Erasmus University Rotterdam
Erasmus School of Economics

Bachelor Thesis - FEB63007

High-dimensional inference: Obtaining p-values in linear models

Variable Selection

Mart Faber
482814

Supervised by: Jesse Hemerik
Second assessor: Stan Koobs

Rotterdam
June 2024

Abstract

The field of high-dimensional inference and variable selection has been filled with research on methods that obtain p-values for variable selection and hypothesis testing. In this paper we focus on variable selection. The low number of observations paired with a large number of variables makes high-dimensionality variable selection difficult to accurately do. Most methods use some form of penalized linear regression, usually the Lasso, as one of the steps for finding accurate p-values for variable selection. Within this piece of research the goal is to compare four variable selection methods: Ridge projection and bias correction, Desparsified Lasso, Multiple sample-splitting and Algorithmic Lasso. The methods are tested on nine variations of simulated data and their performance is evaluated based on their power and family-wise error rate. This research shows that not all methods deal with a lack of sparsity as well as others. Also a trade-off of power compared to the number of false discoveries is visible when comparing results. The Ridge projection and a form of the desparsified Lasso obtain the best results. Which method is better depends on the desire for power and family-wise error rate.

1 Introduction

In 1968 an American robbery trial in California known as People v. Collins, ended with a false conviction based on wrongfully used math and statistics. Through the use of the Multiplication rule, the probability that the accused were not the robbers, with the use of estimated probabilities was presented as a 0.0000083% chance. This caused the jury to return a guilty verdict. The actual meaning of this probability, is the probability that a random defendant has certain traits. The court later on noted that the correct inference would be the probability that no other couple who could have committed the robbery had the same traits as the defendants given that at least one couple had the identified traits. The probability that the defendants were innocent turned out to be over 40% and the guilty verdict was overturned later on. This showed the court that although helpful, complex mathematics should not outweigh the credibility of witnesses and other evidence.

Inference is defined as *"a conclusion reached on the basis of evidence and reasoning"*. As the previous example shows, inference is one of the most important things in statistics. After all, statistics, models and everything else that has followed within the discipline of econometrics is created to convert data into results we can use to make correct inferences about the 'real' world. This paper is an extension of Dezeure et al. (2015) and within this paper, we will take a look at high-dimensional inference for linear models. In this case variable selection specifically, done through estimation of p-values for coefficients. High-dimensional means cases where the number of predictors p is much larger than the sample size n. Many methods have been designed for these high-dimensional cases, where most include the use of Lasso estimation as a tool alongside either a projection, algorithm or heuristic in order to obtain accurate p-values or confidence intervals. One of the main issues with high-dimensionality is the lack of data combined with the large number of variables causing results to be less accurate. Within this research paper, four of these methods will be compared in their ability to provide accurate p-values for the coefficients in a high-dimensional setting, namely: Ridge projection and bias correction, Desparsified Lasso method, Multiple sample-splitting and Lasso projection by Javanmard and Montanari. Their performance will be evaluated based on Power and Family-wise Error Rate (FWER).
Within this piece of research the variable selection methods are tested on simulated datasets. The data is generated with a varying amount of actual non-zero coefficients to test the methods performance in case of sparsity or lack thereof. The correlation between the predictors also varies based on the covariance matrix used to generate the design matrices (X) in the linear model. This is to test which methods can deal with high levels of correlation the best. Besides these assumptions, each method also has its own assumptions. These are all tested for correctness to verify a models performance, as well as to give insight into what it takes for assumptions to hold outside of just the theory. Throughout the results section of this paper, a variation of the Desparsified Lasso method is added to the results. This was done because its performance varied from the regular version while it requires the exact same assumptions and therefore, it is potentially a better performing method. Because of the low amount of observations in the simulated data, the same coefficient matrix and design matrices are used for each method to decrease variance in their performance based on randomness.

In Section 2, a literature review will be done on whatever relevant research has been done in creating methods that are relevant to this paper. Following this, Section 3 will contain a theoretical outline of the methods and methodology used. Section 4 contains a small replication of a part of Dezeure et al. (2015), because this paper is an extension of it. Section 5 contains the information of the simulations done to obtain results while the next section, Section 6 contains these results. Finally, Section 7 contains the conclusion of the entire research.

2 Literature Review

Many of the high-dimensional inference methods are based on the Lasso estimator, proposed by (Tibshirani, 1996). The Lasso method is a regularization technique that performs variable selection and regularization. It does not provide p-values inherently, but helps by enhancing prediction accuracy. It is an extension based on Ridge regression, which was proposed over 25 years earlier (Hoerl and Kennard, 1970). The main difference is that the Lasso also does variable selection. The Lasso was the most attractive regularization method for high-dimensional regression and therefore, slightly adjusted versions were proposed by scientists for various situations. Meinshausen (2007) proposed the relaxed Lasso which produces sparser models with equal or lower prediction loss compared to the regular Lasso estimator for high-dimensional data. The square root Lasso was later created, which achieves near-oracle performance whilst not requiring any knowledge of the standard deviation (Belloni et al., 2011). Another version is the scaled Lasso by Sun and Zhang (2012), which also does not require knowledge of the standard deviation but instead yields an iterative estimation of the noise level by use of a gradient descent algorithm.

Mathematicians were able to use these forms of penalized regression to create methods to obtain p-values for coefficients. We will look at some of these inference methods to calculate p-values, in the high-dimensional setting. In 2009 Roeder and Wasserman (2009) wrote a paper that includes a multi-step variable selection method as extension of the papers Meinshausen (2007) and Meinshausen and Yu (2009). This method uses a heuristic to obtain p-values and confidence intervals. Meinshausen et al. (2012) built upon the idea further, while this idea would later be used to create the Multiple Sample Splitting method in the HDI package by Dezeure et al. (2015). Bühlmann and Mandozzi (2014) provided an empirical comparison of various estimation models in high-dimensional settings, which helped with assumptions required for valid p-values for the Multiple sample-splitting method.
Besides the Multiple Sample-Splitting method, Dezeure et al. (2015) also propose a Desparsified Lasso method for obtaining p-values for coefficients, that was first introduced by Zhang and Zhang (2014). The motivation for this method was that conservative methods, like Berk et al. (2010) published, may not yield sufficiently accurate p-values for common applications with a large number of variables (Zhang and Zhang, 2014). The work of Sun and Zhang (2012) helped by being a method providing an estimate of the noise level in addition to the initial estimator of Beta (Zhang and Zhang, 2014).
The third method for obtaining p-values proposed by Dezeure et al. (2015), is the Ridge projection method. The idea originates from Bühlmann (2013) who wrote a very detailed paper on

the method. The estimation of a Z-matrix in this method is done by the Square Root Lasso as proposed by Belloni et al. (2011), who later wrote a paper on post-selection inference in multiple Z-estimation problems (Belloni et al., 2015).
The mathematical theory behind these methods can be read in the following section.

3 Theory & Methodology

In modeling, one of the simplest yet most effective models is the linear model:

$$Y = X\beta + \varepsilon \tag{1}$$

With X being an $n \times p$ design matrix, β a $p \times 1$ coefficient vector, Y being an $n \times 1$ response vector and ε an $n \times 1$ error vector. Correctly determining which coefficients β_i are non-zero is difficult in a high-dimensional setting, where $p \gg n$. It can be done by constructing p-values for the null-hypotheses:

$$H_{0,i} : \beta_i = 0 \qquad\qquad \forall i, i = 1, 2, ..., p \tag{2}$$

An important set, is the set of non-zero coefficients in a model, which will be denoted as:

$$S_0 = \{j; \beta_j^0 \neq 0, j = 1, ..., p\} \tag{3}$$

The number of non-zero coefficients is denoted as $|S_0|$. This is an important piece of notation for the rest of the paper. If a hat is present ($\hat{S_0}$), it concerns the variables selected by a method. Within this research, four methods that are designed to accurately estimate p-values for these coefficients are compared in performance given various assumptions and violations of these assumptions. This is all done by simulating data and comparing the results through the use of Family-Wise Error Rate(FWER) and Power. Average false discoveries(AVG(V)) is also added in case the FWER does not fully describe the results. Family-wise error rate (FWER) is defined as the probability of making one or more false discoveries, or type I errors when performing multiple hypotheses tests (Goeman and Solari, 2014).
The AVG(V) is similar to the FWER. The difference is that the AVG(V) also takes the number of false discoveries into account. The final measurement is Power. Power is the probability that a test correctly rejects the null hypothesis when the alternative hypothesis is true (Neyman and Pearson, 1933). Here, the power of a method is estimated as the number of correctly identified non-zero coefficients, divided by the total number of non-zero coefficients. It represents the chances of a true positive detection.
When testing multiple hypotheses as in Equation 2, the chances of a false discovery increase and therefore, so does the FWER. In order to control the FWER to be at most size α, a multiple testing correction is done, known as "Holm" or "Bonferroni-Holm" (Holm, 1979). This multiple testing correction is done for each method, except Multiple sample-splitting as it incorporates its own method of multiple testing correction.
Now that we know the measurements that will be used to compare the methods, it is time to introduce the four methods and their assumptions and conditions. Below are the names of the methods and what they will be referred to in the rest of this paper and afterwards.

- Ridge projection and bias correction (RidgeProj)
- Desparsified Lasso method (DesLasso)
- Multiple sample-splitting (Multi Split)
- Lasso projection by Javanmard and Montanari (JM)

Each method will now have its own subsection, containing an explanation of the method as well as providing the assumptions and conditions that have to be met in order for the relevant method to be consistent.

3.1 Ridge projection and bias correction

The first part of the RidgeProj method is simply to estimate the Ridge estimator for β as in Equation 4 below.

$$\hat{\beta}_{\text{Ridge}} = (n^{-1}X^TX + \lambda I)^{-1}n^{-1}X^TY \tag{4}$$

Without additional assumptions on the design matrix X, the system is not identifiable. If $p > n$, then $\text{rank}(X) \leq n < p$ so there are different parameter vectors θ such that $X\beta^0 = X\theta$. By projecting β^0 onto the linear space generated by the n rows of X, the system is identifiable. See Zhang and Zhang (2014) Section 2.1 for more details.
Ridge regression estimates the projected parameter $\theta^0 = P_X\beta^0$, with $P_X = X^T(XX^T)^-X$. Besides the estimation bias by the choice of lambda, the projection bias is then: $B_j = \theta_j^0 - \beta_j^0 = (P_X\beta^0)_j - \beta_j^0 = (P_X)_{jj}\beta_j^0 - \beta_j^0 + \sum_{k\neq j}(P_X)_{jk}\beta_k^0$. Because of these formulas for θ^0 and B_j, we obtain the following formula for the bias corrected β_j:

$$\begin{aligned}
\beta_{\text{corr},j} &= \theta_j^0 - B_j \\
&= (P_X)_{jj}\beta_j^0 - (P_X)_{jj}\beta_j^0 + \beta_j^0\sum_{k\neq j}(P_X)_{jk}\beta_k^0 \\
&= \beta_j^0 - \sum_{k\neq j}(P_X)_{jk}\beta_k^0 \\
\text{So: } \hat{\beta}_{\text{corr},j} &= \hat{\beta}_j - \sum_{k\neq j}(P_X)_{jk}\hat{\beta}_{\text{init};k}
\end{aligned} \tag{5}$$

Where $\hat{\beta}_{\text{init};k}$ is either the scaled Lasso or Cross Validated Lasso estimator, to guarantee a certain estimation accuracy. The Cross-Validated Lasso is used because it often yields good finite sample performance (Dezeure et al., 2015).
Important to note is that the formula as obtained in Equation 5, was obtained by Bühlmann (2013) and differs from the formula found by Dezeure et al. (2015), even though they based their method on the work of Bühlmann (2013). Because Dezeure et al. (2015) did not add much explanation or working out, I followed the work of Bühlmann (2013). However, because the package containing the methods of Dezeure et al. (2015) were used in this paper, the formulas

used by them are added below:

$$\begin{aligned} \text{Bias} &= \sum_{k \neq j} \frac{(P_X)_{jk}}{(P_X)_{jj}} \hat{\beta}_k \\ \text{Corrected Estimator:} &= \frac{\hat{\beta}_{\text{Ridge};j}}{(P_X)_{jj}} - \sum_{k \neq j} \frac{(P_X)_{jk}}{(P_X)_{jj}} \hat{\beta}_k \end{aligned} \tag{6}$$

Now that the estimators are know, the construction of double sided p-values by this method is done by the following formula:

$$P_j = 2(1 - \Phi((a_{n,p;j}(\sigma)|\hat{\beta}_{\text{corr};j}| - \Delta_j)_+)) \tag{7}$$

Where $a_{n,p;j}(\sigma)$ are the normalizing factors bringing the variables Z_j to the $N(0,1)$-scale and Δ_j being constants satisfying a certain upper bound. A simple modification to P_j could provide one-sided p-values if required. A key asymptotic result for obtaining p-values for testing the null-hypotheses $H_{0,j}$ is the following:

$$a_{n,p;j}(\sigma)|\hat{\beta}_{\text{corr};j}| \overset{\text{x}}{\leq} |W| + \Delta_j$$

Where $W \sim N(0,1)$ and $\overset{\text{x}}{\leq}$ denotes stochastically smaller or equal to. Besides this asymptotic result, the following assumptions have to be true for this method:

- Gaussian errors
- Beta-min assumption: $\min_{j \in S_0} |\beta_j^0| \gg \sqrt{|S_0| \log(p)/n}$ for $\hat{S}_{\text{Lasso}}(\lambda)$: In order for variable selection to be accurate, the coefficients need to be sufficiently large.
- Assumption on the minimum of the Covariance matrix of the Ridge estimator multiplied by n:

$$\Omega_{\min}(\lambda) := \min_{j \in \{1,\ldots,p\}} \Omega_{jj}(\lambda) > 0 \tag{8}$$

For other results, proofs and further explanation, we refer to section 2.4-2.5 of Bühlmann (2013).

3.2 Desparsified Lasso Method

The desparsified Lasso method was first introduced by Zhang and Zhang (2014) who focused on constructing confidence intervals for the variables rather than p-values. Hence we focus on the interpretation Dezeure et al. (2015) have on the method.
We first introduce some notation: $X^{(j)}$ is the j-th variable while $X^{(-j)}$ is the collection of all variables except for the j-th. Let $Z^{(j)}$ denote the residuals of regressing $X^{(j)}$ versus $X^{(-j)}$.
If an OLS regression of this is done, we obtain:

$$\hat{\beta}_{\text{OLS};j} = Y^T Z^{(j)} / (X^{(j)})^T Z^{(j)}$$

by linear projection. If $p \gg n$ however, the residuals $Z^{(j)}$ would be equal to zero, causing the projection to be ill-posed.

The idea for this method is to pursue a regularized projection. The first step in this method is instead of an OLS regression, using a Lasso regression of $X^{(j)}$ versus $X^{(-j)}$. Because this regression depends on a regularization parameter λ_j, we get $Z^{(j)} = Z^{(j)}(\lambda_j)$. This Gives the following formula:

$$\frac{Y^t Z^{(j)}}{(X^{(j)})^T Z^{(j)}} = \beta_j^0 + \sum_{k \neq j} P_{jk}\beta_k^0 + \frac{\varepsilon^T Z^{(j)}}{(X^{(j)})^T Z^{(j)}}, \tag{9}$$
$$\text{With } P_{jk} = (X^{(k)})^T Z^{(j)} / (X^{(j)})^T Z^{(j)}$$

If OLS would be used, $P_{jk} = 0$ hence Lasso-residuals are used for $Z^{(j)}$. This does cause a bias to arise and thus a bias-correction has to be made. It is done by plugging in the Lasso estimator $\hat{\beta}$ of the Y versus X regression. The bias-corrected estimator is:

$$\hat{b}_j = \frac{Y^t Z^{(j)}}{(X^{(j)})^T Z^{(j)}} - \sum_{k \neq j} P_{jk}\hat{\beta}_k \tag{10}$$

An important thing to note is that Zhang and Zhang (2014) propose a slightly different tuning parameters to calculate the Z-matrix than Dezeure et al. (2015). This changes the results in terms of power and FWER and because it can easily be implemented through the use of the HDI package, we keep this method in mind for later sections.
Using the result in Equation 10 alongside the results of Equation 9, we get:

$$\sqrt{n}(\hat{\beta}_j - \beta_j^0) = \frac{n^{-1/2}\varepsilon^T Z^{(j)}}{n^{-1}(X^{(j)})^T Z^{(j)}} + \sum_{k \neq j} \sqrt{n} P_{jk}(\beta_k^0 - \hat{\beta}_k)$$

Focusing on the right hand side of the equal sign, the first term has a Gaussian distribution when assuming Gaussian errors. The second term is negligible under the following assumptions:

- The design matrix X has compatibility constant bounded away from zero, and the sparsity is $|S_0| = o(\sqrt{n}/\log(p))$.
- The rows of X are fixed realizations of i.i.d random vectors $\sim N_p(0, \Sigma)$ with the minimal eigenvalue of Σ being bounded away from zero.
- The inverse of Σ is row-sparse with $s_j = \sum_{k \neq j} I((\Sigma^{-1})_{jk} \neq 0) = o(n/log(p))$.

Under these three assumptions, a linear model with fixed design and Gaussian errors has the following properties:

$$\begin{aligned} \sqrt{n}\sigma_\varepsilon^{-1}(\hat{b} - \beta^0) &= W + \Delta, \qquad W \sim N_p(0, \Omega) \\ \Omega_{jk} &= \frac{n(Z^{(j)})^T Z^{(k)}}{[(X^{(j)})^T Z^{(j)}][(X^{(k)})^T Z^{(k)}]} \\ ||\Delta||_\infty &= o_p(1) \end{aligned} \tag{11}$$

The assumptions alongside the properties of Equation 11, have the following asymptotic implications:

$$\sigma_\varepsilon^{-1}\Omega_{jj}^{-1/2}\sqrt{n}(\hat{b}_j - \beta_j^0) \sim N(0, 1) \tag{12}$$

With σ_ε being the error variance as estimated in the Lasso estimation, as in Equation 9. $\Omega_{jj}^{-1/2}$ estimated as in Equation 11. $\sqrt{n}$ the square root of the number of observations. $\hat{\beta}_j$ the Lasso estimator as shown in Equation 9 and β_j^0 the initial estimate of the j-th coefficient in the Lasso projection.
Using the standard normal distribution of the estimated variables, we can find p-values for the null hypotheses: $H_{0,j} : \beta_j = 0$, by using the values for the statistic as in Equation 12 as a z-score for the standard normal distribution and finding the corresponding p-values.

3.3 Multiple Sample-Splitting

The idea behind multiple sample-splitting is quite generic and logical. Given the total collection of variables $\{1, ..., p\}$ and the total sample with indices $\{1, ..., n\}$, split the sample into two equal halves denoted by I_1 and I_2. The idea is to use I_1 for variable selection and I_2 for statistical inference, so constructing p-values. Let $\hat{S}(I_1) \subset \{1, ..., p\}$ be the set of selected variables by the use of I_1.
Next we regress Y_{I_2} on $X_{I_2}^{\hat{S}}$ where $X_{I_2}^{\hat{S}}$ is the collection of X variables selected by the regression using I_1. Following the assumption that the matrix $X_{I_2}^{\hat{S}}$ has full rank $|\hat{S}(I_1)|$, we now obtain (raw) p-values to test: $H_{0,j} : \beta_j^0 = 0$, for $j \in \hat{S}(I_1)$:

$$P_{\text{raw},j} = \begin{cases} P_{\text{t-test},j} \text{ based on } Y_{I_2}, X_{I_2}^{\hat{S}} & \text{if } j \in \hat{S}(I_1) \\ 1 & \text{if } j \notin \hat{S}(I_1) \end{cases} \tag{13}$$

A nice feature of this method, is that an adjustment for multiple testing does not have to be done for all p tests, but it is only needed to control the $|\hat{S}(I_1)|$ tests in I_2. This gives the corrected p-values for variable selection to be:

$$P_{\text{corr},j} = \min(P_{\text{raw},j} \cdot |\hat{S}(I_1)|, 1) \tag{14}$$

Following the steps above is exactly how to run this method. A problem that arises is that the choice for I_1 and I_2 now has a large influence on the outcome in terms of selected variables. This undesired phenomenon is known as a 'p-value lottery' (Dezeure et al., 2015). To overcome this we run the entire method B times, with B being large. This makes it so for every variable j, we obtain B corrected p-values: $P_{\text{corr},j}^{[i]}$ with $j = 1, ..., p$ and $i = 1, ..., B$. Because every half sample is part of the same full sample, there is some dependence between the B p-values for each variable. This method proposes an aggregation for the p-values, using empirical γ-quantiles. The formula for these aggregated p-values becomes:

$$P_j = \min\Big((1 - log(\gamma_{min}))\text{inf}_{\gamma \in (\gamma_{min}, 1)} Q_j(\gamma)\Big), \qquad j = 1, ..., p \tag{15}$$

This method is quite simple and also does not require a lot of assumptions. The first being an approximate screening property, the second an assumption on the cardinality of the number of

selected variables and the third on the reduced design matrix for I_2:

$$\begin{aligned} \hat{S} = \{j; \hat{\beta}_j \neq 0\} \supseteq S_0 \\ |\hat{S}(I_1)| \leq |I_2| \\ \text{rank}(X_{I_2}^{\hat{S}} = |\hat{S}(I_1)| \end{aligned} \tag{16}$$

Important to note is that this method has good stability, but it could require a beta-min assumption on the underlying regression parameters, which is not present in the method (Roeder and Wasserman, 2009). A beta-min assumption for a significance test is counter-intuitive as it aims to estimate the size of these beta's.

3.4 Lasso Projection by Javanmard & Montanari

This method was proposed by Javanmard and Montanari (2014) and is very similar to the method described in Section 3.2. According to Dezeure et al. (2015) the only difference between the two methods, is that in the JM method Z is chosen as the solution of a convex program instead of using the Lasso. This is aimed to relax the sparsity assumption (B3 in Section 3.2) for the design. The way Z is calculated to obtain an unbiased estimator, is done by an algorithm. We will first look at the requirements for this algorithm before presenting it.
The required input for the algorithm is: Response vector y, design matrix X, parameters λ, μ. Where λ is the regularization parameter obtained by the Lasso. If lambda is given, standard Lasso is used. If lambda is not given the square root Lasso is used instead, as proposed in Belloni et al. (2011). The square root Lasso estimator of coefficients and estimates for lambda are defined as the solution to the optimization problem:

$$\begin{aligned} \hat{\beta} \in \arg\min_{\beta \in \mathbb{R}^p} \{\hat{Q}(\beta)\}^{1/2} + \frac{\lambda}{n}||\beta||_1 \\ \lambda = cn^{1/2}\Phi^{-1}(1 - \alpha/2p) \end{aligned} \tag{17}$$

The value for μ is defined as the following:

$$\mu = \frac{1}{\sqrt{n}} \cdot \text{qnorm}\left(1 - \frac{0.1}{p^2}\right)$$

Where *qnorm* is defined as the quantile function of the normal distribution given a certain probability. In this case the probability being $1 - \frac{0.1}{p^2}$.
Now that we have determined how to obtain the required inputs, the algorithm is as follows:

Input: Measurement vector y, design matrix X, parameters λ, μ.
Output: Unbiased estimator $\hat{\theta}^{\mu}$.

1. Let $\hat{\theta}^n = \hat{\theta}^n(Y, X; \lambda)$ be the Lasso estimator as in Equation 17.

2. Set $\hat{\Sigma} \equiv (X^T X)/n$.

3. for $i = 1, 2, ..., p$ do

4. let m_i be the solution of the convex program:

$$\begin{aligned} &\text{minimize } m^T \hat{\Sigma} m \\ &\text{subject to } ||\hat{\Sigma} m - e_i||_\infty \leq \mu \end{aligned} \tag{18}$$

   Where $e_i \in \mathbb{R}^p$ is the vector with one at the i-th position and zero everywhere else.

5. set $M = (m_1, ..., m_p)^T$. If any of the above problems is not feasible, then set $M = I_{p \times p}$.

6. Define the estimator $\hat{\theta}^u$ as follows:

$$\hat{\theta}^u = \hat{\theta}^u(\lambda) + \frac{1}{n} M X^T (Y = X \hat{\theta}^n(\lambda)) \tag{19}$$

After running this algorithm and obtaining estimations of the coefficients $\hat{\theta}_i$ and a formula for $\hat{\Sigma}$, the p-values can be calculated as follows:

$$P_i = 2\left(1 - \Phi\left(\frac{\sqrt{n}|\hat{\theta}_i^u|}{\hat{\sigma}[M\hat{\Sigma}M^T]_{i,i}^{1/2}}\right)\right) \tag{20}$$

Important to note is that this method does not have a built in multiple testing correction method for these p-values and therefore, the *"Holm"* method was added as proposed by Holm (1979). Besides slightly varying from the Desparsified Lasso method of Section 3.2, this method also relaxes assumptions. Within this method, no sparsity assumption on Σ^{-1} is present, the number of selected variables s_j can be as large as p and the sample size has to be at least of size: $n = \Omega((|S_0|\log p)^2)$.

4 Replication

This paper is an extension of Dezeure et al. (2015). In section 6 of that paper, an example of a workflow is presented for the R package. Important to note is that the original paper was published in 2015 so they used a different version of the HDI package, where I used version 0.1.9. Because it is now 2024, my code will run on a newer version of R as well and therefore, results may vary. Below is my replication and its results: After running the Ridge Projection as:

outRidge $\leftarrow$ *ridge.proj*($x =$ *riboflavin$x*, $y =$ *riboflavin$y*)

The next step is to check for corrected p-values with a value of < 0.05:

any(outRidge$pval.corr ≤ 0.05)

The outcome is: "FALSE". Next the 95% confidence intervals of the first 3 predictors are calculated as follows:

confint(outRidge, parm $= 1:3$, *level* $= 0.95$*)*

Below in Table 1 are the values I obtained by running the code compared to the values obtained by Dezeure et al. (2015).

Table 1: Estimated Confidence interval vs. Original values

	My results		Results in Paper	
	Lower	Upper	Lower	Upper
AADK_at	-0.707	1.373	-0.885	1.542
AAPA_at	-1.197	1.131	-1.411	1.228
ABFA_at	-1.181	1.292	-1.394	1.408

A difference arising in the values of the lower and upper bounds makes sense as changes have happened within both the package as well as within the program, but the values being pretty close is also an indication it still works as intended. Because no singular coefficient is significant, a hierarchical structure group test is done:

outClusterGroupBound $\leftarrow$ *clusterGroupBound(x =* *riboflavin$x, y =* *riboflavin$y, alpha* $=$ 0.05)

This results in *outClusterGroupBound* being a list containing a lot of information. The main interest is finding non-zero coefficients and therefore, we logically look for clusters with a lower bound of > 0 with the following code:

significant.cluster.numbers $\leftarrow$ *which(outClusterGroupBound$lowerBound* > 0)

Within the original paper only the root node (node 5) is found as significant, but in my results node 4 is also significant.
Finally, a small adjustment to the original code had to be done to obtain results of the same format. The red square brackets are the part that was cut from my code:

significant.clusters $\leftarrow$ *outClusterGroupBound$members*[[*significant.cluster.numbers*]]

The line of code: *str(significant.clusters)* now grants a list of two. The lists contain the variables in both significant nodes, 4 and 5.

5 Simulations

In order to obtain results within this research, only simulated data was used to test the four methods: *Ridge projection and bias correction* (RidgeProj), *De-sparsified Lasso projection* (DesLasso), *Multiple Sample-Splitting* (Multi Split) and *Lasso projection by Javanmard and Montanari* (JM). Equation 1 at the start of Section 3, contains the explanation of what the chosen values and matrices below entail.
The choice was made for: $n = 50$, $p = 250$ and $nsims = 200$ to create high-dimensional data where $p \gg n$ and to have a reliable number of simulations. Dezeure et al. (2015) used $n = 100, p = 500$ and $nsims = 100$. The decrease in the amount of predictors and observations

is to decrease computational time as computational resources and time was limited for this research. The increase in amount of simulations is have as little randomness affecting the results as possible, which could be the case for some simulations given the low n. The error terms ε are generated $\sim N(0,1)$ as Error Matrix with dimensions: $n \times nsims \rightarrow 50 \times 200$ so that each of the 200 simulations have a unique error term for each of the $n = 50$ observations while not changing the errors from method to method. Three different design matrices(X) are constructed to create simulations with different correlation between the parameters. The reason for the lack of variation in design matrices is that each simulation has different size non-zero coefficients and different error terms and therefore, also changing the design matrix is not essential. Simulating a new design matrix would have been nice but to prevent crashes of the program when dealing with very large matrices, as there were already many crashes without the variation in design matrices I decided against it. The design matrices X are generated $\sim N_p(0, \Sigma)$ with Σ being either of the following three covariance matrices:

$$\begin{aligned} \text{Toeplitz: } \Sigma_{j,k} &= 0.9^{|j-k|} && \forall j \neq k \\ \text{Equal Correlation.1: } \Sigma_{j,k} &= 0.1 && \forall j \neq k \\ \text{Equal Correlation.8: } \Sigma_{j,k} &= 0.8 && \forall j \neq k \\ \Sigma_{j,k} &= 1 && \forall j = k \end{aligned} \tag{21}$$

The Toeplitz covariance matrix is used as a somewhat irregular covariance matrix for a more realistic performance. The choice for an equal correlation of 0.1 is to check performance when little correlation is present and equal correlation of 0.8 is to check performance when there is high correlation between variables. An important thing to note is that these design matrices X are $n \times p$ and will be the exact same for each of the $nsims = 200$ simulations.
The only thing left is to generate the coefficient matrix β so we can find response vector Y. An important choice is the set of non-zero variables, S_0:

$$S_0 = \{j; \beta_j^0 \neq 0, j = 1, ..., p\} \tag{22}$$

And its cardinality $|S_0|$. Due to the required assumption of sparsity for most methods, sparse options as well as less sparse options are used, namely: $|S_0| = \{2, 5, 10\}$. The chosen non-zero coefficients are generated as: $\sim U(0, 5)$, any other values remain zero. For each of the 200 simulations a 250×1 coefficient vector is created this way.
Now following Equation 1 we generate the response vector:
$Y = X\beta + \varepsilon$
Obtaining 200 vectors Y with dimensions 50×1, for each of the simulation options. A total of 9 simulation combinations of $|S_0|$ and X can be created as seen below in Table 2:

Table 2: Choices for $|S_0|$ and covariance matrix for generating X for the simulations

Cardinality($|S_0|$)	Design Matrix(X)
2	Toeplitz
5	Equal Correlation 0.1
10	Equal Correlation 0.8

For each of these 9 options, all four methods are implemented. For the use of each method, the relevant design matrix X and the associated response vector Y are used. For every method the p-values are adjusted for multiple testing by way of Bonferroni-Holm (Holm, 1979), except Multi Split as it has its own method of correction because of the aggregation of p-values. For the RidgeProj, Multi Split and DesLasso methods, betainit is estimated by the cross-validated Lasso. For the DesLasso the Z-matrix found by the first simulation is used for the other 199 as well, which is possible because it only depends on X (Dezeure et al., 2015). The Multi Split method uses $B = 100$, which in hindsight could have been 50 to decrease run time as this yielded approximately the same results. The JM method uses $\alpha = 0.05$. Other specifications regarding the methods can be found in their respective subsections of Section 3.
After running these simulations the Power, FWER and Number of false discoveries (V) are saved as these statistics will determine the performance of each method. The results can be seen in Section 6 where the performance of the methods is evaluated. In Appendix A an overview of the approximate time it took to run each method and the specifications of the program and pc it was done on is given.

6 Results

Within this section the results of the simulations are displayed in tables. Firstly, an overview of the required assumptions per method is provided and if they hold or not. Then an overall performance evaluation is done to determine which method outperforms the others based on power and FWER. An important thing to note is that the desparsified Lasso method has various ways of calculating the Z-matrix. As stated in Section 3.2, the method proposed by Zhang and Zhang (2014) uses a slightly different way of choosing tuning parameters to compute Z. This method was also added to the results section as it obtained results different to the other desparsified Lasso methods results. The method does not require assumptions different from the DesLasso and because it was proposed by Zhang & Zhang, the method will be denoted as Z&Z.

6.1 Assumptions

Important to note before this section containing assumptions, is that these methods are only proven asymptotically. However, these assumptions and violations of them could offer an explanation for potential bad performance.
All the methods make use of a sparsity assumption through the Lasso. This level of assumed sparsity varies per paper as some assume it at a certain level while others assume it below a

certain level or even between two levels. The two main formulas found for sparsity are:

$$|S_0| = o(\sqrt{n/\log(p)}) \approx 3 \tag{23}$$
$$|S_0| = o(n/\log(p)) \approx 9$$

Therefore, the sparsity conditions hold for our simulations with $|S_0| = 2$, if $|S_0| = 5$ the weak condition holds but the other does not, while we know neither holds for $|S_0| = 10$. This is why these cardinalities were chosen, to have a cardinality below the lowest condition, one inbetween both conditions and one above the higher condition. An argument could be given for adding another cardinality of 15 or 20 for example, but due to limited time this was not done.
Another assumption is Gaussian errors, which is an assumption that holds by definition as the errors are simulated as $\varepsilon_i \sim N(0, 1)$.

RidgeProj

R1 Beta-min assumption: $\min_{j \in S_0} |\beta_j^0| \gg \sqrt{|S_0| \log(p)/n}$ for $\hat{S}_{\text{Lasso}}(\lambda)$: In order for variable selection to be accurate, the coefficients need to be sufficiently large.

R2 Assumption on the minimum of the Covariance matrix of the Ridge estimator multiplied by n:

$$\Omega_{\min}(\lambda) := \min_{j \in \{1,\ldots,p\}} \Omega_{jj}(\lambda) > 0$$

For R1 the minimum size the β-coefficients need to have for it to hold, is: $\sqrt{|S_0| \log(p)/n} \approx 0.47, 0.74$ and 1.05 for $|S_0| = 2, 5$ and 10 respectively. Because the size of the non-zero coefficients are simulated as: $\sim U[0, 5]$ the beta-min assumption will not hold for some generated values and therefore, this method will make some errors.
R2 depends per simulation. Unfortunate is that the method in R does not allow for retrieval of the covariance matrix of the Ridge estimation and therefore, this assumption can not be tested and as such, remains a real assumption.

DesLasso

L1. The design matrix X has compatibility constant bounded away from zero, and the sparsity is $|S_0| = o(\sqrt{n}/\log(p))$.

L2. The rows of X are fixed realizations of i.i.d random vectors $\sim N_p(0, \Sigma)$ with the minimal eigenvalue of Σ being bounded away from zero.

L3. The inverse of Σ is row-sparse with $s_j = \sum_{k \neq j} I((\Sigma^{-1})_{jk} \neq 0) = o(n/log(p))$.

For L1 the compatibility constant can have a different value for each simulation done on this method. For this reason the calculation was done for each method. What the compatibility assumption entails exactly can be found in Appendix B. The condition holds if $\phi > 0$ and Table 3 below shows the minimal required values for ϕ.

Table 3: Minimal values of ϕ for each choice of design matrix and cardinality of S_0

	Toeplitz	Equal Corr. 0.1	Equal Corr. 0.8
$\lvert S_0\rvert = 2$	0.77	0.83	1.20
$\lvert S_0\rvert = 5$	0.86	0.95	1.82
$\lvert S_0\rvert = 10$	1.00	1.11	2.67

For L2 we see that the simulations of the X-matrices are $\sim N_p(0, \Sigma)$ with 3 options for the Sigma matrices. Their minimal eigenvalues are: $0.053, 0.900$ and 0.200 for the toeplitz, equal correlation 0.1 and equal correlation 0.8 covariance matrices respectively and therefore, this assumption holds.
L3 provides a problem. The inverses of each of the three covariance matrices do not have a single row containing only zeros. Therefore the row sparsity is equal to $p = 250$ and this assumption does not hold, as $o(n/log(p)) = o(50/log(250)) \approx 9 \neq 250$.

Multi Split

M1. $|\hat{S}(I_1)| \leq |I_2|$

M2. $\text{rank}(X^{\hat{S}}_{I_2}) = |\hat{S}(I_1)|$

Because the sample is split in half using this method, we have $|I_2| = 25$ so M1 holds. For M2 it depends on the simulation results and design matrices. The second assumption requires the matrix $X^{\hat{S}}_{I_2}$, which is the design matrix only including the rows containing the rows of variables selected. It is assumed to have full rank which would mean that all rows are non-zero. Checking for the selected variables in each simulation is a lot of work and therefore, a different approach was done. Firstly, for each design matrix the rank was tested and they are all equal to 50, which is the number of rows so i has full rank. This means that the assumption does not hold if over 50 variables are selected. The Multi Split method ends up never selecting anywhere near 50 variables as can be seen in Section 6 and therefore, this assumption holds.

JM

J1. $|\hat{S}| \leq p$

J1 holds, as $|S_0| \in \{2, 5, 10\}$ and $p = 250$. Other than this the paper states that essentially no assumptions beyond the standard conditions for high-dimensional consistency are required (Javanmard and Montanari, 2014).

6.2 Results Toeplitz

A very notable difference in performance are the high values of the JM method. This shows that the method has less assumptions regarding selection than the others, causing it to select more variables than the other methods. Unfortunately, many of the variables in the set of selected variables, $x \in \hat{S}$ are equal to zero in the actual model, so $x \notin S_0$, giving a high level of false

discoveries throughout all results. These results are not in line with the results of Javanmard and Montanari (2014), but Dezeure et al. (2015) also obtained some less than optimal results using this method. Dezeure et al. (2015) also mentioned their own implementation of the method got better results than the original code that was used for this research, explaining some of the difference. Given the bad performance of this method over all simulations, its performance is not evaluated as it does not perform near as well as the other methods. Besides this method the Z&Z method is also not elaborated on as much, as it is the same method as the DesLasso but with other parameters. It has lower FWER and average false discoveries for all design matrices, while also obtaining lower power for all.

With a high level of sparsity, the methods all perform quite evenly, where the Ridge projection favors trading some of its power for a lower Family-Wise Error Rate. Depending on the desired Power-FWER trade-off, either the Multi Split or RidgeProj method perform best.

When $|S_0| = 5$ we see a large decrease in power for the Multi Split method, to control its FWER. The DesLasso method does not decrease in power much but does increase in FWER to keep its power. The RidgeProj method holds well and outperforms the other three methods as it manages to keep its FWER low, while retaining most of its power.

For $|S_0| = 10$ the methods all perform quite badly as the sparsity assumption does not hold. The Multi Split method ends up with very low power to keep its FWER low while the DesLasso method has high FWER and even higher Avg(V) in order to keep the power high. The RidgeProj method is in between and still obtains relatively good results compared to the other methods.

Overall the Ridge Projection method performs the best for the Toeplitz matrix. Based on the desired trade-off either the Z&Z, DesLasso or Multi Split method could be used too but the less sparse the model is, the worse the performance of Multi Split becomes.

Table 4: Power, Family-wise Error Rate and Average number of false discoveries

	Toeplitz	RidgeProj	DesLasso	Multi Split	JM	Z&Z
	Avg(Power)	0.673	0.780	0.788	0.893	0.765
$\|S_0\| = 2$	FWER	0.010	0.070	0.055	0.910	0.045
	Avg(V)	0.010	0.075	0.055	2.565	0.045
	Avg(Power)	0.663	0.757	0.596	0.871	0.740
$\|S_0\| = 5$	FWER	0.030	0.215	0.055	0.995	0.145
	Avg(V)	0.030	0.240	0.055	5.770	0.145
	Avg(Power)	0.559	0.641	0.084	0.685	0.618
$\|S_0\| = 10$	FWER	0.120	0.440	0.025	1.000	0.305
	Avg(V)	0.130	0.630	0.025	9.755	0.395

6.3 Results Equal Correlation of 0.1

For the simulations where all the variables have equal correlation of 0.1, we see very good performances by all methods except the JM method. For $|S_0| = 2$ the Multi Split method

outperforms RidgeProj as it has higher power and both have a FWER of zero. While the DesLasso does not obtain much more power, it does have some false discoveries.
At $|S_0| = 5$ we see the RidgeProj and Multi Split methods having a FWER of zero again, while maintaining quite high power. RidgeProj now outperforms the Multi Split method, while DesLasso holds a slightly higher power but increases its FWER compared to lower $|S_0|$.
With the cardinality increasing to 10, the Multi Split method again has a FWER of zero but trades in a lot of its power causing it to perform very poorly. The DesLasso method holds quite high power but not much higher than the RidgeProj method, while it obtains an FWER of over 0.3. The RidgeProj method now has some false discoveries, but still holds high power and performs best again.

Table 5: Power, Family-wise Error Rate and Average number of false discoveries

	Equal Corr. 0.1	RidgeProj	DesLasso	Multi Split	JM	Z&Z
	Avg(Power)	0.815	0.888	0.828	0.920	0.855
$\|S_0\| = 2$	FWER	0.000	0.100	0.000	0.715	0.025
	Avg(V)	0.000	0.110	0.000	1.585	0.025
	Avg(Power)	0.814	0.862	0.726	0.898	0.841
$\|S_0\| = 5$	FWER	0.000	0.205	0.000	0.885	0.045
	Avg(V)	0.000	0.250	0.000	3.625	0.045
	Avg(Power)	0.710	0.781	0.146	0.815	0.748
$\|S_0\| = 10$	FWER	0.045	0.330	0.000	0.990	0.105
	Avg(V)	0.055	0.513	0.000	7.590	0.130

6.4 Results Equal Correlation of 0.8

A first disclaimer is that for this design matrix, for some simulation the JM method obtained an error of: #[1] "Warning: Noise estimate problematic", which is explained in Appendix C.
Again for $|S_0| = 2$ the performance of all methods except for JM is quite good, with DesLasso having more power but a non-zero FWER. The performance of the four methods is nearly equal here as they just vary in Power-FWER trade-off again.
For $|S_0| = 5$ the performance of Multi Split drops off a lot as the power approximately halves. DesLasso has more power than RidgeProj but also has a higher FWER. Given how little mistakes RidgeProj makes, it still performs best.
This time for $|S_0| = 10$ the RidgeProj method loses a lot of its power to keep a small FWER. Multi Split barely selects any variables now and the DesLasso does not do bad. The power of the DesLasso has decreased but the FWER barely increased and over half of the non-zero coefficients are selected. This time the trade-off in the Lasso methods that Z&Z shows is quite good, as it does not have a high FWER like DesLasso, but still has power that is very close to the power of DesLasso.

Table 6: Power, Family-wise Error Rate and Average number of false discoveries

	Equal Corr. 0.8	RidgeProj	DesLasso	Multi Split	JM	Z&Z
	Avg(Power)	0.663	0.763	0.655	0.845	0.730
$\|S_0\| = 2$	FWER	0.000	0.075	0.000	0.970	0.040
	Avg(V)	0.000	0.075	0.000	3.745	0.040
	Avg(Power)	0.638	0.743	0.323	0.840	0.707
$\|S_0\| = 5$	FWER	0.010	0.125	0.000	1.000	0.040
	Avg(V)	0.010	0.130	0.000	9.375	0.040
	Avg(Power)	0.426	0.557	0.010	0.662	0.513
$\|S_0\| = 10$	FWER	0.005	0.130	0.000	1.000	0.035
	Avg(V)	0.005	0.150	0.000	15.52	0.040

6.5 Overall Performance

After the previous subsections, a Power/FWER trade-off is visible where each time any of the methods, excluding JM, have higher power than another it comes with a higher family-wise error rate. As a researcher you could choose your method based on your desired trade-off. Important is to know how correlated your variables are and what their sparsity is, as this has a very high impact on the methods performance.

Another part of performance that was not mentioned a lot before, is running time. As researchers we prefer methods that do not take up a lot of time over methods that take a long time. In Table 7 in Appendix A, an overview of the approximate time it takes to run 200 simulations of each method is presented. The table does not provide much help for the Multi Split and JM method, as they both performed the worst in terms of power and FWER, and also take the longest to run. The conclusion still stands that the RidgeProj method and the Z&Z Lasso perform the best.

7 Conclusion

The main goal of this paper is to compare four methods of variable selection for high-dimensional linear models: *Ridge projection & bias correction*, *desparsified Lasso projection*, *Multiple sample-splitting* and *Lasso projection by Javanmard & Montanari*. Their performance is measured by their average Power, Family-wise Error Rate and Average false discoveries. Simulated data was used with variation in the correlation between the variables, size of the coefficients and cardinality of non-zero coefficients.

The *Lasso projection by Javanmard and Montanari* ended up performing so poorly over all simulations that we disregard the results it obtained. Because the *desparsified Lasso projection* had a lot of false discoveries along with high power, a slightly adjusted version of the method was also added which trades-off some of its power for less false discoveries, named the *Z&Z Lasso* (Zhang and Zhang, 2014).

The methods now resulted in a clearly visible trade-off between power and family-wise error rate in each set of simulations. The *Multiple sample-splitting* method having the least false discoveries and often the lowest power along with it. The *desparsified Lasso projection* being the opposite, obtaining high power in exchange for a high false positive rate. In the middle are the *Ridge Projection* and the *Z&Z Lasso*. These two methods appear to perform the best, with the Z&Z Lasso scoring higher on power and family-wise error rate than the Ridge projection estimator. These methods also both performed in a stable way when the sparsity assumption was violated, where the *Multiple sample-splitting* method obtains very low power when less sparsity is present. As a researcher your choice of method depends on both the data and your choice. If you prefer more power in the power/FWER trade-off then a decent choice is the *Z&Z Lasso*. Whereas the other way around the *Ridge projection* would be a good choice. Given the assumptions on these methods, having information on your data would also greatly help out in making the correct choice.

After having done a lot of work on these methods. A future recommendation for research would be to have more options for design matrices and coefficient matrices. Generating a new design matrix for each simulation is also a correct step to make. The correlation between variables can greatly impact the performances of these methods and setting the sizes of the non-zero coefficients to be sufficiently small or large instead of generating them as a random number between zero and five could help test violation of the beta-min assumption. The error vectors not being generated as Gaussian errors would violate an assumption for these methods too, which could be interesting to evaluate performance when assumptions do not hold. In conclusion there is enough to still be researched within this topic and I will personally be looking forward to new discoveries in it.

References

A. Belloni, V. Chernozhukov, and L. Wang. Square-root lasso: pivotal recovery of sparse signals via conic programming. *Biometrika*, 98:791–806, 2011.

A. Belloni, V. Chernozhukov, and K. Kato. Uniform post-selection inference for least absolute deviation regression and other z-estimation problems. *Biometrika*, 102:77–94, 2015.

R. Berk, L. Brown, and L. Zhao. Statistical inference after model selection. *Journal of Quantitative Criminology*, 26:217–236, 2010.

P. Bühlmann. Statistical significance in high-dimensional linear models. *Bernoulli*, 19:1212–1242, 2013.

P. Bühlmann and J. Mandozzi. High-dimensional variable screening and bias in subsequent inference, with an empirical comparison. *Computational Statistics*, 29:407–430, 2014.

R. Dezeure, P. Bühlmann, L. Meier, and N. Meinshausen. High-dimensional inference: Confidence intervals, p-values and r-software hdi. *Statistical Science*, 30:533–558, 2015.

J. Goeman and A. Solari. Multiple hypothesis testing in genomics. *Statistics in Medicine*, 33: 1946–1978, 2014.

A. Hoerl and R. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42:80–86, 1970.

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

A. Javanmard and A. Montanari. Confidence intervals and hypothesis testing for high-dimensional regression. *Journal of Machine Learning Research*, 15:2869–2909, 2014.

N. Meinshausen. Relaxed lasso. *Computational Statistics & Data Analysis*, 52:374–393, 2007.

N. Meinshausen and B. Yu. Lasso type recovery of sparse representations of high-dimensional data. *Annals of Statistics*, 37:246–270, 2009.

N. Meinshausen, L. Meier, and P. Bühlmann. p-values for high-dimensional regression. *Journal of the American Statistical Association*, 104:1671–1681, 2012.

J. Neyman and E. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London*, 231:694–706, 1933.

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

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

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58:267–288, 1996.

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

Appendix

A Simulation Data

The simulations were done on a Lenovo Legion Y540-15IRH. It contains an Intel Core i7-9750H CPU processor with 6 cores. It has a clock speed of 2.6GHz maxing out at 4.5GHz. It has 16GB of ram with a Double Data Rate(DDR) of 2.666MT/s. The simulations were done in Rstudio on R4.4.0. Below is an overview of the approximate time it took for each method to do 200 simulations. A disclaimer is that it is the time it took when using a design matrix generated with a Toeplitz covariance matrix and a cardinality of $|S_0| = 2$ and therefore, others could take longer.

Table 7: Approximate time required for 200 simulations

Ridgeproj	140s
DesLasso	165s
Multi Split	1120s
JM	1160s
ZZ	170s

B Compatibility Condition

Consider a fixed design matrix X, the compatibility condition holds if for some $\phi > 0$ and all β satisfying: $||\beta_{S_0^c}||_1 \leq 3||\beta_{S_0}||_1$,

$$||\beta_{S_0}||_1^2 \leq \beta^T \hat{\Sigma} \beta s_0 / \phi_0^2, \qquad \hat{\Sigma} = n^{-1} X^T X$$

In this piece of research, $||\beta_{S_0^c}||_1$ and $||\beta_{S_0}||_1$ are the sum of the real sizes of all coefficients that are in the active set and its complement. Given the simulated data, every value of $\beta_{S_0^c}$ is equal to 0 and therefore, this is true and we can check if the condition holds.
We need to create a few things in Rstudio for calculation. Firstly, the left side of the equation, which is the square of the sum of values of the coefficient matrices used. Next we need $\hat{\Sigma} = n^{-1} X^T X$, which we make for each design matrix. $s_0 = |S_0| \in \{2, 5, 10\}$ and $\phi > 0$ is the compatibility constant. This being larger than 0 can be tested by changing the equation create:

$$\phi_0^2 \leq \frac{\beta^T \hat{\Sigma} \beta |S_0|}{||\beta_{S_0}||_1^2}$$

Where we obtain the minimal value for ϕ and as long as its larger than zero the *Compatibility Condition* holds. This was done for each of the 9 simulation options and the results of it are in

Section 6.1.

C Error JM method

For the design matrix X with equal correlation of 0.8, for some simulations the JM method gave the error: #[1] "Warning: Noise estimate problematic". This error is given when:

$$\frac{\hat{\sigma_0}}{\hat{\sigma_1}} < 0.5 \text{ OR } \frac{\hat{\sigma_0}}{\hat{\sigma_1}} > 2 \tag{24}$$

Where $\hat{\sigma_1}$ is the estimated standard deviation of the noise and $\hat{\sigma_0}$ is the Median Absolute Deviation (MAD) of *ynorm*, where *ynorm* is defined as:

$$\text{ynorm } = \sqrt{n} \cdot \frac{\hat{y_h}}{\sqrt{\text{diag}(A)}} \tag{25}$$

Where A is defined as $M\hat{\Sigma}M^T$, which is a measure that is minimized to control the variance of the estimation errors. The error arises whenever the ratio of the Median Absolute Deviation to the actual values becomes too small or large. There are some differences in the two measures that are compared but in large lines the error comes down to the measure of dispersion of the data being too large or too small. The result of this error is that the error variance is not controlled by the method and therefore, estimation of coefficients can be biased.
For a more in depth look at this problem, see Javanmard and Montanari (2014) for more information on what the issue entails exactly.