. The x-axis of the right figure is in log-scale.

```
$ Btype   : chr "gaussian"
$ Y       : NULL
$ mixture: logi FALSE
$ call    : language PBsampler(X = X, PE_1 = c(0.5, 0.5, rep(0, p - 2)),
sig2_1 = 2, lbd_1 = 0.5, niter = 10, type = "lasso")
- attr(*, "class")= chr "PB"
```

4.2.3 Importance Sampler

Computing accurate p-value often picks an attention from many scientific studies. Although, using a simple bootstrap method to draw samples from the target distribution directly, call Direct Sampler, is one way to estimate the p-value, when the actual p-value approaches to zero, the estimator from Direct Sampler gets extremely unstable. Figure 4.1 illustrates how the coefficient of variation for the estimated p-value, $CV(\hat{p})$, gets changed by the true p-value. For example, when the true p-value gets smaller than 10^{-10}, CV gets bigger than 8. For such cases when the estimator gets unstable, importance sampling can be a good remedy. See Section 3.3.2 for how importance sampler is constructed.

Sampling from a proposal distribution can be done via `PBsampler`, and let $\{(b^{(t)}, s^{(t)})\}_{1:N}$ be the bootstrap samples generated by `PBsampler`, where N is the total number of samples. Then the

importance weight ω_t can be computed as

$$\omega_t == \frac{\pi(b^{(t)}, s^{(t)}; \tilde{\mu}_{target}, \hat{\sigma}^2_{target}, \lambda_{target})}{\pi(b^{(t)}, s^{(t)}; \tilde{\mu}_{proposal}, \hat{\sigma}^2_{proposal}, \lambda_{proposal})}$$

using the function `hdIS`, which can be called as follows:

```
R> hdIS(PBsample, PETarget, sig2Target, lbdTarget, TsA.method = "default",
+       log = FALSE, parallel = FALSE, ncores = 2L)
```

`PBsample` is an ouput from `PBsampler` which contains samples from a proposal distribution. Parameters of the proposal distribution ($\tilde{\mu}_{proposal}$,$\hat{\sigma}^2_{proposal}$, $\lambda_{proposal}$) is automatically supplied by `PBsample`. Thus, users just need to supply parameters of the target distribution ($\tilde{\mu}_{target}$, $\hat{\sigma}^2_{target}$, λ_{target}) with `PETarget, sig2Target, lbdTarget`. Note that if one sets `PEtype = "coeff"` when he or she runs `PBsampler`, `PETarget` in `hdIS` needs to be $\tilde{\beta}_{target}$ instead of $\tilde{\mu}_{target}$. Again, using the same λ for both the target and the proposal distribution accelerates the computational speed. Set `log = TRUE` for the log scale. See Section 3.3.3 and 3.3.4 for examples.

Let us compute importance weights ω_t using the `PBsampler` result from Section 4.2.2. Assume that the parameters of the proposal distribution are $(\tilde{\beta} = (0, \ldots, 0), \hat{\sigma}^2 = 1, \lambda = 0.5)$. Then one can compute importance weights as follows:

```
> hdIS(PBsample = PBsample, PETarget = rep(0,p), sig2Target = 1,
+       lbdTarget = .5)
 [1] 1.874710e-06 9.031988e-06 2.728438e-03 1.633381e+00 3.555004e-02
 [6] 1.102515e-04 5.982556e-03 3.443649e-01 6.466466e-02 1.223156e-02
```

4.2.4 Metropolis Hastings sampler for lasso estimator

`MHLS` is a Metropolis Hastings sampler which draws lasso estimator $\hat{\beta}$ and its subgradient S conditioned on the lasso selection event, i.e. $\text{supp}(\hat{\beta}) = A$, given point estimates μ_0, σ, and λ. The main function can be called as follows:

```
R> MHLS(X, PE, sig2, lbd, weights = rep(1, ncol(X)), B0, S0,
+       A = which(B0 != 0), tau = rep(1, ncol(X)), niter = 2000, burnin = 0,
```

```
+      PEtype = "coeff", updateS.itv = 1, verbose = FALSE, ...)
```

As in `PBsampler`, `MHLS` requires users to provide `PE`, `sig2`, `lbd`, `weights`, and `PEtype`. `B0` and `S0` are the initial values of $\hat{\beta}$ and S. Supplying `S0` is optional and if it is not provided by the user, `limSolve` package is used to randomly generate `S0` from its feasible reason. The suggested initial values are the output of `lassoFit`. By doing so, Markov chain starts from the stationary distribution and thus no burn-in period is required. `niter` and `burnin` control the number of iteration and the burn-in period, respectively. `tau` controls the standard deviation of proposal distributions. The i-th component of `tau` corresponds to the standard deviation of proposal distribution for $\hat{\beta}_i$. The detailed algorithm is introduced in Section 2.3.2 Algorithm 3.

The output of `MHLS` is a S3 class `MHLS`. The generics functions such as `print`, `summary`, and `plot` are defined. The same example dataset is used to demonetrate `MHLS`.

```
R> MHresult <- MHLS(X = X, PE = lassoEst$B0, sig2 = 1, lbd = 0.5, B0 = lassoEst$B0,
+      S0 = lassoEst$S0, tau = rep(1,p)/2, niter = 5000, burnin = 0)
Last 10 steps of beta's:
        [,1] [,2] [,3] [,4] [,5] [,6] [,7]   [,8] [,9]  [,10]
 [1,] 1.2355    0    0    0    0    0    0 0.9562    0 0.1095
 [2,] 1.4737    0    0    0    0    0    0 0.8126    0 0.1095
 [3,] 1.4737    0    0    0    0    0    0 0.9708    0 0.1230
 [4,] 1.4737    0    0    0    0    0    0 0.9708    0 0.1230
 [5,] 1.0814    0    0    0    0    0    0 0.9708    0 0.0204
 [6,] 1.3636    0    0    0    0    0    0 0.9708    0 0.0204
 [7,] 0.5316    0    0    0    0    0    0 0.8421    0 0.0204
 [8,] 0.9389    0    0    0    0    0    0 0.7380    0 0.0204
 [9,] 0.9389    0    0    0    0    0    0 0.7380    0 0.0204
[10,] 0.9389    0    0    0    0    0    0 0.4246    0 0.0204

last 10 steps of subgradients:
      [,1]   [,2]    [,3]    [,4]   [,5]   [,6]   [,7] [,8]   [,9] [,10]
 [1,]    1 0.7587 -0.0200 -0.0608 0.1226 0.7119 0.9429    1 0.6521     1
 [2,]    1 0.5173 -0.0120 -0.2241 0.1053 0.9948 0.5862    1 0.2101     1
```

```
 [3,]    1 0.5321  0.0273 -0.1079 0.0904 0.9372 0.5970    1 0.2553     1
 [4,]    1 0.7229 -0.0273 -0.1079 0.1234 0.7625 0.8924    1 0.5827     1
 [5,]    1 0.7329 -0.0487 -0.1574 0.1325 0.7721 0.9129    1 0.5912     1
 [6,]    1 0.5840  0.0150 -0.1010 0.0984 0.8871 0.6766    1 0.3455     1
 [7,]    1 0.3827  0.1594  0.1310 0.0289 0.9837 0.3408    1 0.0400     1
 [8,]    1 0.6790  0.2088  0.4894 0.0266 0.5767 0.7622    1 0.6095     1
 [9,]    1 0.4296  0.1870  0.2405 0.0207 0.8993 0.4020    1 0.1392     1
[10,]    1 0.7241  0.2302  0.5811 0.0207 0.5008 0.8223    1 0.7024     1

Acceptance rate:
------------------------------
     beta   subgrad
# Accepted  :   7565   9273
# Moved  :   14997   9998
Acceptance rate  :   0.504   0.927

Call:
MHLS(X = X, PE = lassoEst$B0, sig2 = 1, lbd = 0.5, B0 = lassoEst$B0,
    S0 = lassoEst$S0, tau = rep(1, p)/2, niter = 5000, burnin = 0)
R> summary(MHresult)
$beta
           mean    median       s.d       2.5%     97.5%
 [1,] 0.7589649 0.7557742 0.3299982 0.13019733 1.4244766
 [2,] 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000
 [3,] 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000
 [4,] 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000
 [5,] 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000
 [6,] 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000
 [7,] 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000
 [8,] 0.3069249 0.2600379 0.2298273 0.01167645 0.8421297
 [9,] 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000
```

```
[10,] 0.4015329 0.3496764 0.3033251 0.01424487 1.0969511

$subgradient
            mean      median        s.d       2.5%     97.5%
 [1,] 1.00000000 0.999999997 0.00000000  1.0000000 1.0000000
 [2,] 0.51375213 0.532814981 0.18826983  0.1296041 0.8007815
 [3,] 0.22458260 0.236729660 0.17797797 -0.1441359 0.5203479
 [4,] 0.40521747 0.457410200 0.39274421 -0.4399964 0.9720213
 [5,] 0.01060764 0.008077449 0.07892967 -0.1280815 0.1700554
 [6,] 0.75990969 0.790217581 0.16925422  0.3893181 0.9900324
 [7,] 0.51511813 0.547742320 0.31208919 -0.1256632 0.9756543
 [8,] 0.99999997 0.999999974 0.00000000  1.0000000 1.0000000
 [9,] 0.31167853 0.328255227 0.29859184 -0.2712799 0.8317186
[10,] 1.00000000 1.000000000 0.00000000  1.0000000 1.0000000

attr(,"class")
[1] "summary.MHLS"

R> plot(MHresult, index = 1, skipS = TRUE)
```

`print.MHLS` provides an overview of samples. It shows the last 10 steps of sampled $\hat{\beta}$ and S with thier acceptance rates. `summary.MHLS` summarizes the mean, the median, 2.5% and 97.5% quantiles, and the standard deviation of each $\hat{\beta}_i$ and each S_i for $i \in \mathbb{N}_p$. `plot.MHLS` provides the histogram, the path-plot, the autocorrelation plot for each $\hat{\beta}_i$ and each S_i for $i \in \mathbb{N}_p$. Users can suppress plots for S_i by `skipS=FALSE`. Also By specifying the variable index through `index`, one can choose which $\hat{\beta}_i$ and/or S_i to plot. See Figure 2.4 for example plots generated by `plot.MHLS`.

4.3 Inference

4.3.1 Confidence interval via parametric bootstrap

After drawing samplers with `PBsampler`, consider constructing confidence intervals for $\beta_{0,j}$ for $j \in \mathbb{N}_p$. Denoted bootstrap samples by $\{(b^{(t)}, s^{(t)})\}_{t=1}^N$, where N is the number of iteration. Two ways of constructing confidence intervals is considered in here. The first way is to construct $1-\alpha$ confidence intervals in a naive way. In other words, $\alpha/2$ and $1-\alpha/2$ quantiles of $\{b_i\}_{t=1}^N$ for each $i \in A$ are computed respectively.

The second method is constructing de-biased confidence intervals as in (3.4). Define the de-biased lasso samples by

$$\hat{b}^{(t)} = b^{(t)} + \lambda \hat{\Theta} W s^{(t)},$$

where $\hat{\Theta}$ is computed by `hdi` package. Then quantiles are computed to construct confidence intervals as in Algorithm 6.

Algorithm 6 (Constructing de-biased confidence intervals)**.**

1. Draw $\left\{(b^{(t)}, s^{(t)})\right\}_{t=1}^N$ using `PBsampler`.
2. Plug in $(b^{(t)}, s^{(t)})$ in place of $(\hat{\beta}, S)$ in (3.4) to compute the de-biased estimates $\{\hat{b}^{(t)}\}_{t=1}^N$.
3. Construct a $(1-\alpha)$ confidence interval of $\beta_{0,j}$ with $(q_{j,\alpha/2}, q_{j,1-\alpha/2})$, where $q_{j,\gamma}$ is a γ quantile of $\{\hat{b}_j^{(t)}\}_{t=1}^N$.

These two ways of constructing confidence intervals are implemented in the function `PB.CI` which can be called as follows:

```
R> PB.CI(object, alpha = 0.05, method = "debias", parallel = FALSE,
+     ncores = 2L)
```

`object` is bootstrap samples drawn by `PBsampler`. `alpha` is the significance level which is set to 0.05 by default. `method` controls which method to use to construct confidence intervals. By default, `method` is set to `"debias"` but users can change into `"naive"` to construct confidence intervals in a naive way. See the following code for an example.

```
R> PB.CI(object = PBsample, alpha = 0.05, CImethod = "naive")
              2.5%     97.5%
beta1    0.0000000 1.1868432
beta2    0.0000000 0.9899032
beta3   -0.1964871 0.0000000
beta4   -0.4057522 0.4369896
beta5   -0.9993455 1.5169751
beta6    0.0000000 0.1426440
beta7    0.0000000 0.4300718
beta8    0.0000000 0.0000000
beta9    0.0000000 0.7944642
beta10   0.0000000 0.0000000
```

4.3.2 Post-selection inference

Algorithm 1 and 2 are implemented in `postInference.MHLS` so that one can easily construct confidence sets or confidence intervals of $(X_A^+\mu_0)_B$ for $B \subset A$, where the active set A is selected via lasso. The function can be called as follows:

```
R> postInference.MHLS(LassoEst, Ctype = "CI", X, Y, sig2.hat, tau = rep(1,
+      ncol(X)), alpha = 0.05, MHsamples, target = which(LassoEst$B0 != 0),
+      nChain = 10, Rmethod = "coeff", niterPerChain = 500, parallel = FALSE,
+      ncores = 2L, returnSamples = FALSE, ...)
```

It takes lasso output generated by `lassoFit` as an input argument `LassoEst` so that it can collect all the necessary information from `lassoFit`. `Ctype` can be either `"CI"` or `"CS"` which stand for confidence intervals and confidence sets, respectively. If `Ctype = "CI"`, Algorithm 1 is used to construct confidence intervals of $[X_A^+\mu_0]_j$ for $j \in \mathbb{N}_{|A|}$. Otherwise, a confidence set is constructed by Algorithm 2. In this case, with `target` argument users can specify the subset of A of which they want to construct a confidence set. By default, `target` is set to A so that it constructs a confidence set of $X_A^+\mu_0$. `Rmethod` controls which method to use to construct $\widehat{C}$, a confidence set of μ_0. By letting `Rmethod = "coeff"`, I build $\widehat{C}$ with the center $P_A y$. If `Rmethod = "mu"`, $\widehat{C}$ is built by two-step method by Zhou et al. (2019). See Section 2.2.3 for more detail.

Since Metropolis Hastings sampler is used to draw samples to construct confidence intervals or confidence sets, it can be time-consuming if one wants to construct, say, another confidence set with different choice of `target` but with the same dataset. In such way, during the first run, users can set `returnSamples = TRUE` which makes the output to include samples generated by Metropolis Hastings sampler. Then those samples can be supply back to `postInfernece.MHLS` under `MHsamplers` argument so that sampling step can be omitted. See the example code below to ease the understanding. In the example, first, the confidence intervals for post-selection estimators $(X_A^+\mu_0)_j$ for $j \in A$ is constructed and samples are saved with `returnSamples = TRUE`. Then using the same MH samples, the confidence set for $X_A^+\mu_0$ is constructed.

```
R> P1 <- postInference.MHLS(LassoEst= lassoEst, X = X, Y = Y, sig2.hat = 1,
+      alpha = .05, nChain = 3, niterPerChain = 20, parallel = parallel,
+      returnSamples = TRUE)
> P1$confidenceInterval
$Coeff
[1]  1.1752132 -0.6126123  2.1886790

$CI
 Var LowerBound UpperBound
   1 -0.2554922  2.0876358
   8 -1.4820561 -0.6126123
  10  0.6373363  3.0850577

R> postInference.MHLS(LassoEst = lassoEst, MHsamples = P1$MHsamples,
+      Ctype = "CS", target = which(lassoEst$B0 != 0), X = X, Y = Y)
$confidenceSet
$confidenceSet$Target
[1]  1  8 10

$confidenceSet$Center
[1]  1.1752132 -0.6126123  2.1886790
```

```
$confidenceSet$l2_Radius
   97.5%
1.673057


$call
postInference.MHLS(LassoEst = lassoEst, Ctype = "CS", X = X,
    Y = Y, MHsamples = P1$MHsamples, target = which(lassoEst$B0 !=
        0))
```

4.4 Prostate cancer data example

In this section, the prostate cancer data is used to demonstrate my package which is available from http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/prostate.data. The same data set was used in Chatterjee and Lahiri (2011), Tibshirani et al. (2016) and Liu et al. (2018). The response variable y is log(prostate specific antigen) (*lpsa*). The rest of the variables except *train* are used as explanatory variables, X. The number of samples n is 97 and the number of explanatory variables p is 8.

After loading the data, Guassian quantile transformation is used on each X_i for $i \in \mathbb{N}_8$. Then each X_i and y are centered and scaled so that they have the mean 0 and the standard deviation 1.

```
R> Data <- read.table("http://www-stat.stanford.edu/~tibs/ElemStatLearn/
+      datasets/prostate.data", sep="\t", header = TRUE, row.names = 1)
R> y <- Data$lpsa
R> x <- as.matrix(Data[,1:8])
R> n <- nrow(x)
R> p <- ncol(x)
R> y <- (y - mean(y)) / sd(y)
R> x <- apply(x,2, function(x) qnorm(rank(x) / (n+1)))
R> x <- apply(x,2, function(x) (x-mean(x))/sd(x))
```

First the lasso estimator is fitted using λ chosen by cross-validation with one standard error rule.

```
R> lassoEst <- lassoFit(x,y,type = "lasso",lbd = "cv.1se")
```

```
R> lassoEst
$B0
[1] 0.47102520 0.12280336 0.00000000 0.00000000 0.11694123 0.00000000 0.00000000
[8] 0.01127849

$S0
[1] 1.0000000 1.0000000 0.1946447 0.7234005 1.0000000 0.8031447 0.8608784
[8] 1.0000000

$lbd
[1] 0.160958

$weights
[1] 1 1 1 1 1 1 1 1

$group
[1] 1 2 3 4 5 6 7 8
```

Note that `weight` and `group` is set by $\mathbf{1}_p$ and $\mathbb{N}_p$ by default. λ chosen by one standard error rule is 0.161. The output shows that the active set A is $\{1, 2, 5, 8\}$ which corresponds to log of cancer volume (`lcavol`), log of prostate weight (`lweight`), seminal vesicle invasion (`svi`), and percentage Gleason scores 4 or 5 (`pgg45`), respectively.

Consider testing the significance of $[X_A^+ \mu_0]_j$ for $j \in \mathbb{N}_{|A|}$. Since the selection procedure was done using the same dataset, the confidence interval should be generated under the selection condition $\mathcal{A} = A$. With 95% confidence level, it turns out that the first variable (`lcavol`) is significant while every other one is not significant.

```
R> postInference.MHLS(LassoEst = lassoEst, X=x, Y=y, sig2.hat = 1,
+      Rmethod = "coeff", nChain = 20, niterPerChain = 500, tau = rep(1,p)/4,
+      parallel = TRUE, ncores = 10)
$confidenceInterval
$confidenceInterval$Coeff
```

```
[1] 0.5118173 0.2494862 0.1935013 0.1016934


$confidenceInterval$CI
 Var  LowerBound UpperBound
   1  0.06297174  0.9723036
   2 -0.07152630  0.7988247
   5 -0.27717815  0.3088623
   8 -0.41410973  0.6845259



$call
postInference.MHLS(LassoEst = lassoEst, X = x, Y = y, sig2.hat = 1,
    tau = rep(1, p)/4, nChain = 20, Rmethod = "coeff", niterPerChain = 500,
    parallel = TRUE, ncores = 10)
```

Ignoring the fact that the selection procedure was done with the same dataset, if one just decide to construct the confidence intervals of $[X_A^+\mu_0]_j$ for $j \in \mathbb{N}_{|A|}$ using least squares, the result gets different. Below is the confidence intervals from the least squares. For a fair comparison, assume that $\sigma^2 = 1$ is known.

```
R> A <- which(lassoEst$B0 != 0)
R> lmFit <- lm(y~x[,A]+0)
R> sig2 <- 1
R> cbind(coef(lmFit) - qnorm(0.975)*sqrt(diag(sig2 * solve(t(x[,A])%*%x[,A]))),
+      coef(lmFit) + qnorm(0.975)*sqrt(diag(sig2 * solve(t(x[,A])%*%x[,A]))))
                      [,1]      [,2]
x[, A]lcavol   0.25355018 0.7700845
x[, A]lweight  0.04060231 0.4583700
x[, A]svi     -0.05763687 0.4446395
x[, A]pgg45   -0.13564586 0.3390327
```

The least squares confidence intervals are a lot shorter and consequently make `lcavol` and `lweight` significant.

CHAPTER 5

Summary and Discussion

The dissertation focused on high-dimensional data and proposed a new inference method which can generates confidence intervals and confidence sets given the lasso selection event using methods of estimator augmentation by Zhou (2014). Bootstrap is not a practical method to sample from $[y \mid \mathcal{A}(y) = A]$ due to the tiny probability of the target event. My Metropolis-Hastings algorithm enables drawing samples from $[y \mid \mathcal{A}(y) = A]$ which gives a great flexibility to study inference. Consequently, beyond generating confidence intervals of post-selection estimators, it could also generate confidence sets of the post-selection estimators. With other pre-existing methods, the only way to construct confidence sets is to use the Cartesian product of individual confidence intervals. Confidence intervals generated by my method met desired coverage rates with shorter length compared to those by Lee's method (Lee et al., 2016). Also my confidence sets showed better efficiency in terms of volume when they were compared to the Cartesian product of confidence intervals by Lee's method. I then presented some works to advocate advantages of using group structure in studying high-dimensional inference with strong numerical results. Lastly, I have developed an R pacakge `EAinference` to facilitate applying my methods which mainly relate to estimator augmentation in lasso, group lasso, scaled lasso, and scaled group lasso estimator.

Few possible future works can include an inference method of the post-selection estimators when the variables are selected via group lasso. As discussed in Chapter 3, grouping variables has advantages on dealing with disruption caused by high-correlation or when individual signal strength is not strong enough to detect. Therefore, quantifying uncertainty of each group given the selection procedure can be one interesting topic. Also, instead of focusing on the conditional distribution $[\hat{\beta}_{\mathcal{A}}, S, \mathcal{A} \mid \mathcal{A} = A]$, one can focus on more general cases. One way is separating the active set into strong signals and the weak signals which may improve the performance of my proposed method of constructing confidence intervals and sets. The similar idea was used in Liu et al. (2018). Also in `EAinference` package, Metropolis Hastings algorithm was coded in R. By

switching into C++ code, a significant increase in computational speed can be expected. Due to the complicated constraints imposed on the subgradient of the group lasso estimator, developing a MCMC sampler under the fixed active set for group lasso is a very challenging but an interesting future topic. Lastly, providing rigorous proof

Bibliography

Berk, R., Brown, L., Buja, A., Zhang, K., and Zhao, L. (2013), "Valid post-selection inference," *Ann. Statist.*, 41, 802–837.

Breheny, P. and Huang, J. (2015), "Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors," *Statistics and Computing*, 25, 173–187.

Chatterjee, A. and Lahiri, S. N. (2011), "Bootstrapping lasso estimators," *J. Amer. Statist. Assoc.*, 106, 608–625.

Dezeure, R., Bühlmann, P., Meier, L., and Meinshausen, N. (2015), "High-dimensional Inference: Confidence intervals, p-values and R-Software hdi," *Statistical Science*, 30, 533–558.

Dezeure, R., Bühlmann, P., and Zhang, C.-H. (2017), "High-dimensional simultaneous inference with the bootstrap," *TEST*, 26, 751–758.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004), "Least angle regression," *Ann. Statist.*, 32, 407–499, with discussion, and a rejoinder by the authors.

Fan, J. and Li, R. (2001), "Variable selection via nonconcave penalized likelihood and its oracle properties," *J. Amer. Statist. Assoc.*, 96, 1348–1360.

Hastie, T., Tibshirani, R., and Wainwright, M. (2015), *Statistical learning with sparsity: the lasso and generalizations*, vol. 143 of *Monographs on Statistics and Applied Probability*, CRC Press, Boca Raton, FL.

Huang, J. and Zhang, T. (2010), "The benefit of group sparsity," *The Annals of Statistics*, 38, 1978–2004.

Ivanova, N., Dobrin, R., Lu, R., Kotenko, I., Levorse, J., DeCoste, C., Schafer, X., Lun, Y., and Lemischka, I. R. (2006), "Dissecting self-renewal in stem cells with RNA interference," *Nature*, 442, 533–538.

Javanmard, A. and Montanari, A. (2014), "Confidence Intervals and Hypothesis Testing for High-Dimensional Regression," *Journal of Machine Learning Research*, 15, 2869–2909.

Kivaranovic, D. and Leeb, H. (2018), "Expected length of post-model-selection confidence intervals conditional on polyhedral constraints," *arXiv:1803.01665*.

Lee, J. D., Sun, D. L., Sun, Y., and Taylor, J. E. (2016), "Exact post-selection inference, with application to the lasso," *Ann. Statist.*, 44, 907–927.

Leeb, H. and Pötscher, B. M. (2006), "Can one estimate the conditional distribution of post-model-selection estimators?" *Ann. Statist.*, 34, 2554–2591.

Liu, K., Markovic, J., and Tibshirani, R. (2018), "More powerful post-selection inference, with application to the Lasso," *arXiv:1801.09037*.

Lockhart, R., Taylor, J., Tibshirani, R. J., and Tibshirani, R. (2014), "A significance test for the lasso," *Ann. Statist.*, 42, 413–468.

Mammen, E. (1993), "Bootstrap and wild bootstrap for high-dimensional linear models," *Ann. Statist.*, 21, 255–285.

Martin Sill, Thomas Hielscher, N. B. and Zucknick, M. (2014), "c060: Extended Inference with Lasso and Elastic-Net Regularized Cox and Generalized Linear Models," *Journal of Statistical Software*, 62.

Meinshausen, N. (2015), "Group bound: Confidence intervals for groups of variables in sparse high dimensional regression without assumptions on the design," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77, 923–945.

Meinshausen, N. and Bühlmann, P. (2010), "Stability selection," *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 72, 417–473.

Meinshausen, N., Meier, L., and Bühlmann, P. (2009), "p-Values for High-Dimensional Regression," *Journal of the American Statistical Association*, 104, 1671–1681.

Mitra, R. and Zhang, C.-H. (2016), "The benefit of group sparsity in group inference with de-biased scaled group Lasso," *Electronic Journal of Statistics*, 10, 1829–1873.

Nickl, R. and van de Geer, S. (2013), "Confidence sets in sparse regression," *Ann. Statist.*, 41, 2852–2876.

Pötscher, B. M. (1991), "Effects of model selection on inference," *Econom. Theory*, 7, 163–185.

Sun, T. and Zhang, C.-H. (2012), "Scaled sparse linear regression," *Biometrika*, 99, 879–898.

Taylor, J. and Tibshirani, R. (2018), "Post-selection inference for ℓ_1-penalized likelihood models," *Canad. J. Statist.*, 46, 41–61.

Tian, X. and Taylor, J. (2017), "Asymptotics of selective inference," *Scand. J. Stat.*, 44, 480–499.

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

Tibshirani, R. J. (2013), "The lasso problem and uniqueness," *Electron. J. Stat.*, 7, 1456–1490.

Tibshirani, R. J., Rinaldo, A., Tibshirani, R., and Wasserman, L. (2018), "Uniform asymptotic inference and the bootstrap after model selection," *Ann. Statist.*, 46, 1255–1287.

Tibshirani, R. J., Taylor, J., Lockhart, R., and Tibshirani, R. (2016), "Exact post-selection inference for sequential regression procedures," *J. Amer. Statist. Assoc.*, 111, 600–620.

Tibshirani, R. J., Tibshirani, R., Taylor, J., Loftus, J., and Reid, S. (2017), "selectiveInference: Tools for Post-Selection Inference," *R package version 1.2.4*.

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

van de Geer, S. and Stucky, B. (2016), "χ^2-confidence sets in high-dimensional regression," in *Statistical analysis for high-dimensional data*, Springer, Cham, vol. 11 of *Abel Symp.*, pp. 279–306.

Wasserman, L. and Roeder, K. (2009), "High-dimensional variable selection," *The Annals of Statistics*, 37, 2178–2201.

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

Zhang, C.-H. (2010), "Nearly unbiased variable selection under minimax concave penalty," *Ann. Statist.*, 38, 894–942.

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

Zhang, X. and Cheng, G. (2017), "Simultaneous Inference for High-dimensional Linear Models," *Journal of the American Statistical Association*, 112, 757–768.

Zhou, K., Li, K.-C., and Zhou, Q. (2019), "Honest confidence sets for high-dimensional regression by projection and shrinkage," *arXiv:1902.00535*.

Zhou, Q. (2014), "Monte Carlo simulation for lasso-type problems by estimator augmentation," *J. Amer. Statist. Assoc.*, 109, 1495–1516.

Zhou, Q. and Min, S. (2017a), "Estimator augmentation with applications in high-dimensional group inference," *Electron. J. Statist.*, 11, 3039–3080.

— (2017b), "Uncertainty quantification under group sparsity," *Biometrika*, 104, 613–632.

Zou, H. and Hastie, T. (2005), "Regularization and variable selection via the elastic net," *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 67, 301–320.