Group Projects on Monte Carlo Simulation Design.

P8160 Advanced Statistical Computing

P1

Data Generation

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

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

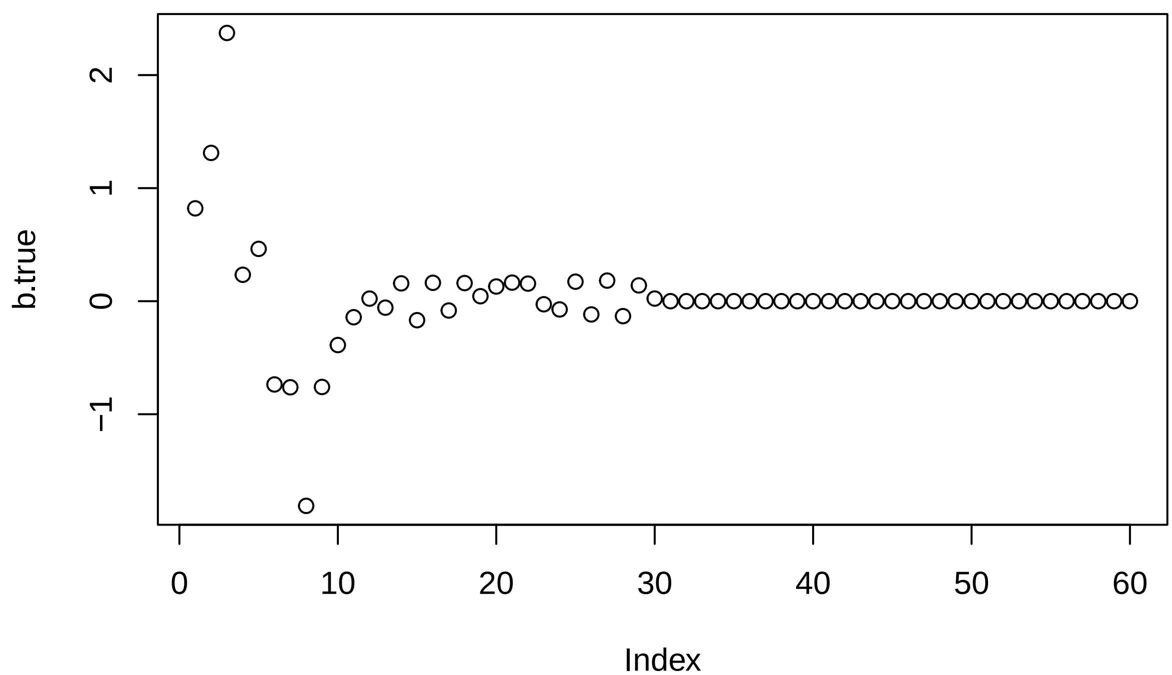
```
## [1] 0.2023449
```

```r
# 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)
```

```r
## 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")
```

```
## True non-zero effects: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
```

```r
## plot
plot(b.true)
```



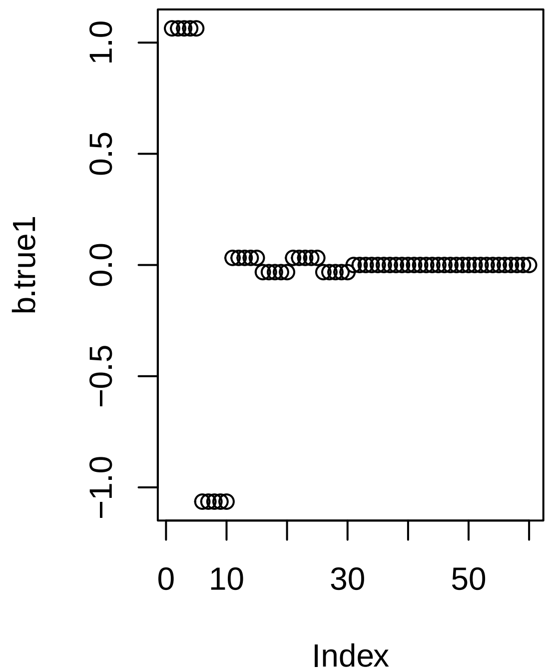
Plot for beta

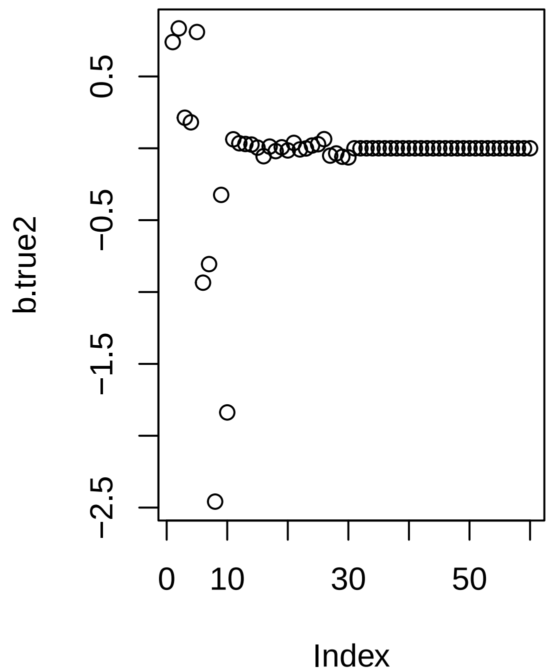
```r
par(mfrow = c(1,2))
n <- 1000
```

```r
# 20+20+20
p <- 60

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

```
## [1] 0.06398707
```

```r
# 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)
```






```
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
```

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

P2

Forward Selection

```r
FS_calculation <- function(b.true, selected_vars) {
  # Names of all variables
  all_vars <- names(b.true)
  is_non_zero <- b.true != 0
  is_selected <- all_vars %in% selected_vars

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

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

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

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)

c <- c(0.1,0.5,1,2,5,7.5,10)
LEN <- length(c)
```

```
MSEFS <- SFSsen<-SFSspe<-WCFSsen<- WCFSspe<- WIFSsen<- WIFSspe<-rep(0,LEN)

for (k in 1:LEN) {



### 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(c[k]*thr+abs(rnorm(5)),-c[k]*thr-abs(rnorm(5)))
b.true1.weak1   <- runif(10,-c[k]*thr,c[k]*thr)
b.true1.weak2   <- runif(10,-c[k]*thr,c[k]*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),
```

```r
                    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

MSEFS[k]<-mse_FS/LOOP
SFSsen[k]<-Strong_FS_sensitivity
SFSspe[k]<-Strong_FS_specificity
WCFSsen[k]<-Weakcor_FS_sensitivity
WCFSspe[k]<-Weakcor_FS_specificity
WIFSsen[k]<-Weakind_FS_sensitivity
WIFSspe[k]<-Weakind_FS_specificity


}
```

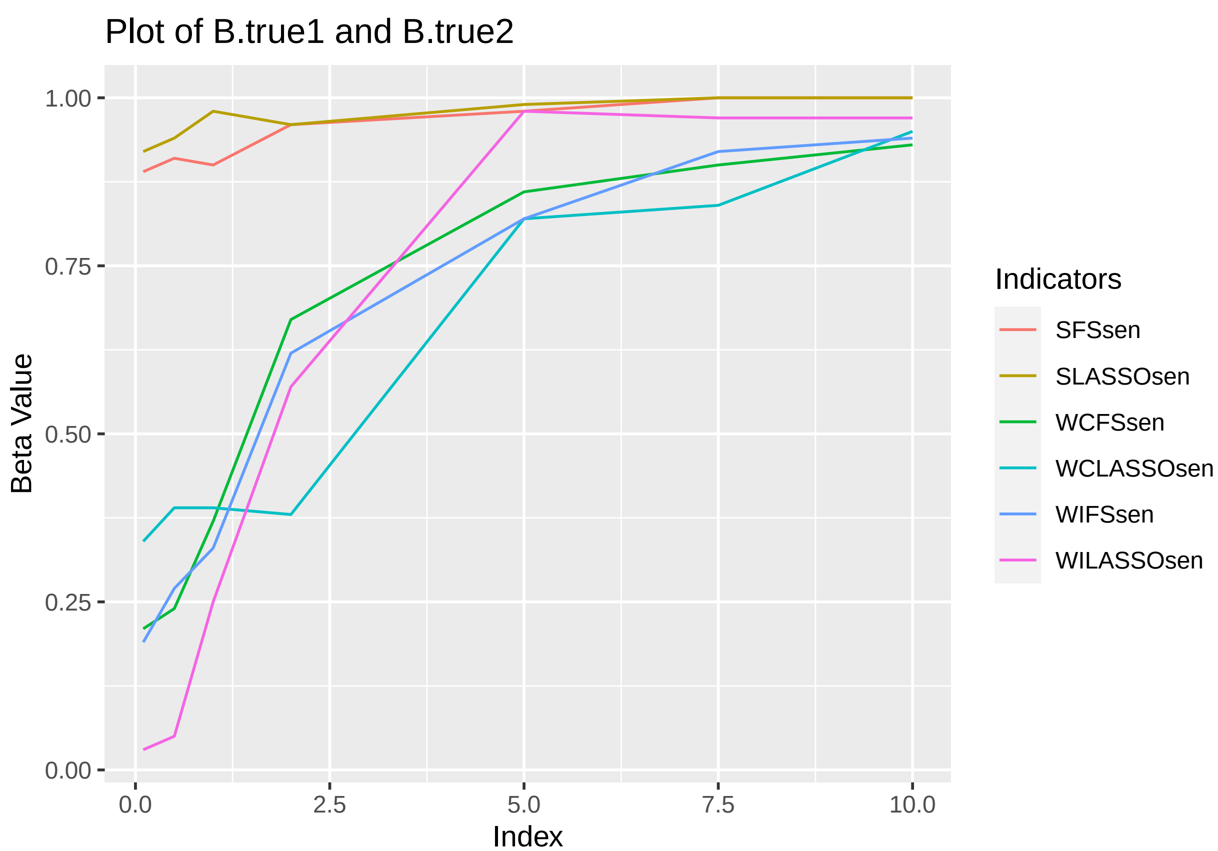
Simulation for Forward Selection

```r
set.seed(2024)

MSELASSO <- SLASSOsen<-SLASSOspe<-WCLASSOsen<- WCLASSOspe<- WILASSOsen<- WILASSOspe<-rep(0,LEN)

for (k in 1:LEN) {



### 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*c[k]+abs(rnorm(5)),-thr*c[k]-abs(rnorm(5)))
b.true1.weak1   <- runif(10,-thr*c[k],thr*c[k])
b.true1.weak2   <- runif(10,-thr*c[k],thr*c[k])
## 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"
```

```r
# 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-erro
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

MSELASSO[k]<-mse_LASSO/LOOP
SLASSOsen[k]<-Strong_LASSO_sensitivity
SLASSOspe[k]<-Strong_LASSO_specificity
WCLASSOsen[k]<-Weakcor_LASSO_sensitivity
WCLASSOspe[k]<-Weakcor_LASSO_specificity
WILASSOsen[k]<-Weakind_LASSO_sensitivity
WILASSOspe[k]<-Weakind_LASSO_specificity
```

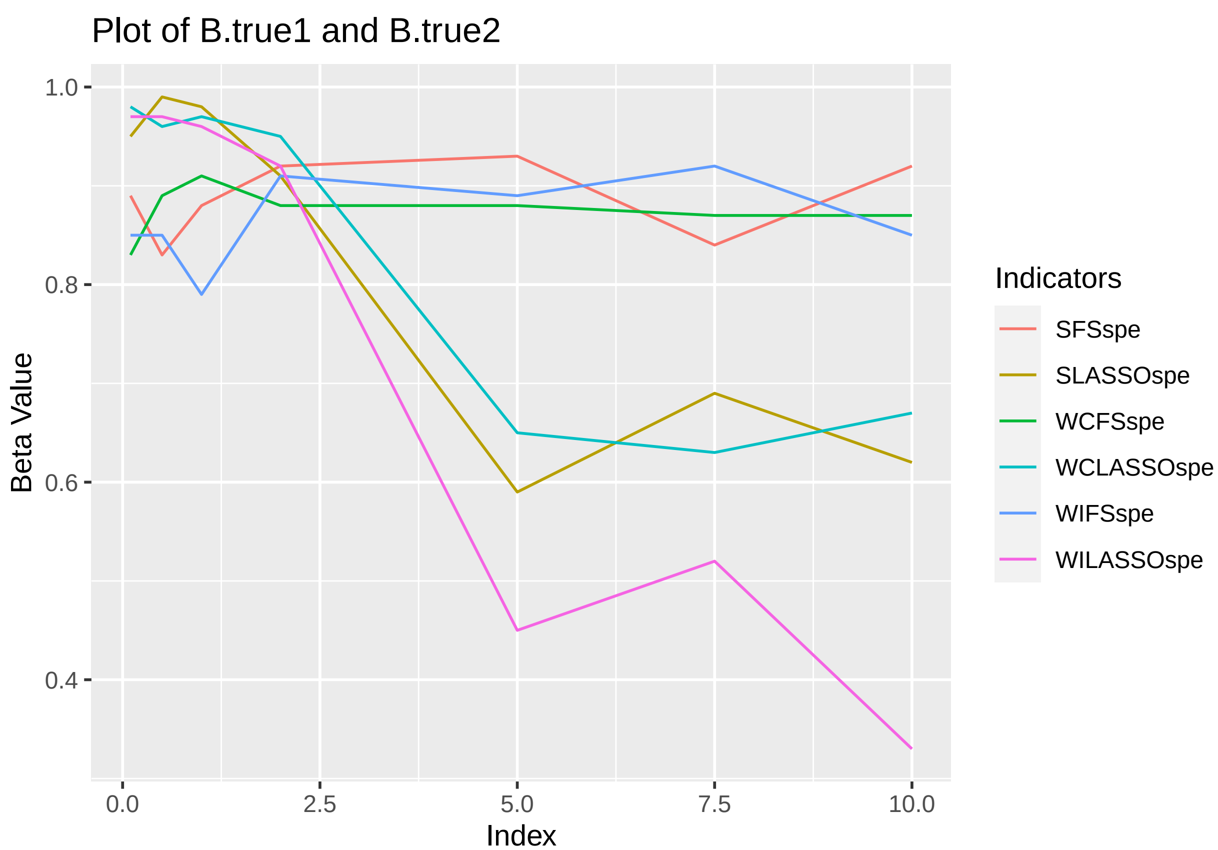
```
}
```

```
mse_sen<-as.data.frame(cbind(c,SLASSOsen,WCLASSOsen,WILASSOsen,
      SFSsen,WCFSsen,WIFSsen))
mse_sen$c <- as.numeric(mse_sen$c)
mse_sen <- mse_sen|>
  pivot_longer(
      c('SLASSOsen','WCLASSOsen','WILASSOsen',
      'SFSsen','WCFSsen','WIFSsen'),
       names_to = "Indicators", values_to = "values")
p2 <- ggplot(mse_sen, aes(x = c, y = values, color = Indicators)) +
  geom_line() + # Using points to plot the data
  labs(title = "Plot of B.true1 and B.true2", x = "Index", y = "Beta Value")

p2
```




```
p_plotly2 <- ggplotly(p2)



mse_spe<-as.data.frame(cbind(c,SLASSOspe,WCLASSOspe,WILASSOspe,
      SFSspe,WCFSspe,WIFSspe))
```

```
mse_spe$c <- as.numeric(mse_spe$c)
mse_spe <- mse_spe|>
  pivot_longer(
       c('SLASSOspe','WCLASSOspe','WILASSOspe',
      'SFSspe','WCFSspe','WIFSspe'),
       names_to = "Indicators", values_to = "values")
p3 <- ggplot(mse_spe, aes(x = c, y = values, color = Indicators)) +
  geom_line() + # Using points to plot the data
  labs(title = "Plot of B.true1 and B.true2", x = "Index", y = "Beta Value")

p3
```




```
p_plotly3 <- ggplotly(p3)
```