Group Projects on Monte Carlo Simulation Design.

P8160 Advanced Statistical Computing

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

Background: When the number of candidate predictors in a linear model is large, variable selection is a common practice to find an optimal model that balances between model fitness and model complexity.

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.

Your tasks: In modern applications with high-dimensional covariates, traditional variable selection methods often struggle with the presence of "weak" predictors. Design a simulation study to investigate and illustrate (1) how well each of the two methods in identifying weak and strong predictors, and (2) how missing "weak" predictors impacts the parameter estimations.

To do so, you need to simulate data with a combination of `strong'',`weak-but-correlated'' and "weak-and-independent'' predictors. Their definition can be found in the following.

Definition of strong signals —

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

Definition of weak-but-correlated signals —

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

Definition of weak-and-independent signals —

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

Answer

Understand the system

1. How to generate independent arrays?

Generate the first array with random numbers. This array can be of any length and contain numbers from any distribution you choose, as long as it's a random selection.

Generate the second array in a way that ensures it has no linear relationship with the first array. One common method is to also generate this array randomly, independently of the first array, using the same or a different distribution.

Data Generation

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

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

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

```
# 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
```

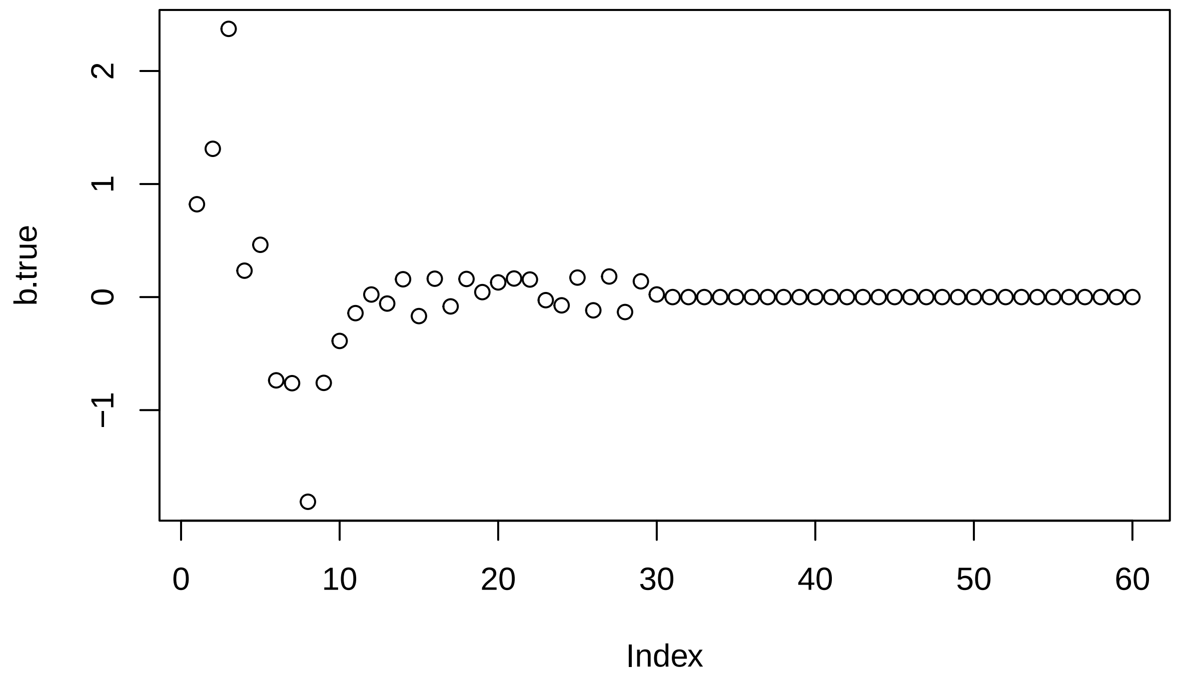
```r
b.true0 <- rep(0,p/2)
## combine them together
b.true       <- c(b.true1.strong,b.true1.weak1,b.true1.weak2,b.true0)
## name b.true
names(b.true) <- paste0("X", seq(1, 60))


# 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")
```

```
## 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)
```



Forward Selection

```
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)
```

```
##
## Call:
## lm(formula = y ~ X3 + X7 + X2 + X8 + X4 + X9 + X6 + X1 + X24 +
##     X16 + X38 + X58 + X12 + X50 + X13 + X47 + X32, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1702 -0.5383  0.0054  0.6455  3.5095
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.00434    0.11677   8.601 4.48e-13 ***
## X3           2.70688    0.35400   7.647 3.49e-11 ***
## X7          -0.96804    0.10902  -8.879 1.25e-13 ***
## X2           2.09181    0.36694   5.701 1.82e-07 ***
## X8          -1.45009    0.10381 -13.969  < 2e-16 ***
## X4           0.73251    0.12007   6.101 3.33e-08 ***
## X9          -0.59498    0.10708  -5.556 3.33e-07 ***
## X6          -1.12057    0.35812  -3.129  0.00243 **
## X1           0.39920    0.12290   3.248  0.00168 **
## X24         -0.28055    0.12207  -2.298  0.02409 *
## X16          0.28980    0.11297   2.565  0.01213 *
## X38         -0.25588    0.11775  -2.173  0.03267 *
## X58         -0.29981    0.12100  -2.478  0.01528 *
## X12         -0.17683    0.11577  -1.527  0.13052
## X50         -0.08856    0.04764  -1.859  0.06660 .
## X13         -0.17582    0.10846  -1.621  0.10884
## X47         -0.06140    0.03685  -1.666  0.09945 .
## X32         -0.15934    0.11807  -1.350  0.18088
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.056 on 82 degrees of freedom
## Multiple R-squared:  0.9363, Adjusted R-squared:  0.9231
## F-statistic:  70.9 on 17 and 82 DF,  p-value: < 2.2e-16
```

```
# Selected ones
```

LASSO

```
# 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]
```

```
##            V1            V2            V3            V4            V6
##  0.0927686804  1.2188941785  2.0242810294  0.5368293567 -0.2073960672
##            V7            V8            V9           V21           V24
## -0.9242284249 -1.1126633162 -0.2672232396  0.0237099029 -0.0520026667
##           V32           V38           V54
## -0.0316290260 -0.0007703875 -0.0002583006
```

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

Forward Selection

```
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 (FALS
  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

```
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
```

```
## [1] 0.812
```

```
Strong_FS_specificity = Strong_FS_specificity_sum/LOOP
Strong_FS_specificity
```

```
## [1] 0.808
```

```
Weakcor_FS_sensitivity  = Weakcor_FS_sensitivity_sum/LOOP
Weakcor_FS_sensitivity
```

```
## [1] 0.499
```

```
Weakcor_FS_specificity  = Weakcor_FS_specificity_sum/LOOP
Weakcor_FS_specificity
```

```
## [1] 0.793
```

```
Weakind_FS_sensitivity  = Weakind_FS_sensitivity_sum/LOOP
Weakind_FS_sensitivity
```

```
## [1] 0.432
```

```
Weakind_FS_specificity  = Weakind_FS_specificity_sum/LOOP
Weakind_FS_specificity
```

```
## [1] 0.733
```

```
mse_FS/LOOP
```

```
## [1] 0.5639514
```

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-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
```

```
## [1] 0.796
```

```r
Strong_LASSO_specificity = Strong_LASSO_specificity_sum/LOOP
Strong_LASSO_specificity
```

```
## [1] 0.902
```

```
Weakcor_LASSO_sensitivity  = Weakcor_LASSO_sensitivity_sum/LOOP
Weakcor_LASSO_sensitivity
```

```
## [1] 0.456
```

```
Weakcor_LASSO_specificity  = Weakcor_LASSO_specificity_sum/LOOP
Weakcor_LASSO_specificity
```

```
## [1] 0.888
```

```
Weakind_LASSO_sensitivity  = Weakind_LASSO_sensitivity_sum/LOOP
Weakind_LASSO_sensitivity
```

```
## [1] 0.31
```

```
Weakind_LASSO_specificity  = Weakind_LASSO_specificity_sum/LOOP
Weakind_LASSO_specificity
```

```
## [1] 0.812
```

```
mse_LASSO/LOOP
```

```
## [1] 1.114227
```

(2) how missing "weak" predictors impacts the parameter estimations.

1.Bias

2.Variance

3.Model Interpretability and Performance