



COLUMBIA UNIVERSITY IRVING MEDICAL CENTER

P8160 GROUP PROJECT REPORT

Project 2: Design a simulation study to compare variable selection methods

Advisor:

Prof. Ying Wei

TA: Baoyi Shi

Group Members:

Jasmine Zhang(yz4720),

Zhuodiao Kuang(zk2275),

Peng Su(ps3395)

Department of Biostatistics

March 3, 2024

1 Introduction

Variable selection[2] is inevitable when linear model is fitted with hundreds of thousand of covariates when dealing with real world problems. In order to balance model fitness and complexity, it is important to include the variables with significant impact on the response in the final model and exclude ones that are meaningless. Under such high dimensional circumstances, many problems can occur [5]. When variable selection methods decide on final models, it is possible for them to overlook the variables with weaker coefficients because the calculation algorithm might classify those variables as insignificant. Collinearity also happens when correlation occurs between variables, which can lead to overall uncertainty of the model [1].

With these potential problems, it is natural to wonder how traditional variable selection methods perform in regarding weak predictors. In order to investigate the study question further, this study analyzes method performance in accurately identifying different types of predictors and elucidates the consequences of omitting weak predictors on the accuracy of the model's parameter estimates.

2 Methods to be studied

Step-wise Forward Selection and LASSO[6] regression are both powerful tools in comparative analysis due to their variable selection capabilities[3], while they operate under different principles and assumptions. Step-wise Forward Selection starts with the intercept only model and adds one predictor at a time to the model based on the criteria that such predictor reduces the AIC of the model the most. As a contrast, LASSO Regression fits the full model first and then exclude some predictors by shrinking their coefficients to 0 through a tuning parameter. In both cases, different types of data predictors can affect variable selection process and impact parameter estimation for selected predictors in the final model. In particular, when data contains many weak variables with lower coefficient compared with strong predictors, including or excluding them may influence the estimated parameter of strong covariance, which have a stronger association with the response variable.

In order to design a simulation study, variables are classified into three categories for method analysis. The first category is strong predictors, where the absolute value of their coefficient is greater than a threshold defined by $c\sqrt{\log(p)/n}$. The second category is weak predictors[4] that are correlated to other predictors. The absolute value of their coefficient is smaller or equal to $c\sqrt{\log(p)/n}$ while $\text{corr}(X_j, X_{j'}) \neq 0$. The third category is weak and independent predictors, with absolute value of coefficient smaller or equal to $c\sqrt{\log(p)/n}$ and $\text{corr}(X_j, X_{j'}) = 0$. Refer to Appendix for full definition.

Through this analysis, strengths and limitations of Step-wise Forward Selection and LASSO Regression will be investigated, offering guidance on their application in statistical modeling within high-dimensional data environments.

3 Methods for generating data

The process of random number generation algorithm starts with drawing data from standard normal distribution. Correlation with other predictors are established through an additive relationship between predictor values. The predictors are then multiplied to corresponding coefficients for a weak or strong signal, as defined earlier. In the context of the Group 1 simulation, the true beta coefficients (β_{true}) are initially configured to remain constant for

a singular iteration of the test. However, to facilitate a robust statistical analysis that includes the computation of the mean squared error (MSE) and other relevant metrics, it is recommended to diversify the generation of β_{true} values across subsequent iterations. These variations should adhere to the established definitions' criteria to ensure consistency and validity in the simulation's framework. Notably, within the visualization presented, the c -value, which serves as a critical threshold delineating the signal strength in the predefined categories, is uniformly set to 1. This standardization is pivotal for maintaining a controlled environment in which the performance of the discussed methodologies can be accurately assessed and compared, particularly in their efficacy in identifying and differentiating between predictor variables of varying significance.

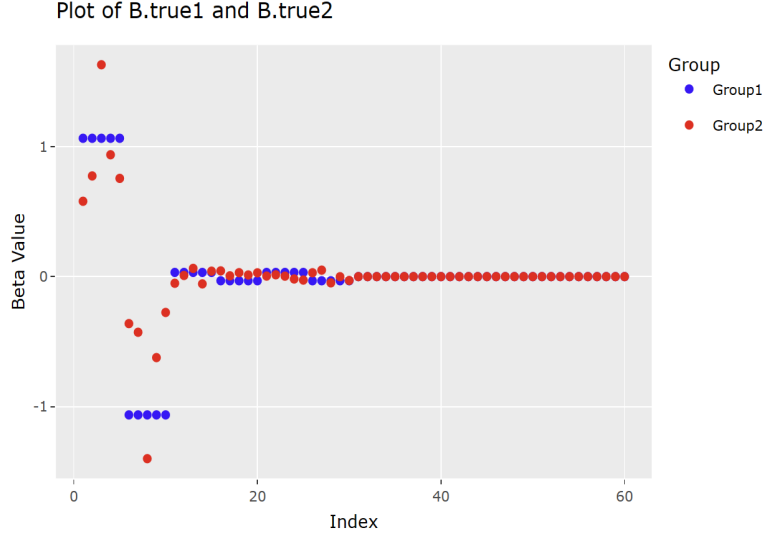


Figure 1: Two groups of true betas generated for the simulation

4 Design of simulation settings and performance measures

The simulation assesses the methods' sensitivity and specificity across predictor types, in addition to the mean squared error (MSE) of model fits. Sensitivity refers to the correct identification rate of true positives and specificity to the true negatives, shedding light on each method's precision in selecting pertinent predictors and dismissing inapplicable ones.

In order to investigate the impact of missing weak predictors on model parameter prediction, a simulation study was designed. LASSO and forward selection were applied as a variable selection method and the results were collected based on 100 repeat. Firstly, for each loop, the true data and corresponding parameters for each variable were generated by the random method introduced before and then fitted to a LASSO and forward selection model. As the simulation data contained strong, weak but correlated and weak, and independent variables, then secondly, by extracting the selected predictors and their coefficients that were included in the final two models, the number of three different types of predictors were collected.

In addition, by comparing the model estimated coefficients for different predictors with the true parameters, the mean square error (MSE) for the estimated coefficients associated with strong predictors were calculated for each repeat times. As these two method may included different predictors in the final model based on different selection and model evaluation algorithms, the relationship of the selected weak predictors and the MSE of coefficients for

strong predictors was analyzed between these two models, which may also represent the impact of missing weak covariance on model parameter estimation. Additionally, in order to test whether different constant C may lead to different relationship between two types of weak predictors and the estimated parameters of strong variables, same simulation pipeline was applied to two situations where C equals to 1, and 30, respectively.

5 Results

5.1 Identifying weak and strong predictors

Utilizing the beta coefficients, predictor matrix X , and response variable Y in the first question, we conducted a comprehensive analysis through 100 iterative executions of variable selection employing two distinct methodologies, subsequently calculating the mean values for sensitivity, specificity, and mean squared error (MSE) to ascertain the average performance metrics.

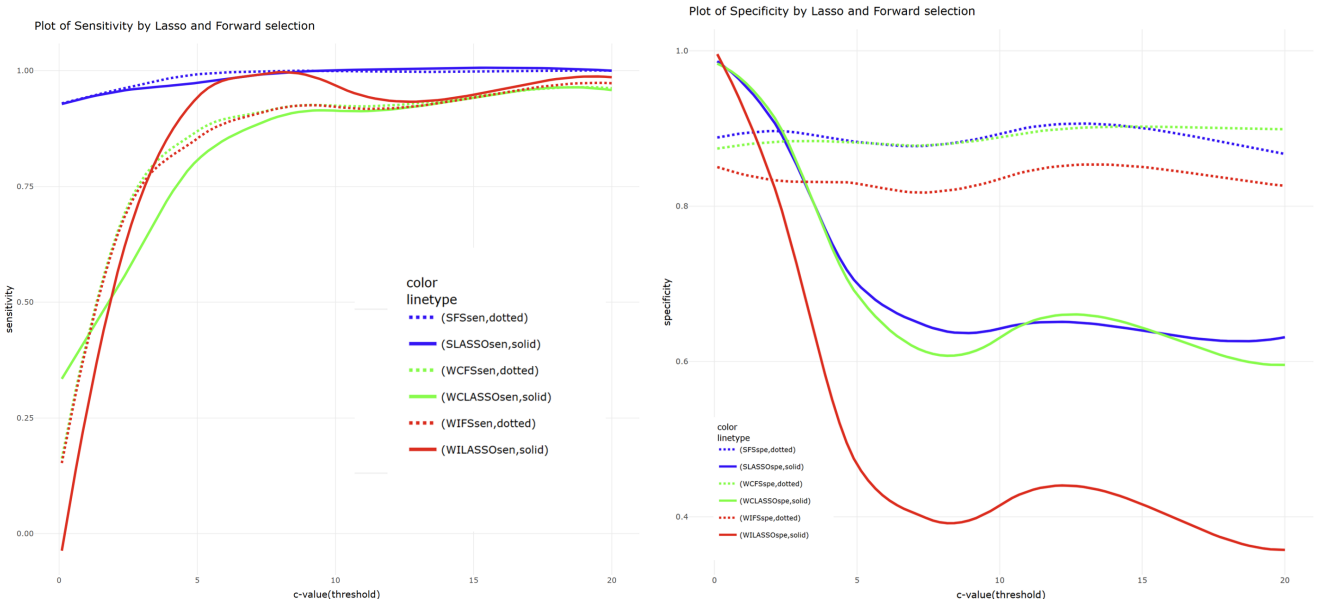


Figure 2: The sensitivity and the specificity by LASSO and forward selection

Examining the plot¹, we observe that the sensitivity trends across all six trajectories exhibit an upward inclination. It's noteworthy that dotted lines represent outcomes from the forward selection method, whereas solid lines are attributed to the LASSO technique. The convergence of these lines when the c -value exceeds 10 suggests an evolution of weak signals into strong signals, culminating in the successful selection of all variables. Particularly, at lower c -values, both LASSO and forward selection demonstrate pronounced sensitivity within the strong signals cohort.

In contrast, the specificity patterns are distinctly more pronounced, with LASSO exhibiting reduced specificity, indicating a propensity for misclassifying weak-and-independent signals.

This delineation underscores the nuanced behavior of the two methods under varying conditions: while both methods show comparable sensitivity for strong signals at lower c -values, their performance diverges for weak-but-correlated and weak-and-independent signals. Specifically, in the forward selection paradigm, weak-but-correlated and weak-and-independent signals exhibit similar detection rates, a pattern not mirrored in the LASSO framework,

¹For clarity regarding the abbreviations presented in the accompanying plot, we provide the following key: "S" denotes strong signals, and while "LASSO" is a well-understood term, we abbreviate forward selection as "FS". Consequently, "SLASSOspe" signifies the specificity metric for the selection of strong signals utilizing the LASSO method.

where weak-but-correlated covariates are more readily identified when the c -value falls below 1. This detailed analysis facilitates a deeper understanding of the variable selection dynamics and the inherent strengths and weaknesses of each method in handling complex signal patterns within high-dimensional datasets.

5.2 The effects of missing "weak" predictors

When C equals to 1, from Fig.1 (A), which displayed the mean predictor counts and the mean MSE, it is noticeable that LASSO tends to select fewer weak predictors than forward method and the number of weak but correlated predictors shows a statistically significant difference between these two models (p -value = 0.000013), while the MSE of parameters of strong signals that estimated by LASSO is significantly lower than forward model. The box plot in Fig.1 (B), which showed the distribution of the MSE over 100 simulation times also indicated the same trends. This may indicated that decreasing weak but correlated predictors may also decrease the MSE of coefficients for strong predictors. This may caused by the potential collinearity between weak but correlated and strong predictors, which might affect the model's prediction of coefficients for strong predictors and then lead to a higher parameter MSE.

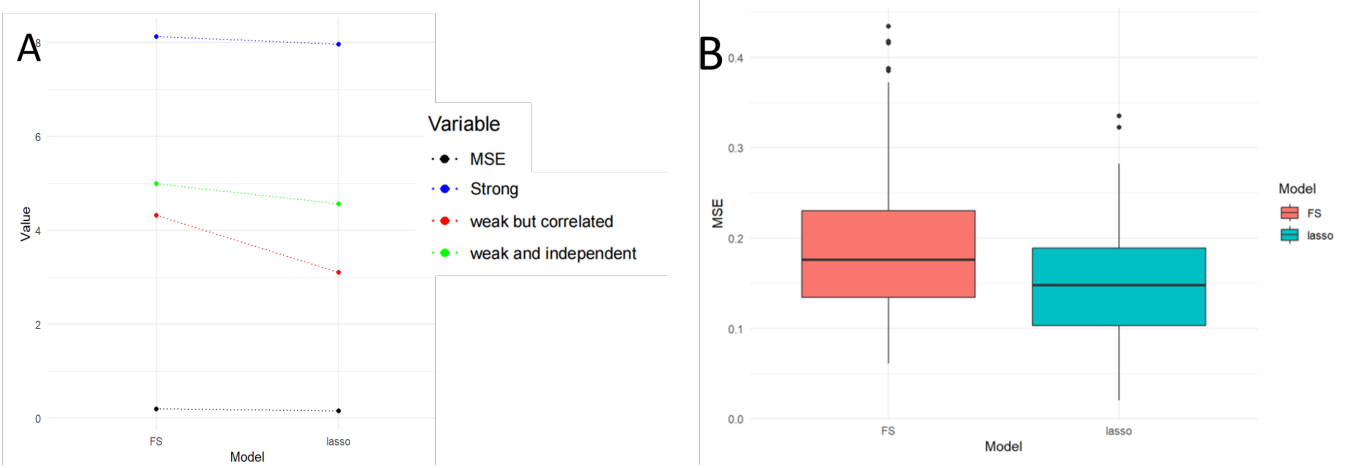


Figure 3: when $C = 1$, (A) The mean predictor counts and the mean MSE for LASSO and Forward selection model. (B) The distribution of the predicted parameters MSE of strong signals by two different models among 100 simulations.

Moreover, when C was set to 30, the MSE and the count of weak and independent predictor have significant difference between LASSO and Forward selection model (p -value = 0.0001011, 0.0000001 respectively), and the mean MSE of strong predictors' coefficient is increasing when the number of weak and independent predictor decreasing (Fig.2 (A) and (B)). In other word, missing weak-and-independent predictors may increase the MSE of estimated parameters for strong predictors. This association may indicate that although some independent predictors may have weak signals, they may still contains important information and excluding them may negatively impact the accuracy of the parameter estimation of the models and then affect the overall performance of the model.

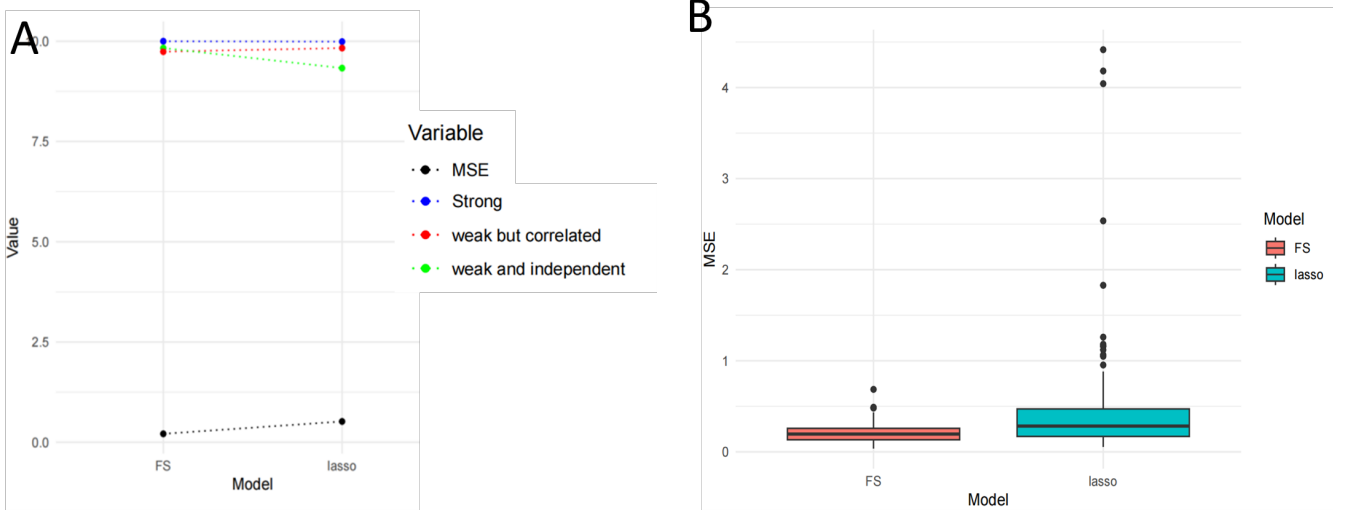


Figure 4: when $C = 30$, (A) The mean predictor counts and the mean MSE for LASSO and Forward selection model. (B) The distribution of the predicted parameters MSE of strong signals by two different models among 100 simulations.

6 Discussion/Conclusions

The study offers significant insights into the performance of Step-wise Forward Selection and LASSO Regression in high-dimensional data settings. Both methods exhibit unique strengths and weaknesses in variable selection, especially in the context of handling weak predictors. This discussion can delve into the nuanced differences between these methodologies, emphasizing how each method's algorithmic approach impacts the accuracy and reliability of model parameter estimation.

A critical finding from the simulation study is the differential impact of weak predictors on model performance, contingent upon the value of the constant C . When C is small, omitting weak but correlated predictors can enhance model accuracy for strong predictors. Conversely, a larger C value reveals that excluding weak and independent predictors detrimentally affects the precision of parameter estimates. This dichotomy underscores the complex interplay between predictor types and model accuracy, suggesting that a one-size-fits-all approach to variable selection may be inadequate.

The statistical analysis conducted highlights the significant differences in model performance metrics, such as the mean square error (MSE), between the two variable selection methods under different conditions. These differences are not just numerically significant but also reveal deeper insights into how each method processes and values different types of predictors within a model. This part of the discussion can focus on interpreting these findings, providing a critical assessment of the implications for statistical modeling and prediction accuracy.

The findings from this study have broader implications for the field of high-dimensional data analysis. The nuanced understanding of variable selection methods provided by this research can inform best practices, guiding researchers in choosing the most appropriate method based on the specific characteristics of their data and the objectives of their analysis. This discussion can explore these implications, emphasizing the importance of method selection in achieving reliable and valid results in high-dimensional statistical modeling.

7 Contribution statement

Jasmine Zhang: Focused on the background and introduction of methods, and part of data algorithms. Zhuodiao Kuang: Designed the content and structure of this article; Focused on the explanation of methods used, and illustrated the modeling process; Tidied the appendix; Coordinated the team. Peng Su: Focused on the effect of missing predictors, part of disucssion/conclusions.

References

- [1] Cheng, J., Sun, J., Yao, K., Xu, M., & Cao, Y. (2022). A variable selection method based on mutual information and variance inflation factor. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 268, 120652.
- [2] Ding, J., Tarokh, V., & Yang, Y. (2018). Model selection techniques: An overview. *IEEE Signal Processing Magazine*, 35(6), 16–34.
- [3] Hastie, T., Tibshirani, R., & Tibshirani, R. J. (2017). Extended comparisons of best subset selection, forward stepwise selection, and the lasso. *arXiv preprint arXiv:1707.08692*.
- [4] Li, Y., Hong, H. G., Ahmed, S. E., & Li, Y. (2019). Weak signals in high-dimensional regression: Detection, estimation and prediction. *Applied stochastic models in business and industry*, 35(2), 283–298.
- [5] Ratner, B. (2010). Variable selection methods in regression: Ignorable problem, outing notable solution. *Journal of Targeting, Measurement and Analysis for Marketing*, 18, 65–75.
- [6] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1), 267–288.

8 Appendix

8.1 Formulas

Step-wise forward method: Starting with the empty model, and iteratively adds the variables that best improves the model fit. That is often done by sequentially adding predictors with the largest reduction in AIC. For linear models,

$$AIC = n \ln \left(\sum_{i=1}^n (y_i - \hat{y}_i)^2 / n \right) + 2p,$$

where \hat{y}_i is the fitted values from a model, and p is the dimension of the model (i.e., number of predictors plus 1).

Automated LASSO regression LASSO is another popular method for variable selection. It estimates the model parameters by optimizing a penalized loss function:

$$\min_{\beta} \frac{1}{2n} \sum_{i=1}^n (y_i - x_i \beta)^2 + \lambda \left\| \sum_{k=1}^p |\beta_k| \right\|$$

where λ is a tuning parameter. Cross-validation (CV) is the most common selection criteria for LASSO.

Strong and Weak Predictors

Definition of strong signals:

$$S_1 = \{j : |\beta_j| > c \sqrt{\log(p)/n}, \text{ some } c > 0, 1 \leq j \leq p\}$$

Definition of weak-but-correlated signals:

$$S_2 = \{j : 0 < |\beta_j| \leq c \sqrt{\log(p)/n}, \text{ some } c > 0, \text{ corr}(X_j, X_{j'}) \neq 0, \text{ for some } j' \in S_1, 1 \leq j \leq p\}$$

Definition of weak-and-independent signals:

$$S_2 = \{j : 0 < |\beta_j| \leq c \sqrt{\log(p)/n}, \text{ some } c > 0, \text{ corr}(X_j, X_{j'}) = 0, \text{ for some } j' \in S_1, 1 \leq j \leq p\}$$

Model Diagnostic

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

where TP represents true positives (the number of non-zero coefficients correctly identified), and FN stands for false negatives (the number of non-zero coefficients not selected by the model).

$$\text{Specificity} = \frac{TN}{TN + FP}$$

where TN denotes true negatives (the number of zero coefficients correctly identified as irrelevant), and FP signifies false positives (the number of zero coefficients incorrectly selected as relevant).

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

where Y_i represents the actual value for the i th observation, \hat{Y}_i is the predicted value, and n is the number of observations.

8.2 Code

```

1
2 \subsubsection{General setting for the first simulation test}
3
4 ## Data Generation
5
6 ‘‘{r}
7 set.seed(1)
8 n <- 100
9 # 20+20+20
10 p <- 60
11
12 # thresh1
13 thr <- sqrt(log(p)/n)
14 thr
15
16 # X
17 # make sure that 1-20 and 41-60 are not correlated,
18 # which means they are independently generated
19 # generate 21-40 with weak-but-correlated signals
20
21 # Responding beta's are not 0
22 X1.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
23 X2.1 <- 3*X1.1+matrix(rnorm(n * p/3/2), n, p/3/2)
24 X3.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
25 # Responding beta's are 0
26 X1.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
27 X2.0 <- 3*X1.0+matrix(rnorm(n * p/3/2), n, p/3/2)
28 X3.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
29
30 X<-cbind(X1.1, X2.1, X3.1, X1.0, X2.0, X3.0)
31
32
33 # beta
34
35
36 ## positive and negative for the first half

```

```

37 b.true1.strong <- c(thr+abs(rnorm(5)),-thr-abs(rnorm(5)))
38 b.true1.weak1  <- runif(10,-thr,thr)
39 b.true1.weak2  <- runif(10,-thr,thr)
40 ## zero for the second half
41 b.true0 <- rep(0,p/2)
42 ## combine them together
43 b.true    <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
44 ## name b.true
45 names(b.true) <- paste0("X", seq(1, 60))
46
47
48 # Y
49 Y <- 1 + X %*% b.true + rnorm(n)
50 df <- data.frame(cbind(X, Y))
51 names(df)[p + 1] <- "y"
52
53 cat("True non-zero effects:", which(b.true != 0), "\n")
54 ## plot
55 plot(b.true)
56
57 ‘‘‘
58
59
60 # Plot for beta
61
62
63 ‘‘{r}
64 par(mfrow = c(1,2))
65 n <- 100
66 # 20+20+20
67 p <- 60
68
69 # thresh1
70 thr <- sqrt(log(p)/n)
71 thr
72
73 # beta
74 ##positiveandnegativeforthefirsthalf
75 b.true1.strong<-c(rep(thr+1,5),rep(-thr-1,5))
76 b.true1.weak1 <-c(rep(thr/2,5),rep(-thr/2,5))
77 b.true1.weak2 <-c(rep(thr/2,5),rep(-thr/2,5))
78 ##zeroforthesecondhalf
79 b.true0<-rep(0,p/2)
80

```

```

81 b.true1          <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
82 plot(b.true1)
83
84 # beta
85
86
87 ## positive and negative for the first half
88 b.true1.strong <- c(thr+abs(rnorm(5,thr,1)),-thr-abs(rnorm(5,thr,1)))
89 b.true1.weak1  <- runif(10,-thr,thr)
90 b.true1.weak2  <- runif(10,-thr,thr)
91 ## zero for the second half
92 b.true0 <- rep(0,p/2)
93 ## combine them together
94 b.true2          <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
95 plot(b.true2)
96 ‘ ‘ ‘
97
98
99
100 ‘ ‘ {r}
101
102 index<-c(1:60)
103 group1<-rep("Group1",60)
104 group2<-rep("Group2",60)
105 B.true1<-cbind(b.true1,index,group1)
106 B.true2<-cbind(b.true2,index,group2)
107 B<-rbind(B.true1,B.true2)
108
109 library(ggplot2)
110
111 # Assuming B is correctly created and is a data frame
112 B <- data.frame(B) # Ensuring B is a data frame
113 colnames(B) <- c("Value", "Index", "Group") # Naming columns for clarity
114
115 # Converting the appropriate columns to the correct data types
116 B$Index <- as.numeric(B$Index)
117 B$Group <- as.factor(B$Group)
118 B$Value <- as.numeric(B$Value)
119
120 # Plotting with ggplot2
121 p1<-ggplot(B, aes(x = Index, y = Value, color = Group)) +
122   geom_point() + # Using points to plot the data
123   ylim(-2,2)+
124   labs(title = "Plot of B.true1 and B.true2", x = "Index", y = "Beta Value") +

```

```

125 scale_color_manual(values = c("Group1" = "blue", "Group2" = "red")) # Customizing colors
126
127
128
129 library(htmltools)
130 library(plotly)
131
132 # Assuming B is your data frame set up correctly
133
134 # Create a ggplot2 object
135 p <- ggplot(B, aes(x = Index, y = Value, color = Group)) +
136   geom_point() + # Using points to plot the data
137   labs(title = "Plot of B.true1 and B.true2", x = "Index", y = "Beta Value") +
138   scale_color_manual(values = c("Group1" = "blue", "Group2" = "red")) # Customizing colors
139
140 # Convert the ggplot2 object to a plotly object
141 p_plotly1 <- ggplotly(p)
142
143
144 ‘‘‘
145
146
147
148
149 ## Forward Selection
150
151 ‘‘{r}
152 set.seed(1)
153 # Forward Selection(default setting is k=log(n) from AIC)
154 fit.forward <- step(object = lm(y ~ 1, data = df),
155                     scope = formula(lm(y ~ ., data = df)),
156                     direction = "forward", trace = 0) # AIC
157 summary(fit.forward)
158 # Selected ones
159
160
161 ‘‘‘
162
163
164
165
166 ## LASSO
167
168 ‘‘{r}

```

```

169 # LASSO
170 fit.lasso <- cv.glmnet(X, Y, nfolds = 10, type.measure = "mse") # 5-fold CV using mean squared
    error
171 param.best <- fit.lasso$glmnet.fit$beta[, fit.lasso$lambda == fit.lasso$lambda.1se] # one standard
    -error rule
172 param.best[param.best != 0]
173 '''
174
175
176
177 # (1) how well each of the two methods in identifying weak and strong predictors;
178
179 ## Forward Selection
180
181
182 '''{r}
183 FS_calculation <- function(b.true, selected_vars) {
184     # Names of all variables
185     all_vars <- names(b.true)
186
187     # Convert b.true to a binary vector indicating whether each variable is non-zero (TRUE) or zero
        (FALSE)
188     is_non_zero <- b.true != 0
189
190     # Create a binary vector indicating whether each variable is selected (TRUE) or not (FALSE)
191     is_selected <- all_vars %in% selected_vars
192
193     # True Positives (TP): Non-zero variables that were selected
194     TP <- sum(is_non_zero & is_selected)
195
196     # False Negatives (FN): Non-zero variables that were not selected
197     FN <- sum(is_non_zero & !is_selected)
198
199     # True Negatives (TN): Zero variables that were not selected
200     TN <- sum(!is_non_zero & !is_selected)
201
202     # False Positives (FP): Zero variables that were selected
203     FP <- sum(!is_non_zero & is_selected)
204
205     # Calculate sensitivity and specificity
206     sensitivity <- TP / (TP + FN)
207     specificity <- TN / (TN + FP)
208
209     list(sensitivity = sensitivity, specificity = specificity)

```

```

210 }
211
212
213
214 '''
215
216
217 ### Simulation for Forward Selection
218
219 '{{{r}
220 set.seed(2024)
221 ### BREAD
222 # Calculate sensitivity and specificity
223 Strong_FS_sensitivity_sum <-
224 Strong_FS_specificity_sum <-
225 Weakcor_FS_sensitivity_sum <-
226 Weakcor_FS_specificity_sum <-
227 Weakind_FS_sensitivity_sum <-
228 Weakind_FS_specificity_sum <-
229 mse_FS<-0
230
231 for (i in 1:LOOP) {
232 # Data Generation
233 # Responding beta's are not 0
234 X1.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
235 X2.1 <- 3*X1.1+matrix(rnorm(n * p/3/2), n, p/3/2)
236 X3.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
237 # Responding beta's are 0
238 X1.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
239 X2.0 <- 3*X1.0+matrix(rnorm(n * p/3/2), n, p/3/2)
240 X3.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
241
242 X<-cbind(X1.1, X2.1, X3.1, X1.0, X2.0, X3.0)
243
244
245 # beta
246
247
248 ## positive and negative for the first half
249 b.true1.strong <- c(thr+abs(rnorm(5)),-thr-abs(rnorm(5)))
250 b.true1.weak1 <- runif(10,-thr,thr)
251 b.true1.weak2 <- runif(10,-thr,thr)
252 ## zero for the second half
253 b.true0 <- rep(0,p/2)

```

```

254 ## combine them together
255 b.true          <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
256 ## name b.true
257 names(b.true) <- paste0("X", seq(1, 60))
258
259
260 # Y
261 Y <- 1 + X %*% b.true + rnorm(n)
262 df <- data.frame(cbind(X, Y))
263 names(df)[p + 1] <- "y"
264
265 # Selection
266 fit.forward <- step(object = lm(y ~ 1, data = df),
267                      scope = formula(lm(y ~ ., data = df)),
268                      direction = "forward", trace = 0) # AIC
269
270
271 # Calculate MSE
272 predictions <- fit.forward$fitted.values
273 mse_FS <- mse_FS + mean((Y - predictions)^2)
274
275 # Groups for the sensitivity and the specificity
276 selected_vars<-names(fit.forward$coefficients[-1])
277 FS_group1 <- FS_calculation(b.true[c(1:10,31:40)], selected_vars)
278 Strong_FS_sensitivity_sum <- Strong_FS_sensitivity_sum + FS_group1$sensitivity
279 Strong_FS_specificity_sum <- Strong_FS_specificity_sum + FS_group1$specificity
280
281 FS_group2 <- FS_calculation(b.true[c(11:20,41:50)], selected_vars)
282 Weakcor_FS_sensitivity_sum <- Weakcor_FS_sensitivity_sum + FS_group2$sensitivity
283 Weakcor_FS_specificity_sum <- Weakcor_FS_specificity_sum + FS_group2$specificity
284
285 FS_group3 <- FS_calculation(b.true[c(21:30,51:60)], selected_vars)
286 Weakkind_FS_sensitivity_sum <- Weakkind_FS_sensitivity_sum + FS_group3$sensitivity
287 Weakkind_FS_specificity_sum <- Weakkind_FS_specificity_sum + FS_group3$specificity
288
289 }
290
291 ### BREAD
292 Strong_FS_sensitivity = Strong_FS_sensitivity_sum/LOOP
293 Strong_FS_sensitivity
294 Strong_FS_specificity = Strong_FS_specificity_sum/LOOP
295 Strong_FS_specificity
296
297 Weakcor_FS_sensitivity = Weakcor_FS_sensitivity_sum/LOOP

```

```

298 Weakcor_FS_sensitivity
299 Weakcor_FS_specificity = Weakcor_FS_specificity_sum/LOOP
300 Weakcor_FS_specificity
301
302 Weakind_FS_sensitivity = Weakind_FS_sensitivity_sum/LOOP
303 Weakind_FS_sensitivity
304 Weakind_FS_specificity = Weakind_FS_specificity_sum/LOOP
305 Weakind_FS_specificity
306
307 mse_FS/LOOP
308 '''
309
310
311
312 ## LASSO
313
314
315
316 '''{r}
317 # calculate sensitivity and specificity
318 LASSO_calculation <- function(selected_coefs, non_zero_indices, zero_indices) {
319   true_positives <- sum(selected_coefs[non_zero_indices] != 0)
320   true_negatives <- sum(selected_coefs[zero_indices] == 0)
321   false_negatives <- sum(selected_coefs[non_zero_indices] == 0)
322   false_positives <- sum(selected_coefs[zero_indices] != 0)
323
324   sensitivity <- true_positives / (true_positives + false_negatives)
325   specificity <- true_negatives / (true_negatives + false_positives)
326
327   return(list(sensitivity = sensitivity, specificity = specificity))
328 }
329 '''
330
331 ### Simulation for Forward Selection
332
333
334 '''{r}
335 set.seed(2024)
336
337 ### BREAD
338 # Calculate sensitivity and specificity
339 Strong_LASSO_sensitivity_sum <-
340 Strong_LASSO_specificity_sum <-
341 Weakcor_LASSO_sensitivity_sum <-

```



```

342 Weakcor_LASSO_specificity_sum <-
343 Weakind_LASSO_sensitivity_sum <-
344 Weakind_LASSO_specificity_sum <-
345     mse_LASSO<-0
346
347 for (i in 1:LOOP) {
348   # Data Generation
349   # Responding beta's are not 0
350   X1.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
351   X2.1 <- 3*X1.1+matrix(rnorm(n * p/3/2), n, p/3/2)
352   X3.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
353   # Responding beta's are 0
354   X1.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
355   X2.0 <- 3*X1.0+matrix(rnorm(n * p/3/2), n, p/3/2)
356   X3.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
357
358   X<-cbind(X1.1, X2.1, X3.1, X1.0, X2.0, X3.0)
359
360
361   # beta
362
363
364   ## positive and negative for the first half
365   b.true1.strong <- c(thr+abs(rnorm(5)),-thr-abs(rnorm(5)))
366   b.true1.weak1   <- runif(10,-thr,thr)
367   b.true1.weak2   <- runif(10,-thr,thr)
368   ## zero for the second half
369   b.true0 <- rep(0,p/2)
370   ## combine them together
371   b.true    <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
372   ## name b.true
373   names(b.true) <- paste0("X", seq(1, 60))
374
375
376   # Y
377   Y <- 1 + X %*% b.true + rnorm(n)
378   df <- data.frame(cbind(X, Y))
379   names(df)[p + 1] <- "y"
380
381   # Selection
382   # LASSO
383   fit.lasso <- cv.glmnet(X, Y, nfolds = 10, type.measure = "mse") # 5-fold CV using mean squared
     error
384   param.best <- fit.lasso$glmnet.fit$beta[, fit.lasso$lambda == fit.lasso$lambda.1se] # one standard

```

```

    -error rule
385 param.best[param.best != 0]
386
387
388 # Calculate MSE
389 predictions <- predict(fit.lasso, s = fit.lasso$lambda.1se, newx = as.matrix(X))
390 mse_LASSO <- mse_LASSO+mean((Y - predictions)^2)
391
392
393
394
395 # calculate SS for each group
396 LASSO_group1 <- LASSO_calculation(param.best, 1:10, 31:40)
397 Strong_LASSO_sensitivity_sum <- Strong_LASSO_sensitivity_sum + LASSO_group1$sensitivity
398 Strong_LASSO_specificity_sum <- Strong_LASSO_specificity_sum + LASSO_group1$specificity
399
400 LASSO_group2 <- LASSO_calculation(param.best, 11:20, 41:50)
401 Weakcor_LASSO_sensitivity_sum <- Weakcor_LASSO_sensitivity_sum + LASSO_group2$sensitivity
402 Weakcor_LASSO_specificity_sum <- Weakcor_LASSO_specificity_sum + LASSO_group2$specificity
403
404 LASSO_group3 <- LASSO_calculation(param.best, 21:30, 51:60)
405 Weakind_LASSO_sensitivity_sum <- Weakind_LASSO_sensitivity_sum + LASSO_group3$sensitivity
406 Weakind_LASSO_specificity_sum <- Weakind_LASSO_specificity_sum + LASSO_group3$specificity
407
408
409 }
410
411 ### BREAD
412 Strong_LASSO_sensitivity = Strong_LASSO_sensitivity_sum/LOOP
413 Strong_LASSO_sensitivity
414 Strong_LASSO_specificity = Strong_LASSO_specificity_sum/LOOP
415 Strong_LASSO_specificity
416
417 Weakcor_LASSO_sensitivity = Weakcor_LASSO_sensitivity_sum/LOOP
418 Weakcor_LASSO_sensitivity
419 Weakcor_LASSO_specificity = Weakcor_LASSO_specificity_sum/LOOP
420 Weakcor_LASSO_specificity
421
422 Weakind_LASSO_sensitivity = Weakind_LASSO_sensitivity_sum/LOOP
423 Weakind_LASSO_sensitivity
424 Weakind_LASSO_specificity = Weakind_LASSO_specificity_sum/LOOP
425 Weakind_LASSO_specificity
426
427 mse_LASSO/LOOP

```

```

428 ```
429
430
431 \subsubsection{Dashboard for question 1}
432
433 ```{r setup, include=FALSE}
434 library(flexdashboard)
435 library(tidyverse)
436 library(plotly)
437 # Install thematic and un-comment for themed static plots (i.e., ggplot2)
438 # thematic::thematic_rmd()
439 load("mse_sen224.RData")
440 load("mse_spe224.RData")
441 ```
442
443 Column {data-width=650 .tabset}
444 -----
445
446 ### Sensitivity
447
448 ```{r, echo=FALSE}
449 mse_sen_new <- mse_sen|>
450   pivot_longer(
451     c('SLASSOsen', 'WCLASSOsen', 'WILASSOsen',
452       'SFSsen', 'WCFSSen', 'WIFSSen'),
453     names_to = "Sensitivity", values_to = "values")
454 data_sen<- data.frame(
455   x = mse_sen_new$c,
456   y = mse_sen_new$values,
457   group = factor(rep(1:6, 8)),
458   color = rep(c("SLASSOsen", "WCLASSOsen", "WILASSOsen", "SFSsen", "WCFSSen", "WIFSSen"), 8),
459   linetype = rep(c("solid", "solid", "solid", "dotted", "dotted", "dotted"), 8)
460 )
461
462 # Plot
463 p2 <- ggplot(data_sen, aes(x=x, y=y, color=color, group=group, linetype=linetype)) +
464   geom_smooth(size=1, se = FALSE) +
465   scale_color_manual(values=c("SLASSOsen"="blue", "SFSsen"="blue",
466                               "WCLASSOsen"="green", "WCFSSen"="green",
467                               "WILASSOsen"="red", "WIFSSen"="red")) +
468   labs(title="Plot of Sensitivity by Lasso and Forward selection", x="c-value(threshold)", y="
469         sensitivity")+
470   scale_linetype_manual(values=c("solid"="solid", "dotted"="dotted")) +
471   theme_minimal() +

```

```

471 theme(legend.title=element_blank())+
472 guides(size=FALSE)
473
474 p_plotly2 <- ggplotly(p2)
475 p_plotly2
476
477 ' ' '
478
479
480
481
482 ### Specificity
483
484
485 ' ' {r,echo=FALSE}
486 mse_spe_new <- mse_spe|>
487   pivot_longer(
488     c('SLASSOspe', 'WCLASSOspe', 'WILASSOspe',
489       'SFSspe', 'WCFSspe', 'WIFSspe'),
490     names_to = "Specificity", values_to = "values")
491 data_spe_new<- data.frame(
492   x = mse_spe_new$c,
493   y = mse_spe_new$values,
494   group = factor(rep(1:6, 8)),
495   color = rep(c("SLASSOspe", "WCLASSOspe", "WILASSOspe", "SFSspe", "WCFSspe", "WIFSspe"), 8),
496   linetype = rep(c("solid", "solid", "solid", "dotted", "dotted", "dotted"), 8)
497 )
498
499 # Plot
500 p3 <- ggplot(data_spe_new, aes(x=x, y=y, color=color, group=group, linetype=linetype)) +
501   geom_smooth(size=1, se = FALSE) +
502   scale_color_manual(values=c("SLASSOspe"="blue", "SFSspe"="blue",
503                               "WCLASSOspe"="green", "WCFSspe"="green",
504                               "WILASSOspe"="red", "WIFSspe"="red")) +
505   labs(title="Plot of Specificity by Lasso and Forward selection", x="c-value(threshold)", y="
506         specificity")+
507   scale_linetype_manual(values=c("solid"="solid", "dotted"="dotted")) +
508   theme_minimal() +
509   theme(legend.title=element_blank())+
510   guides(size=FALSE)
511
512 p_plotly3 <- ggplotly(p3)
513 p_plotly3

```

```

514 ' '
515
516
517 Column {data-width=350}
518 -----
519
520 ### Other Information
521
522
523 Abbreviation
524
525 S - Strong signals
526
527 WC - Weak but correlated signals
528
529 WI - Weak and independent signals
530
531 FS - Forward selection
532
533 spe - specificity
534
535 sen - sensitivity
536
537
538 eg:
539 'SLASSOspe' stands for the specificity of selecting strong signals by LASSO.
540
541 Definition of strong signals ---
542
543 
$$S_1 = \{j: |\beta_j| > c \sqrt{\log(p)/n}, \text{some } c > 0, 1 \leq j \leq p\}$$

544
545 Definition of weak-but-correlated signals ---
546
547 
$$S_2 = \{j: 0 < |\beta_j| \leq c \sqrt{\log(p)/n}, \text{some } c > 0, \text{corr}(X_j, X_{j'}) \neq 0, \\ \text{for some } j' \in S_1, 1 \leq j \leq p\}$$

548
549
550
551 Definition of weak-and-independent signals ---
552
553 
$$S_3 = \{j: 0 < |\beta_j| \leq c \sqrt{\log(p)/n}, \text{some } c > 0, \text{corr}(X_j, X_{j'}) = 0, \\ \text{for all } j' \in S_1, 1 \leq j \leq p\}$$

554
555 
$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$


```

```

556
557 ### Beta Generation
558
559 ‘‘{r, include=FALSE}
560 n <- 1000
561 # 20+20+20
562 p <- 60
563
564 # thresh1
565 thr <- sqrt(log(p)/n)
566 thr
567
568 # beta
569 ##positiveandnegativeforthefirsthalf
570 b.true1.strong<-c(rep(thr+1,5),rep(-thr-1,5))
571 b.true1.weak1 <-c(rep(thr/2,5),rep(-thr/2,5))
572 b.true1.weak2 <-c(rep(thr/2,5),rep(-thr/2,5))
573 ##zeroforthesecondhalf
574 b.true0<-rep(0,p/2)
575
576 b.true1          <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
577
578
579 # beta
580
581
582 ## positive and negative for the first half
583 b.true1.strong <- c(thr+abs(rnorm(5,thr,1)),-thr-abs(rnorm(5,thr,1)))
584 b.true1.weak1  <- runif(10,-thr,thr)
585 b.true1.weak2  <- runif(10,-thr,thr)
586 ## zero for the second half
587 b.true0 <- rep(0,p/2)
588 ## combine them together
589 b.true2          <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
590
591 ‘‘‘
592
593
594
595 ‘‘{r, echo=FALSE}
596
597 index<-c(1:60)
598 group1<-rep("Group1",60)
599 group2<-rep("Group2",60)

```

```

600 B.true1<-cbind(b.true1,index,group1)
601 B.true2<-cbind(b.true2,index,group2)
602 B<-rbind(B.true1,B.true2)
603
604
605 # Assuming B is correctly created and is a data frame
606 B <- data.frame(B) # Ensuring B is a data frame
607 colnames(B) <- c("Value", "Index", "Group") # Naming columns for clarity
608
609 # Converting the appropriate columns to the correct data types
610 B$Index <- as.numeric(B$Index)
611 B$Group <- as.factor(B$Group)
612 B$Value <- as.numeric(B$Value)
613
614
615
616 # Assuming B is your data frame set up correctly
617
618 # Create a ggplot2 object
619 p1 <- ggplot(B, aes(x = Index, y = Value, color = Group)) +
620   geom_point() + # Using points to plot the data
621   labs(title = "Plot of B.true1 and B.true2", x = "Index", y = "Beta Value") +
622   scale_color_manual(values = c("Group1" = "blue", "Group2" = "red")) # Customizing colors
623
624 # Convert the ggplot2 object to a plotly object
625 p_plotly1 <- ggplotly(p1)
626 p_plotly1
627
628 ‘‘‘
629
630
631
632
633 \subsubsection{Simulation for question 2}
634 ‘‘{r setup, include=FALSE}
635 knitr::opts_chunk$set(echo = TRUE)
636 library(tidyverse)
637 require(survival)
638 require(quantreg)
639 require(glmnet)
640 require(MASS)
641 require(pROC)
642
643 LOOP = 100

```

```

644 n <- 100
645 # 20+20+20
646 p <- 60
647 c = 30
648
649 # thresh1
650 thr <- c * sqrt(log(p)/n)
651 '''
652
653
654 # For foward selection
655
656 '''{r}
657 #function of calculating effect of missing weak beta to strong beta for FS
658 FS_weak_effect <- function(model){
659   # model coef data
660   coef_names <- names(coef(model))
661   coef_values <- coef(model)
662
663   FS_coef_df <- data.frame(Predictor = b_true_df$Predictor,
664                             Coefficient = 0,
665                             Group = b_true_df$Group)
666
667   FS_coef_df$Coefficient <- ifelse(FS_coef_df$Predictor %in% coef_names,
668                                     coef_values[match(FS_coef_df$Predictor, coef_names)],
669                                     0)
670   return(FS_coef_df)
671 }
672 '''
673
674 '''{r}
675 set.seed(2024)
676 mse_strong <-
677   strong_nonzero_count <-
678   weak1_nonzero_count <-
679   weak2_nonzero_count <-
680   pearson_corr_strong <-
681   spearman_corr_strong <- 0
682
683 results = list()
684 #repeat 100 times for FS
685 for (i in 1:LOOP) {
686   # Data Generation
687   # Responding beta's are not 0

```



```

688 X1.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
689 X2.1 <- 3*X1.1+matrix(rnorm(n * p/3/2), n, p/3/2)
690 X3.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
691 # Responding beta's are 0
692 X1.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
693 X2.0 <- 3*X1.0+matrix(rnorm(n * p/3/2), n, p/3/2)
694 X3.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
695
696 X<-cbind(X1.1, X2.1, X3.1, X1.0, X2.0, X3.0)
697
698 # beta
699 ## positive and negative for the first half
700 b.true1.strong <- c(thr+abs(rnorm(5)),-thr-abs(rnorm(5)))
701 b.true1.weak1 <- runif(10,-thr,thr)
702 b.true1.weak2 <- runif(10,-thr,thr)
703 ## zero for the second half
704 b.true0 <- rep(0,p/2)
705 ## combine them together
706 b.true <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
707 ## name b.true
708 names(b.true) <- paste0("X", seq(1, 60))
709
710 # Y
711 Y <- 1 + X %*% b.true + rnorm(n)
712 df <- data.frame(cbind(X, Y))
713 names(df)[p + 1] <- "y"
714
715 # true beta dataframe
716 b_true_df <- data.frame(Predictor = names(b.true), Coefficient = as.numeric(b.true))
717 b_true_df$Coefficient
718 b_true_df$Group <- ifelse(b_true_df$Coefficient %in% b.true1.strong, "strong",
719                           ifelse(b_true_df$Coefficient %in% b.true1.weak1, "weak_non",
720                                   ifelse(b_true_df$Coefficient %in% b.true1.weak2,
721                                           "weak_corr", "zero")))
722
723 # Selection
724 fit.forward <- step(object = lm(y ~ 1, data = df),
725                    scope = formula(lm(y ~ ., data = df)),
726                    direction = "forward", trace = 0) # AIC
727
728 # model coef data
729 FS_coef_df = FS_weak_effect(fit.forward)
730
731 # number of beta

```

```

732 strong_nonzero_count <- sum(FS_coef_df$Group == "strong" & FS_coef_df$Coefficient != 0)
733 weak1_nonzero_count <- sum(FS_coef_df$Group == "weak_non" & FS_coef_df$Coefficient != 0)
734 weak2_nonzero_count <- sum(FS_coef_df$Group == "weak_corr" & FS_coef_df$Coefficient != 0)
735
736 # calculate strong beta MSE
737 ## merge model coef and true beta
738 merged_df <- merge(b_true_df, FS_coef_df, by = "Predictor", suffixes = c("_true", "_FS"))
739
740 strong_rows <- merged_df[merged_df$Group_true == "strong", ]
741
742 ## mse
743 mse_strong <- mean((strong_rows$Coefficient_true - strong_rows$Coefficient_FS)^2)
744
745 # correlation
746 model_strong <- FS_coef_df[FS_coef_df$Group == "strong", ]
747
748 true_strong <- b_true_df[b_true_df$Group == "strong", ]
749
750 ## pearson
751 pearson_corr_strong <- cor(model_strong$Coefficient, true_strong$Coefficient, method = "
  pearson")
752
753 ## spearman
754 spearman_corr_strong <- cor(model_strong$Coefficient, true_strong$Coefficient, method = "
  spearman")
755
756 results[[i]] <- list(
757   mse_strong = mse_strong ,
758   strong_nonzero_count = strong_nonzero_count ,
759   weak1_nonzero_count = weak1_nonzero_count ,
760   weak2_nonzero_count = weak2_nonzero_count ,
761   pearson_corr_strong = pearson_corr_strong ,
762   spearman_corr_strong = spearman_corr_strong
763 )
764 }
765
766 #result data
767 results_df <- as.data.frame(do.call(rbind, results))
768 results_df <-
769   results_df |>
770   mutate(mse_strong = as.numeric(mse_strong),
771          strong_nonzero_count = as.numeric(strong_nonzero_count),
772          weak1_nonzero_count = as.numeric(weak1_nonzero_count),
773          weak2_nonzero_count = as.numeric(weak2_nonzero_count),

```

```

774     pearson_corr_strong = as.numeric(pearson_corr_strong),
775     spearman_corr_strong = as.numeric(spearman_corr_strong))
776
777 FS_mean_values <- colMeans(results_df)
778
779 #result data mean
780 res_FS <- data.frame(
781   mse = FS_mean_values["mse_strong"],
782   strong = FS_mean_values["strong_nonzero_count"],
783   weak_non = FS_mean_values["weak1_nonzero_count"],
784   weak_corr = FS_mean_values["weak2_nonzero_count"],
785   pearson = FS_mean_values["pearson_corr_strong"],
786   spearman = FS_mean_values["spearman_corr_strong"],
787   model = "FS"
788 )
789 rownames(res_FS) <- NULL
790 ' '
791
792 # For lasso
793 '{r}
794 # LASSO
795 fit.lasso <- cv.glmnet(X, Y, nfolds = 10, type.measure = "mse") # 5-fold CV using mean squared
796   error
797 param.best <- fit.lasso$glmnet.fit$beta[, fit.lasso$lambda == fit.lasso$lambda.1se] # one standard
798   -error rule
799 param = param.best[param.best != 0]
800 ' '
801 '{r}
802 #function of calculating effect of missing weak beta to strong beta for lasso
803 lasso_weak_effect <- function(param){
804
805   # model coef data
806   coef_names <- gsub("V", "X", names(param))
807   formula_str <- paste("y ~", paste(coef_names, collapse = " + "))
808   slm <- lm(formula_str, data = df)
809
810   recoef_names <- names(coef(slm))
811   recoef_values <- coef(slm)
812
813   lasso_coef_df <- data.frame(Predictor = b_true_df$Predictor,
814                               Coefficient = 0,
815                               Group = b_true_df$Group)

```

```

816
817 lasso_coef_df$Coefficient <- ifelse(lasso_coef_df$Predictor %in% recoef_names,
818                                     recoef_values[match(lasso_coef_df$Predictor, recoef_names)],
819                                     0)
820
821 return(lasso_coef_df)
822 }
823
824 '''
825 set.seed(2024)
826 lasso_mse_strong <-
827   lasso_strong_nonzero_count <-
828   lasso_weak1_nonzero_count <-
829   lasso_weak2_nonzero_count <-
830   lasso_pearson_corr_strong <-
831   lasso_spearman_corr_strong <- 0
832 results = list()
833
834 #repeat 100 times for FS
835 for (i in 1:LOOP) {
836   # Data Generation
837   # Responding beta's are not 0
838   X1.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
839   X2.1 <- 3*X1.1+matrix(rnorm(n * p/3/2), n, p/3/2)
840   X3.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
841   # Responding beta's are 0
842   X1.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
843   X2.0 <- 3*X1.0+matrix(rnorm(n * p/3/2), n, p/3/2)
844   X3.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
845
846   X<-cbind(X1.1, X2.1, X3.1, X1.0, X2.0, X3.0)
847
848   # beta
849   ## positive and negative for the first half
850   b.true1.strong <- c(thr+abs(rnorm(5)),-thr-abs(rnorm(5)))
851   b.true1.weak1   <- runif(10,-thr,thr)
852   b.true1.weak2   <- runif(10,-thr,thr)
853   ## zero for the second half
854   b.true0 <- rep(0,p/2)
855   ## combine them together
856   b.true   <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
857   ## name b.true
858   names(b.true) <- paste0("X", seq(1, 60))
859

```

```

860 # Y
861 Y <- 1 + X %*% b.true + rnorm(n)
862 df <- data.frame(cbind(X, Y))
863 names(df)[p + 1] <- "y"
864
865 # true beta dataframe
866 b_true_df <- data.frame(Predictor = names(b.true), Coefficient = as.numeric(b.true))
867 b_true_df$Coefficient
868 b_true_df$Group <- ifelse(b_true_df$Coefficient %in% b.true1.strong, "strong",
869                           ifelse(b_true_df$Coefficient %in% b.true1.weak1, "weak_non",
870                                   ifelse(b_true_df$Coefficient %in% b.true1.weak2,
871                                           "weak_corr", "zero")))
872
873 # Selection
874 # LASSO
875 # 5-fold CV using mean squared error
876 fit.lasso <- cv.glmnet(X, Y, nfolds = 10, type.measure = "mse")
877 # one standard-error rule
878 param.best <- fit.lasso$glmnet.fit$beta[, fit.lasso$lambda == fit.lasso$lambda.1se]
879 #selected model paramters
880 best_param <- param.best[param.best != 0]
881
882 # model coef data
883 lasso_coef_df = lasso_weak_effect(best_param)
884
885 # number of beta
886 lasso_strong_nonzero_count <- sum(lasso_coef_df$Group == "strong" & lasso_coef_df$Coefficient !=
887                                   0)
888 lasso_weak1_nonzero_count <- sum(lasso_coef_df$Group == "weak_non" & lasso_coef_df$Coefficient !=
889                                   0)
890 lasso_weak2_nonzero_count <- sum(lasso_coef_df$Group == "weak_corr" & lasso_coef_df$Coefficient
891                                   != 0)
892
893 # calculate strong beta MSE
894 ## merge model coef and true beta
895 lasso_merged_df <- merge(b_true_df, lasso_coef_df, by = "Predictor", suffixes = c("_true", "_
896                                   lasso"))
897
898 lasso_strong_rows <- lasso_merged_df[lasso_merged_df$Group_true == "strong", ]
899
900 ## mse
901 lasso_mse_strong <- mean((lasso_strong_rows$Coefficient_true - lasso_strong_rows$Coefficient_
902                                   lasso)^2)
903

```

```

899 # corrlation
900 lasso_model_strong <- lasso_coef_df[lasso_coef_df$Group == "strong", ]
901
902 lasso_true_strong <- b_true_df[b_true_df$Group == "strong", ]
903
904 ## pearson
905 lasso_pearson_corr_strong <- cor(lasso_model_strong$Coefficient, lasso_true_strong$Coefficient,
906   method = "pearson")
907
908 ## spearman
909 lasso_spearman_corr_strong <- cor(lasso_model_strong$Coefficient, lasso_true_strong$Coefficient,
910   method = "spearman")
911
912 results[[i]] <- list(
913   lasso_mse_strong = lasso_mse_strong,
914   lasso_strong_nonzero_count = lasso_strong_nonzero_count,
915   lasso_weak1_nonzero_count = lasso_weak1_nonzero_count,
916   lasso_weak2_nonzero_count = lasso_weak2_nonzero_count,
917   lasso_pearson_corr_strong = lasso_pearson_corr_strong,
918   lasso_spearman_corr_strong = lasso_spearman_corr_strong
919 )
920 }
921
922 #result data
923 lasso_results_df <- as.data.frame(do.call(rbind, results))
924 lasso_results_df <-
925   lasso_results_df |>
926   mutate(lasso_mse_strong = as.numeric(lasso_mse_strong),
927     lasso_strong_nonzero_count = as.numeric(lasso_strong_nonzero_count),
928     lasso_weak1_nonzero_count = as.numeric(lasso_weak1_nonzero_count),
929     lasso_weak2_nonzero_count = as.numeric(lasso_weak2_nonzero_count),
930     lasso_pearson_corr_strong = as.numeric(lasso_pearson_corr_strong),
931     lasso_spearman_corr_strong = as.numeric(lasso_spearman_corr_strong))
932
933 mean_values <- colMeans(lasso_results_df)
934
935 #result data mean
936 res_lasso <- data.frame(
937   mse = mean_values["lasso_mse_strong"],
938   strong = mean_values["lasso_strong_nonzero_count"],
939   weak_non = mean_values["lasso_weak1_nonzero_count"],
940   weak_corr = mean_values["lasso_weak2_nonzero_count"],
941   pearson = mean_values["lasso_pearson_corr_strong"],

```

```

941 spearman = mean_values["lasso_spearman_corr_strong"],
942 model = "lasso"
943 )
944 rownames(res_lasso) <- NULL
945 '''
946
947 '''{r}
948 #t.test
949 p_values <- numeric()
950
951 col_indices <- seq_len(ncol(lasso_results_df))
952
953 #test for each col
954 for (i in col_indices) {
955   t_result <- t.test(results_df[[i]], lasso_results_df[[i]], paired = TRUE)
956   p_values[i] <- t_result$p.value
957 }
958
959 t_test_results_df <- data.frame(p_value = p_values)
960
961 t_test_results_df$col <- c("mse", "strong", "weak_non", "weak_corr", "pearson", "spearman")
962 t_test_results_df|>
963   knitr::kable()
964 '''
965
966
967 '''{r}
968 #visualization
969 res = rbind(res_FS, res_lasso)
970 res |>
971   knitr::kable()
972 res_long <- tidyr::gather(res, key = "variable", value = "value", -model)
973
974 p <- ggplot(res_long, aes(x = model, y = value, color = variable, group = variable)) +
975   geom_point() +
976   labs(x = "Model", y = "Value", color = "Variable")
977
978 p + geom_path(aes(group = variable), linetype = "dotted") +
979   scale_color_manual(values = c("strong" = "blue", "weak_non" = "green",
980                                "weak_corr" = "red", "pearson" = "purple",
981                                "spearman" = "orange",
982                                "mse" = "black"),
983                     labels = c("strong" = "Strong", "weak_non" = "weak and independent",
984                                "weak_corr" = "weak but correlated",

```

```
985         "pearson" = "Pearson", "spearman" = "Spearman",  
986         "mse" = "MSE"))+  
987     theme_minimal()  
988     ""
```