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.

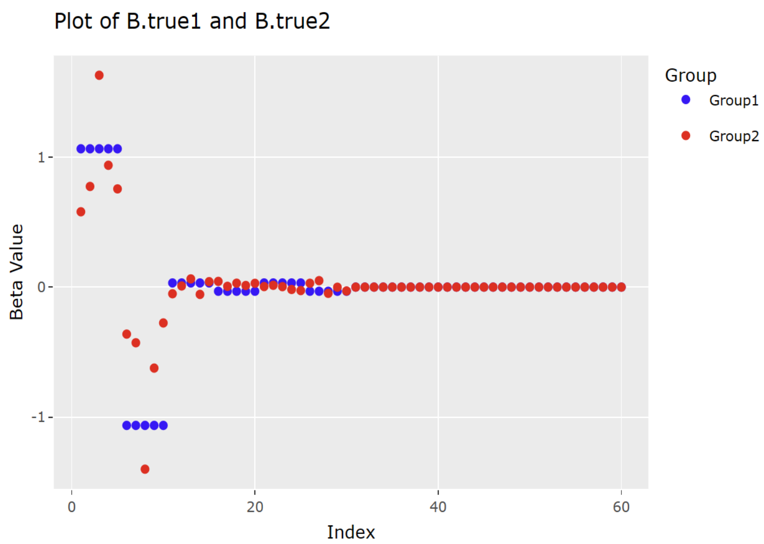

Figure 1: Two groups of true betas generated for the simulation

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.

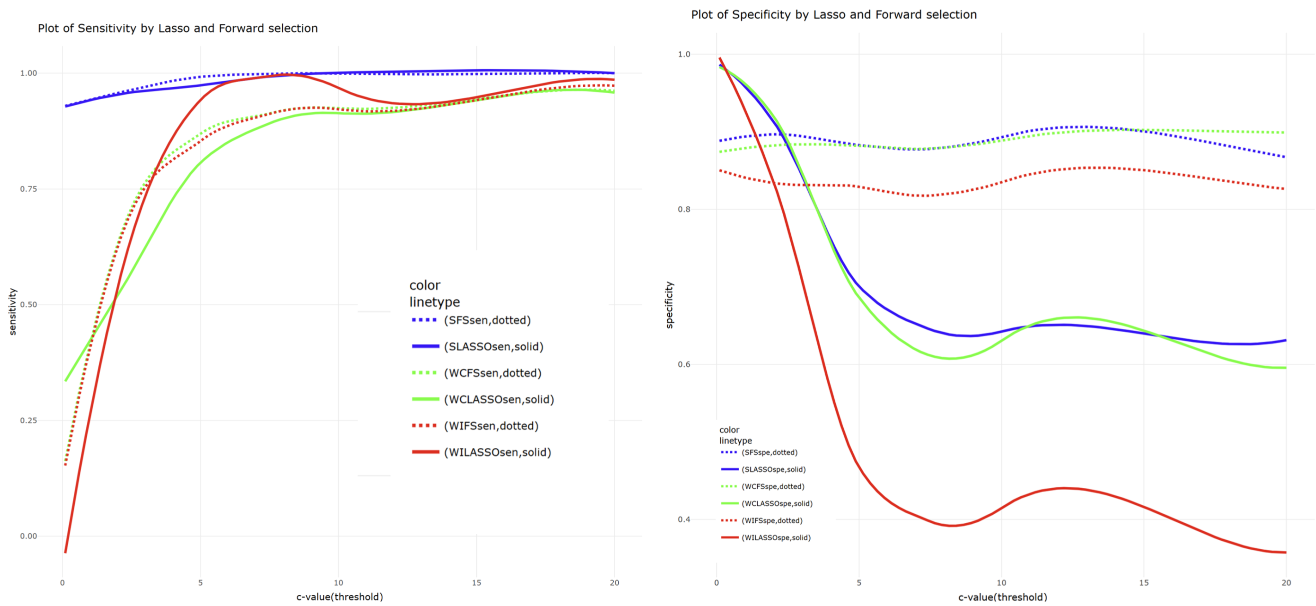

Figure 2: The sensitivity and the specificity by LASSO and forward selection

Examining the plot[1], 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,

[1]For 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.

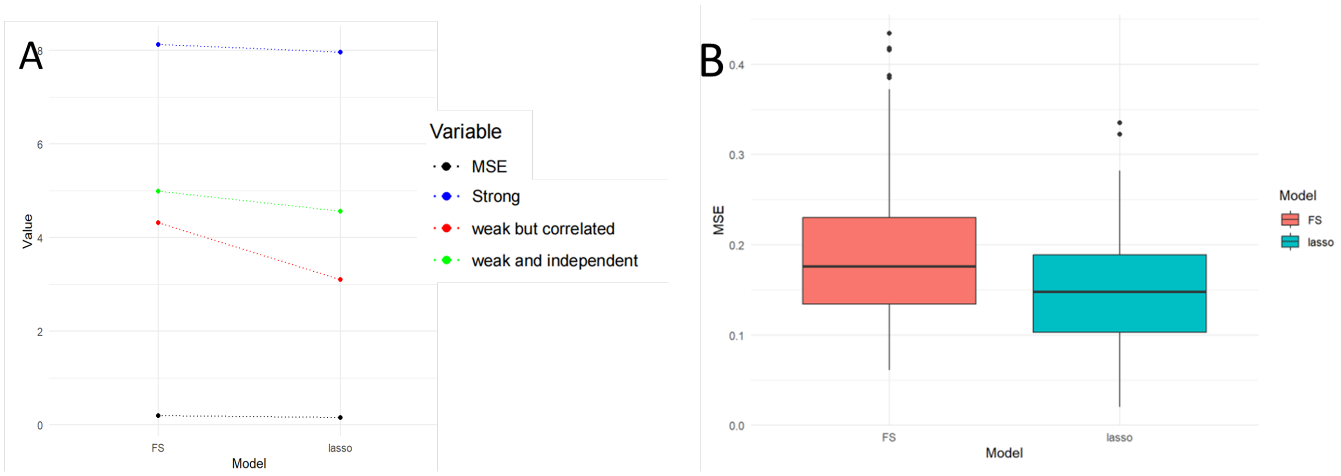

Figure 3: when C = 1, (A) The mean predictor counts and the mean MSE for LASSO and Forward selection model. (B) The distribution of the predicted parameters MSE of strong signals by two different models among 100 simulations.

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.

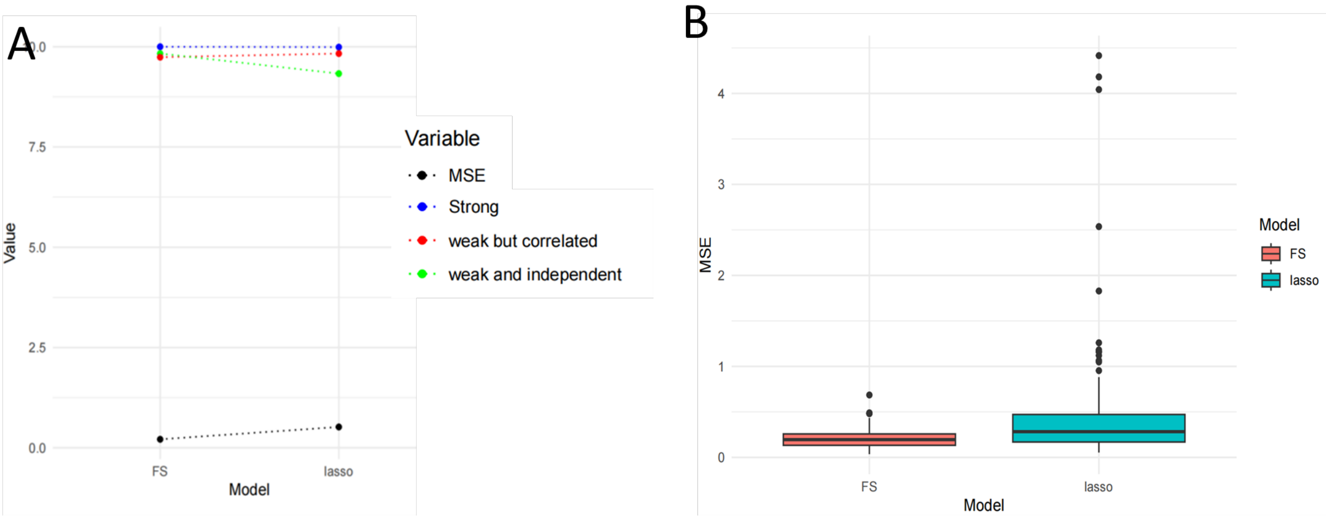

Figure 4: when C = 30, (A) The mean predictor counts and the mean MSE for LASSO and Forward selection model. (B) The distribution of the predicted parameters MSE of strong signals by two different models among 100 simulations.

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,

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

where $\widehat{y}_i$ 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:

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

where λ is a tunning 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 ith observation, $\hat{Y}_i$ is the predicted value, and n is the number of observations.

8.2 Code

```

\subsubsection{General setting for the first simulation test}

## Data Generation

```{r}
set.seed(1)
n <- 100
# 20+20+20
p <- 60

# thresh1
thr <- sqrt(log(p)/n)
thr

# X
# make sure that 1-20 and 41-60 are not correlated,
# which means they are independently generated
# generate 21-40 with weak-but-correlated signals

# Responding beta's are not 0
X1.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
X2.1 <- 3*X1.1+matrix(rnorm(n * p/3/2), n, p/3/2)
X3.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
# Responding beta's are 0
X1.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
X2.0 <- 3*X1.0+matrix(rnorm(n * p/3/2), n, p/3/2)
X3.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)

X<-cbind(X1.1, X2.1, X3.1, X1.0, X2.0, X3.0)



# beta



## positive and negative for the first half
```
```

````
b.true1.strong <- c(thr+abs(rnorm(5)),-thr-abs(rnorm(5)))
b.true1.weak1   <- runif(10,-thr,thr)
b.true1.weak2   <- runif(10,-thr,thr)
## zero for the second half
b.true0 <- rep(0,p/2)
## combine them together
b.true       <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
## name b.true
names(b.true) <- paste0("X", seq(1, 60))


# Y
Y <- 1 + X %*% b.true + rnorm(n)
df <- data.frame(cbind(X, Y))
names(df)[p + 1] <- "y"

cat("True non-zero effects:", which(b.true != 0), "\n")
## plot
plot(b.true)

```


# Plot for beta


```{r}
par(mfrow = c(1,2))
n <- 100
# 20+20+20
p <- 60

# thresh1
thr <- sqrt(log(p)/n)
thr

# beta
##positiveandnegativeforthefirsthalf
 b.true1.strong<-c(rep(thr+1,5),rep(-thr-1,5))
 b.true1.weak1 <-c(rep(thr/2,5),rep(-thr/2,5))
 b.true1.weak2 <-c(rep(thr/2,5),rep(-thr/2,5))
##zeroforthesecondhalf
 b.true0<-rep(0,p/2)

````

````
b.true1        <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
plot(b.true1)

# beta


## positive and negative for the first half
b.true1.strong <- c(thr+abs(rnorm(5,thr,1)),-thr-abs(rnorm(5,thr,1)))
b.true1.weak1    <- runif(10,-thr,thr)
b.true1.weak2    <- runif(10,-thr,thr)
## zero for the second half
b.true0 <- rep(0,p/2)
## combine them together
b.true2        <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
plot(b.true2)
```



```{r}

index<-c(1:60)
group1<-rep("Group1",60)
group2<-rep("Group2",60)
B.true1<-cbind(b.true1,index,group1)
B.true2<-cbind(b.true2,index,group2)
B<-rbind(B.true1,B.true2)

library(ggplot2)

# Assuming B is correctly created and is a data frame
B <- data.frame(B) # Ensuring B is a data frame
colnames(B) <- c("Value", "Index", "Group") # Naming columns for clarity

# Converting the appropriate columns to the correct data types
B$Index <- as.numeric(B$Index)
B$Group <- as.factor(B$Group)
B$Value <- as.numeric(B$Value)

# Plotting with ggplot2
p1<-ggplot(B, aes(x = Index, y = Value, color = Group)) +
  geom_point() + # Using points to plot the data
  ylim(-2,2)+
  labs(title = "Plot of B.true1 and B.true2", x = "Index", y = "Beta Value") +
````

````
  scale_color_manual(values = c("Group1" = "blue", "Group2" = "red")) # Customizing colors



library(htmltools)
library(plotly)

# Assuming B is your data frame set up correctly

# Create a ggplot2 object
p <- ggplot(B, aes(x = Index, y = Value, color = Group)) +
  geom_point() + # Using points to plot the data
  labs(title = "Plot of B.true1 and B.true2", x = "Index", y = "Beta Value") +
  scale_color_manual(values = c("Group1" = "blue", "Group2" = "red")) # Customizing colors

# Convert the ggplot2 object to a plotly object
p_plotly1 <- ggplotly(p)


```





## Forward Selection

```{r}
set.seed(1)
# Forward Selection(default setting is k=log(n) from AIC)
fit.forward <- step(object = lm(y ~ 1, data = df),
                    scope = formula(lm(y ~ ., data = df)),
                    direction = "forward", trace = 0) # AIC
summary(fit.forward)
# Selected ones


```





## LASSO

```{r}
````

````
# LASSO
fit.lasso <- cv.glmnet(X, Y, nfolds = 10, type.measure = "mse") # 5-fold CV using mean squared error
param.best <- fit.lasso$glmnet.fit$beta[, fit.lasso$lambda == fit.lasso$lambda.1se] # one standard-error rule
param.best[param.best != 0]
```



# (1) how well each of the two methods in identifying weak and strong predictors;

## Forward Selection


```{r}
FS_calculation <- function(b.true, selected_vars) {
  # Names of all variables
  all_vars <- names(b.true)

  # Convert b.true to a binary vector indicating whether each variable is non-zero (TRUE) or zero (FALSE)
  is_non_zero <- b.true != 0

  # Create a binary vector indicating whether each variable is selected (TRUE) or not (FALSE)
  is_selected <- all_vars %in% selected_vars

  # True Positives (TP): Non-zero variables that were selected
  TP <- sum(is_non_zero & is_selected)

  # False Negatives (FN): Non-zero variables that were not selected
  FN <- sum(is_non_zero & !is_selected)

  # True Negatives (TN): Zero variables that were not selected
  TN <- sum(!is_non_zero & !is_selected)

  # False Positives (FP): Zero variables that were selected
  FP <- sum(!is_non_zero & is_selected)

  # Calculate sensitivity and specificity
  sensitivity <- TP / (TP + FN)
  specificity <- TN / (TN + FP)

  list(sensitivity = sensitivity, specificity = specificity)
````

~~~
}



```


### Simulation for Forward Selection

```{r}
set.seed(2024)
### BREAD
# Calculate sensitivity and specificity
Strong_FS_sensitivity_sum <-
Strong_FS_specificity_sum <-
Weakcor_FS_sensitivity_sum <-
Weakcor_FS_specificity_sum <-
Weakind_FS_sensitivity_sum <-
Weakind_FS_specificity_sum <-
mse_FS<-0

for (i in 1:LOOP) {
# Data Generation
  # Responding beta's are not 0
X1.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
X2.1 <- 3*X1.1+matrix(rnorm(n * p/3/2), n, p/3/2)
X3.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
# Responding beta's are 0
X1.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
X2.0 <- 3*X1.0+matrix(rnorm(n * p/3/2), n, p/3/2)
X3.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)

X<-cbind(X1.1, X2.1, X3.1, X1.0, X2.0, X3.0)


# beta


## positive and negative for the first half
b.true1.strong <- c(thr+abs(rnorm(5)),-thr-abs(rnorm(5)))
b.true1.weak1   <- runif(10,-thr,thr)
b.true1.weak2   <- runif(10,-thr,thr)
## zero for the second half
b.true0 <- rep(0,p/2)
~~~

```
## combine them together
b.true        <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
## name b.true
names(b.true) <- paste0("X", seq(1, 60))


# Y
Y <- 1 + X %*% b.true + rnorm(n)
df <- data.frame(cbind(X, Y))
names(df)[p + 1] <- "y"

# Selection
fit.forward <- step(object = lm(y ~ 1, data = df),
                    scope = formula(lm(y ~ ., data = df)),
                    direction = "forward", trace = 0) # AIC


# Calculate MSE
predictions <- fit.forward$fitted.values
mse_FS <- mse_FS +  mean((Y - predictions)^2)

# Groups for the sensitivity and the specificity
selected_vars<-names(fit.forward$coefficients[-1])
FS_group1 <- FS_calculation(b.true[c(1:10,31:40)], selected_vars)
Strong_FS_sensitivity_sum  <- Strong_FS_sensitivity_sum + FS_group1$sensitivity
Strong_FS_specificity_sum  <- Strong_FS_specificity_sum + FS_group1$specificity

FS_group2 <- FS_calculation(b.true[c(11:20,41:50)], selected_vars)
Weakcor_FS_sensitivity_sum  <- Weakcor_FS_sensitivity_sum + FS_group2$sensitivity
Weakcor_FS_specificity_sum  <- Weakcor_FS_specificity_sum + FS_group2$specificity

FS_group3 <- FS_calculation(b.true[c(21:30,51:60)], selected_vars)
Weakind_FS_sensitivity_sum  <- Weakind_FS_sensitivity_sum + FS_group3$sensitivity
Weakind_FS_specificity_sum  <- Weakind_FS_specificity_sum + FS_group3$specificity

}

### BREAD
Strong_FS_sensitivity = Strong_FS_sensitivity_sum/LOOP
Strong_FS_sensitivity
Strong_FS_specificity = Strong_FS_specificity_sum/LOOP
Strong_FS_specificity

Weakcor_FS_sensitivity  = Weakcor_FS_sensitivity_sum/LOOP
```

````
Weakcor_FS_sensitivity
Weakcor_FS_specificity  = Weakcor_FS_specificity_sum/LOOP
Weakcor_FS_specificity

Weakind_FS_sensitivity  = Weakind_FS_sensitivity_sum/LOOP
Weakind_FS_sensitivity
Weakind_FS_specificity  = Weakind_FS_specificity_sum/LOOP
Weakind_FS_specificity

mse_FS/LOOP
```



## LASSO



```{r}
# calculate sensitivity and specificity
LASSO_calculation <- function(selected_coefs, non_zero_indices, zero_indices) {
  true_positives <- sum(selected_coefs[non_zero_indices] != 0)
  true_negatives <- sum(selected_coefs[zero_indices] == 0)
  false_negatives <- sum(selected_coefs[non_zero_indices] == 0)
  false_positives <- sum(selected_coefs[zero_indices] != 0)

  sensitivity <- true_positives / (true_positives + false_negatives)
  specificity <- true_negatives / (true_negatives + false_positives)

  return(list(sensitivity = sensitivity, specificity = specificity))
}
```

### Simulation for Forward Selection


```{r}
set.seed(2024)

### BREAD
# Calculate sensitivity and specificity
Strong_LASSO_sensitivity_sum <-
Strong_LASSO_specificity_sum <-
Weakcor_LASSO_sensitivity_sum <-
````

```
Weakcor_LASSO_specificity_sum <-
Weakind_LASSO_sensitivity_sum <-
Weakind_LASSO_specificity_sum <-
  mse_LASSO<-0

for (i in 1:LOOP) {
# Data Generation
  # Responding beta's are not 0
X1.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
X2.1 <- 3*X1.1+matrix(rnorm(n * p/3/2), n, p/3/2)
X3.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
# Responding beta's are 0
X1.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
X2.0 <- 3*X1.0+matrix(rnorm(n * p/3/2), n, p/3/2)
X3.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)

X<-cbind(X1.1, X2.1, X3.1, X1.0, X2.0, X3.0)


# beta


## positive and negative for the first half
b.true1.strong <- c(thr+abs(rnorm(5)),-thr-abs(rnorm(5)))
b.true1.weak1   <- runif(10,-thr,thr)
b.true1.weak2   <- runif(10,-thr,thr)
## zero for the second half
b.true0 <- rep(0,p/2)
## combine them together
b.true       <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
## name b.true
names(b.true) <- paste0("X", seq(1, 60))


# Y
Y <- 1 + X %*% b.true + rnorm(n)
df <- data.frame(cbind(X, Y))
names(df)[p + 1] <- "y"

# Selection
# LASSO
fit.lasso <- cv.glmnet(X, Y, nfolds = 10, type.measure = "mse") # 5-fold CV using mean squared error
param.best <- fit.lasso$glmnet.fit$beta[, fit.lasso$lambda == fit.lasso$lambda.1se] # one standard
```

```
    -error rule
param.best[param.best != 0]


# Calculate MSE
predictions <- predict(fit.lasso, s = fit.lasso$lambda.1se, newx = as.matrix(X))
mse_LASSO <- mse_LASSO+mean((Y - predictions)^2)




# calculate SS for each group
LASSO_group1 <- LASSO_calculation(param.best, 1:10, 31:40)
Strong_LASSO_sensitivity_sum  <- Strong_LASSO_sensitivity_sum + LASSO_group1$sensitivity
Strong_LASSO_specificity_sum  <- Strong_LASSO_specificity_sum + LASSO_group1$specificity

LASSO_group2 <- LASSO_calculation(param.best, 11:20, 41:50)
Weakcor_LASSO_sensitivity_sum  <- Weakcor_LASSO_sensitivity_sum + LASSO_group2$sensitivity
Weakcor_LASSO_specificity_sum  <- Weakcor_LASSO_specificity_sum + LASSO_group2$specificity

LASSO_group3 <- LASSO_calculation(param.best, 21:30, 51:60)
Weakind_LASSO_sensitivity_sum  <- Weakind_LASSO_sensitivity_sum + LASSO_group3$sensitivity
Weakind_LASSO_specificity_sum  <- Weakind_LASSO_specificity_sum + LASSO_group3$specificity


}

### BREAD
Strong_LASSO_sensitivity = Strong_LASSO_sensitivity_sum/LOOP
Strong_LASSO_sensitivity
Strong_LASSO_specificity = Strong_LASSO_specificity_sum/LOOP
Strong_LASSO_specificity

Weakcor_LASSO_sensitivity  = Weakcor_LASSO_sensitivity_sum/LOOP
Weakcor_LASSO_sensitivity
Weakcor_LASSO_specificity  = Weakcor_LASSO_specificity_sum/LOOP
Weakcor_LASSO_specificity

Weakind_LASSO_sensitivity  = Weakind_LASSO_sensitivity_sum/LOOP
Weakind_LASSO_sensitivity
Weakind_LASSO_specificity  = Weakind_LASSO_specificity_sum/LOOP
Weakind_LASSO_specificity

mse_LASSO/LOOP
```

~~~~
```


\subsubsection{Dashboard for question 1}

```{r setup, include=FALSE}
library(flexdashboard)
library(tidyverse)
library(plotly)
# Install thematic and un-comment for themed static plots (i.e., ggplot2)
# thematic::thematic_rmd()
load("mse_sen224.RData")
load("mse_spe224.RData")
```

Column {data-width=650 .tabset}
-----------------------------------------------------------------------

### Sensitivity

```{r,echo=FALSE}
mse_sen_new <- mse_sen|>
  pivot_longer(
      c('SLASSOsen','WCLASSOsen','WILASSOsen',
      'SFSsen','WCFSsen','WIFSsen'),
       names_to = "Sensitivity", values_to = "values")
data_sen<- data.frame(
  x = mse_sen_new$c,
  y = mse_sen_new$values,
  group = factor(rep(1:6, 8)),
  color = rep(c("SLASSOsen", "WCLASSOsen", "WILASSOsen", "SFSsen", "WCFSsen", "WIFSsen"), 8),
  linetype = rep(c("solid", "solid","solid","dotted","dotted","dotted"), 8)
)

# Plot
p2 <- ggplot(data_sen, aes(x=x, y=y, color=color, group=group, linetype=linetype)) +
  geom_smooth(size=1,se = FALSE) +
  scale_color_manual(values=c("SLASSOsen"="blue", "SFSsen"="blue",
                             "WCLASSOsen"="green", "WCFSsen"="green",
                             "WILASSOsen"="red", "WIFSsen"="red")) +
  labs(title="Plot of Sensitivity by Lasso and Forward selection", x="c-value(threshold)", y="
    sensitivity")+
  scale_linetype_manual(values=c("solid"="solid", "dotted"="dotted")) +
  theme_minimal() +
~~~~

~~~~
  theme(legend.title=element_blank())+
  guides(size=FALSE)

 p_plotly2 <- ggplotly(p2)
 p_plotly2

```




### Specificity


```{r,echo=FALSE}
mse_spe_new <- mse_spe|>
  pivot_longer(
       c('SLASSOspe','WCLASSOspe','WILASSOspe',
      'SFSspe','WCFSspe','WIFSspe'),
       names_to = "Specificity", values_to = "values")
data_spe_new<- data.frame(
  x = mse_spe_new$c,
  y = mse_spe_new$values,
  group = factor(rep(1:6, 8)),
  color = rep(c("SLASSOspe", "WCLASSOspe", "WILASSOspe", "SFSspe", "WCFSspe", "WIFSspe"), 8),
  linetype = rep(c("solid", "solid","solid","dotted","dotted","dotted"), 8)
)

# Plot
p3 <- ggplot(data_spe_new, aes(x=x, y=y, color=color, group=group, linetype=linetype)) +
  geom_smooth(size=1,se = FALSE) +
  scale_color_manual(values=c("SLASSOspe"="blue", "SFSspe"="blue",
                              "WCLASSOspe"="green", "WCFSspe"="green",
                              "WILASSOspe"="red", "WIFSspe"="red")) +
    labs(title="Plot of Specificity by Lasso and Forward selection", x="c-value(threshold)", y="
    specificity")+
  scale_linetype_manual(values=c("solid"="solid", "dotted"="dotted")) +
  theme_minimal() +
  theme(legend.title=element_blank())+
  guides(size=FALSE)

p_plotly3 <- ggplotly(p3)
p_plotly3

~~~~

````
```


Column {data-width=350}
-----------------------------------------------------------------------

### Other Information


Abbreviation

S - Strong signals

WC - Weak but correlated signals

WI - Weak and independent signals

FS - Forward selection

spe - specificity

sen - sensitivity


eg:
'SLASSOspe' stands for the specificity of selecting strong signals by LASSO.

Definition of strong signals ---

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

Definition of weak-but-correlated signals  ---

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



Definition of weak-and-independent signals  ---

$$S_3=\{j: 0<|\beta_j|\le c\sqrt{log (p) / n},\mbox{ some } c>0, \mbox{corr}(X_j, X_j')= 0, \mbox{for all } j'\in S_1,  1\le j \le p\}$$

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

```
### Beta Generation

```{r, include=FALSE}
n <- 1000
# 20+20+20
p <- 60

# thresh1
thr <- sqrt(log(p)/n)
thr

# beta
##positiveandnegativeforthefirsthalf
 b.true1.strong<-c(rep(thr+1,5),rep(-thr-1,5))
 b.true1.weak1 <-c(rep(thr/2,5),rep(-thr/2,5))
 b.true1.weak2 <-c(rep(thr/2,5),rep(-thr/2,5))
##zeroforthesecondhalf
 b.true0<-rep(0,p/2)

b.true1       <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)


# beta


## positive and negative for the first half
b.true1.strong <- c(thr+abs(rnorm(5,thr,1)),-thr-abs(rnorm(5,thr,1)))
b.true1.weak1   <- runif(10,-thr,thr)
b.true1.weak2   <- runif(10,-thr,thr)
## zero for the second half
b.true0 <- rep(0,p/2)
## combine them together
b.true2       <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)

```



```{r, echo=FALSE}

index<-c(1:60)
group1<-rep("Group1",60)
group2<-rep("Group2",60)
```
```

```
B.true1<-cbind(b.true1,index,group1)
B.true2<-cbind(b.true2,index,group2)
B<-rbind(B.true1,B.true2)


# Assuming B is correctly created and is a data frame
B <- data.frame(B) # Ensuring B is a data frame
colnames(B) <- c("Value", "Index", "Group") # Naming columns for clarity

# Converting the appropriate columns to the correct data types
B$Index <- as.numeric(B$Index)
B$Group <- as.factor(B$Group)
B$Value <- as.numeric(B$Value)



# Assuming B is your data frame set up correctly

# Create a ggplot2 object
p1 <- ggplot(B, aes(x = Index, y = Value, color = Group)) +
  geom_point() + # Using points to plot the data
  labs(title = "Plot of B.true1 and B.true2", x = "Index", y = "Beta Value") +
  scale_color_manual(values = c("Group1" = "blue", "Group2" = "red")) # Customizing colors

# Convert the ggplot2 object to a plotly object
p_plotly1 <- ggplotly(p1)
p_plotly1

```


\subsubsection{Simulation for question 2}
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
require(survival)
require(quantreg)
require(glmnet)
require(MASS)
require(pROC)

LOOP = 100
```

````
n <- 100
# 20+20+20
p <- 60
c = 30

# thresh1
thr <- c * sqrt(log(p)/n)
```


# For foward selection

```{r}
#function of calculating effect of missing weak beta to strong beta for FS
FS_weak_effect <- function(model){
  # model coef data
  coef_names <- names(coef(model))
  coef_values <- coef(model)

  FS_coef_df <- data.frame(Predictor = b_true_df$Predictor,
                           Coefficient = 0,
                           Group = b_true_df$Group)

  FS_coef_df$Coefficient <- ifelse(FS_coef_df$Predictor %in% coef_names,
                                   coef_values[match(FS_coef_df$Predictor, coef_names)],
                                   0)
  return(FS_coef_df)
}
```

```{r}
set.seed(2024)
mse_strong <-
  strong_nonzero_count <-
  weak1_nonzero_count <-
  weak2_nonzero_count <-
  pearson_corr_strong <-
  spearman_corr_strong <- 0

results = list()
#repeat 100 times for FS
for (i in 1:LOOP) {
  # Data Generation
  # Responding beta's are not 0
````

```
X1.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
X2.1 <- 3*X1.1+matrix(rnorm(n * p/3/2), n, p/3/2)
X3.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
# Responding beta's are 0
X1.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
X2.0 <- 3*X1.0+matrix(rnorm(n * p/3/2), n, p/3/2)
X3.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)

X<-cbind(X1.1, X2.1, X3.1, X1.0, X2.0, X3.0)

# beta
## positive and negative for the first half
b.true1.strong <- c(thr+abs(rnorm(5)),-thr-abs(rnorm(5)))
b.true1.weak1   <- runif(10,-thr,thr)
b.true1.weak2   <- runif(10,-thr,thr)
## zero for the second half
b.true0 <- rep(0,p/2)
## combine them together
b.true       <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
## name b.true
names(b.true) <- paste0("X", seq(1, 60))

# Y
Y <- 1 + X %*% b.true + rnorm(n)
df <- data.frame(cbind(X, Y))
names(df)[p + 1] <- "y"

# true beta dataframe
b_true_df <- data.frame(Predictor = names(b.true), Coefficient = as.numeric(b.true))
b_true_df$Coefficient
b_true_df$Group <- ifelse(b_true_df$Coefficient %in% b.true1.strong, "strong",
                        ifelse(b_true_df$Coefficient %in% b.true1.weak1, "weak_non",
                               ifelse(b_true_df$Coefficient %in% b.true1.weak2,
                                      "weak_corr", "zero")))

# Selection
fit.forward <- step(object = lm(y ~ 1, data = df),
                    scope = formula(lm(y ~ ., data = df)),
                    direction = "forward", trace = 0) # AIC

# model coef data
FS_coef_df = FS_weak_effect(fit.forward)

# number of beta
```

```r
  strong_nonzero_count <- sum(FS_coef_df$Group == "strong" & FS_coef_df$Coefficient != 0)
  weak1_nonzero_count <- sum(FS_coef_df$Group == "weak_non" & FS_coef_df$Coefficient != 0)
  weak2_nonzero_count <- sum(FS_coef_df$Group == "weak_corr" & FS_coef_df$Coefficient != 0)

  # calculate strong beta MSE
  ## merge model coef and true beta
  merged_df <- merge(b_true_df, FS_coef_df, by = "Predictor", suffixes = c("_true", "_FS"))

  strong_rows <- merged_df[merged_df$Group_true == "strong", ]

  ## mse
  mse_strong <- mean((strong_rows$Coefficient_true - strong_rows$Coefficient_FS)^2)

  # corrlation
  model_strong <- FS_coef_df[FS_coef_df$Group == "strong", ]

  true_strong <- b_true_df[b_true_df$Group == "strong", ]

  ## pearson
  pearson_corr_strong <- cor(model_strong$Coefficient, true_strong$Coefficient, method =   "
    pearson")

  ## spearman
  spearman_corr_strong <- cor(model_strong$Coefficient, true_strong$Coefficient, method =   "
    spearman")

  results[[i]] <- list(
    mse_strong  = mse_strong ,
    strong_nonzero_count  = strong_nonzero_count ,
    weak1_nonzero_count  = weak1_nonzero_count ,
    weak2_nonzero_count  = weak2_nonzero_count ,
    pearson_corr_strong  = pearson_corr_strong ,
    spearman_corr_strong  = spearman_corr_strong
  )
}

#result data
results_df <- as.data.frame(do.call(rbind, results))
results_df <-
  results_df |>
  mutate(mse_strong = as.numeric(mse_strong),
         strong_nonzero_count = as.numeric(strong_nonzero_count),
         weak1_nonzero_count = as.numeric(weak1_nonzero_count),
         weak2_nonzero_count = as.numeric(weak2_nonzero_count),
```

````
         pearson_corr_strong = as.numeric(pearson_corr_strong),
         spearman_corr_strong = as.numeric(spearman_corr_strong))

FS_mean_values <- colMeans(results_df)

#result data mean
res_FS <- data.frame(
  mse = FS_mean_values["mse_strong"],
  strong = FS_mean_values["strong_nonzero_count"],
  weak_non = FS_mean_values["weak1_nonzero_count"],
  weak_corr = FS_mean_values["weak2_nonzero_count"],
  pearson = FS_mean_values["pearson_corr_strong"],
  spearman = FS_mean_values["spearman_corr_strong"],
  model = "FS"
)
rownames(res_FS) <- NULL
```

# For lasso
```{r}
# LASSO
fit.lasso <- cv.glmnet(X, Y, nfolds = 10, type.measure = "mse") # 5-fold CV using mean squared error
param.best <- fit.lasso$glmnet.fit$beta[, fit.lasso$lambda == fit.lasso$lambda.1se] # one standard-error rule
param = param.best[param.best != 0]
```



```{r}
#function of calculating effect of missing weak beta to strong beta for lasso
lasso_weak_effect <- function(param){

  # model coef data
  coef_names <- gsub("V", "X", names(param))
  formula_str <- paste("y ~", paste(coef_names, collapse = " + "))
  slm <- lm(formula_str, data = df)

  recoef_names <- names(coef(slm))
  recoef_values <- coef(slm)

  lasso_coef_df <- data.frame(Predictor = b_true_df$Predictor,
                              Coefficient = 0,
                              Group = b_true_df$Group)
````

```
  lasso_coef_df$Coefficient <- ifelse(lasso_coef_df$Predictor %in% recoef_names,
                              recoef_values[match(lasso_coef_df$Predictor, recoef_names)],
                              0)
  return(lasso_coef_df)
}
```

```{r}
set.seed(2024)
lasso_mse_strong <-
  lasso_strong_nonzero_count <-
  lasso_weak1_nonzero_count <-
  lasso_weak2_nonzero_count <-
  lasso_pearson_corr_strong <-
  lasso_spearman_corr_strong <- 0
results = list()

#repeat 100 times for FS
for (i in 1:LOOP) {
  # Data Generation
  # Responding beta's are not 0
  X1.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
  X2.1 <- 3*X1.1+matrix(rnorm(n * p/3/2), n, p/3/2)
  X3.1 <- matrix(rnorm(n * p/3/2), n, p/3/2)
  # Responding beta's are 0
  X1.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)
  X2.0 <- 3*X1.0+matrix(rnorm(n * p/3/2), n, p/3/2)
  X3.0 <- matrix(rnorm(n * p/3/2), n, p/3/2)

  X<-cbind(X1.1, X2.1, X3.1, X1.0, X2.0, X3.0)

  # beta
  ## positive and negative for the first half
  b.true1.strong <- c(thr+abs(rnorm(5)),-thr-abs(rnorm(5)))
  b.true1.weak1   <- runif(10,-thr,thr)
  b.true1.weak2   <- runif(10,-thr,thr)
  ## zero for the second half
  b.true0 <- rep(0,p/2)
  ## combine them together
  b.true       <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
  ## name b.true
  names(b.true) <- paste0("X", seq(1, 60))

```

```r
# Y
Y <- 1 + X %*% b.true + rnorm(n)
df <- data.frame(cbind(X, Y))
names(df)[p + 1] <- "y"

# true beta dataframe
b_true_df <- data.frame(Predictor = names(b.true), Coefficient = as.numeric(b.true))
b_true_df$Coefficient
b_true_df$Group <- ifelse(b_true_df$Coefficient %in% b.true1.strong, "strong",
                         ifelse(b_true_df$Coefficient %in% b.true1.weak1, "weak_non",
                                ifelse(b_true_df$Coefficient %in% b.true1.weak2,
                                       "weak_corr", "zero")))

# Selection
# LASSO
# 5-fold CV using mean   squared error
fit.lasso <- cv.glmnet(X, Y, nfolds = 10, type.measure = "mse")
# one standard-error rule
param.best <- fit.lasso$glmnet.fit$beta[, fit.lasso$lambda == fit.lasso$lambda.1se]
#selected model paramters
best_param <- param.best[param.best != 0]

# model coef data
lasso_coef_df = lasso_weak_effect(best_param)

# number of beta
lasso_strong_nonzero_count <- sum(lasso_coef_df$Group == "strong" & lasso_coef_df$Coefficient !=
  0)
lasso_weak1_nonzero_count <- sum(lasso_coef_df$Group == "weak_non" & lasso_coef_df$Coefficient !
  = 0)
lasso_weak2_nonzero_count <- sum(lasso_coef_df$Group == "weak_corr" & lasso_coef_df$Coefficient
  != 0)

# calculate strong beta MSE
## merge model coef and true beta
lasso_merged_df <- merge(b_true_df, lasso_coef_df, by = "Predictor", suffixes = c("_true", "_
  lasso"))

lasso_strong_rows <- lasso_merged_df[lasso_merged_df$Group_true == "strong", ]

## mse
lasso_mse_strong <- mean((lasso_strong_rows$Coefficient_true - lasso_strong_rows$Coefficient_
  lasso)^2)

```

```
  # corrlation
  lasso_model_strong <- lasso_coef_df[lasso_coef_df$Group == "strong", ]

  lasso_true_strong <- b_true_df[b_true_df$Group == "strong", ]

  ## pearson
  lasso_pearson_corr_strong <- cor(lasso_model_strong$Coefficient, lasso_true_strong$Coefficient,
    method =  "pearson")

  ## spearman
  lasso_spearman_corr_strong <- cor(lasso_model_strong$Coefficient, lasso_true_strong$Coefficient,
     method =   "spearman")

  results[[i]] <- list(
    lasso_mse_strong = lasso_mse_strong,
    lasso_strong_nonzero_count = lasso_strong_nonzero_count,
    lasso_weak1_nonzero_count = lasso_weak1_nonzero_count,
    lasso_weak2_nonzero_count = lasso_weak2_nonzero_count,
    lasso_pearson_corr_strong = lasso_pearson_corr_strong,
    lasso_spearman_corr_strong = lasso_spearman_corr_strong
  )

}

#result data
lasso_results_df <- as.data.frame(do.call(rbind, results))
lasso_results_df <-
  lasso_results_df |>
  mutate(lasso_mse_strong = as.numeric(lasso_mse_strong),
         lasso_strong_nonzero_count = as.numeric(lasso_strong_nonzero_count),
         lasso_weak1_nonzero_count = as.numeric(lasso_weak1_nonzero_count),
         lasso_weak2_nonzero_count = as.numeric(lasso_weak2_nonzero_count),
         lasso_pearson_corr_strong = as.numeric(lasso_pearson_corr_strong),
         lasso_spearman_corr_strong = as.numeric(lasso_spearman_corr_strong))

mean_values <- colMeans(lasso_results_df)

#result data mean
res_lasso <- data.frame(
  mse = mean_values["lasso_mse_strong"],
  strong = mean_values["lasso_strong_nonzero_count"],
  weak_non = mean_values["lasso_weak1_nonzero_count"],
  weak_corr = mean_values["lasso_weak2_nonzero_count"],
  pearson = mean_values["lasso_pearson_corr_strong"],
```

```
  spearman = mean_values["lasso_spearman_corr_strong"],
  model = "lasso"
)
rownames(res_lasso) <- NULL
```

```{r}
#t.test
p_values <- numeric()

col_indices <- seq_len(ncol(lasso_results_df))

#test for each col
for (i in col_indices) {
  t_result <- t.test(results_df[[i]], lasso_results_df[[i]], paired = TRUE)
  p_values[i] <- t_result$p.value
}

t_test_results_df <- data.frame(p_value = p_values)

t_test_results_df$col <- c("mse", "strong", "weak_non", "weak_corr", "pearson", "spearman")
t_test_results_df|>
  knitr::kable()
```


```{r}
#visualization
res = rbind(res_FS,res_lasso)
res |>
  knitr::kable()
res_long <- tidyr::gather(res, key = "variable", value = "value", -model)

p <- ggplot(res_long, aes(x = model, y = value, color = variable, group = variable)) +
  geom_point() +
  labs(x = "Model", y = "Value", color = "Variable")

p + geom_path(aes(group = variable), linetype = "dotted") +
  scale_color_manual(values = c("strong" = "blue", "weak_non" = "green",
                                "weak_corr" = "red", "pearson" = "purple",
                                "spearman" = "orange",
                                "mse" = "black"),
                     labels = c("strong" = "Strong", "weak_non" = "weak and independent",
                                "weak_corr" = "weak but correlated",
```

````
                                          "pearson" = "Pearson", "spearman" = "Spearman",
                                          "mse" = "MSE"))+
  theme_minimal()
```
````
````