234 final project

load packages

```r
# Load required packages
library(hdi)
library(stabs)
library(glmnet)
library(tidyverse)
library(MASS)
```

load data y is the log transformed production rate of riboflavin, essentially vitamin B x is the log transformed gene data

```r
data("riboflavin")
X <- riboflavin$x
y <- riboflavin$y
p <- ncol(X)
n <- nrow(X)

cat("Dimension of predictors: ", paste(dim(X), collapse = " x "), "
")
```

```
Dimension of predictors:  71 x 4088
```

```r
cat("Number of individual sample: ", length(y), "
")
```

```
Number of individual sample:  71
```

permute 4082 genes and keep 6 unpermuted

```r
marginal_corr <- apply(X, 2, function(col) abs(cor(col, y)))
top200 <- order(marginal_corr, decreasing = TRUE)[1:200]

set.seed(123)
true_genes_id <- sample(top200, 6)
true_genes <- marginal_corr[true_genes_id]

# Permute all other genes
n <- nrow(X)
noise_genes_id <- setdiff(1:ncol(X), true_genes_id)
noise_genes <- X[, -true_genes_id]

perm <- sample(1:n, n, replace = FALSE)
X_perm <- X
X_perm[, noise_genes_id] <- X[perm, noise_genes_id]

X_sim <- X_perm
```

Lasso Regularization Path

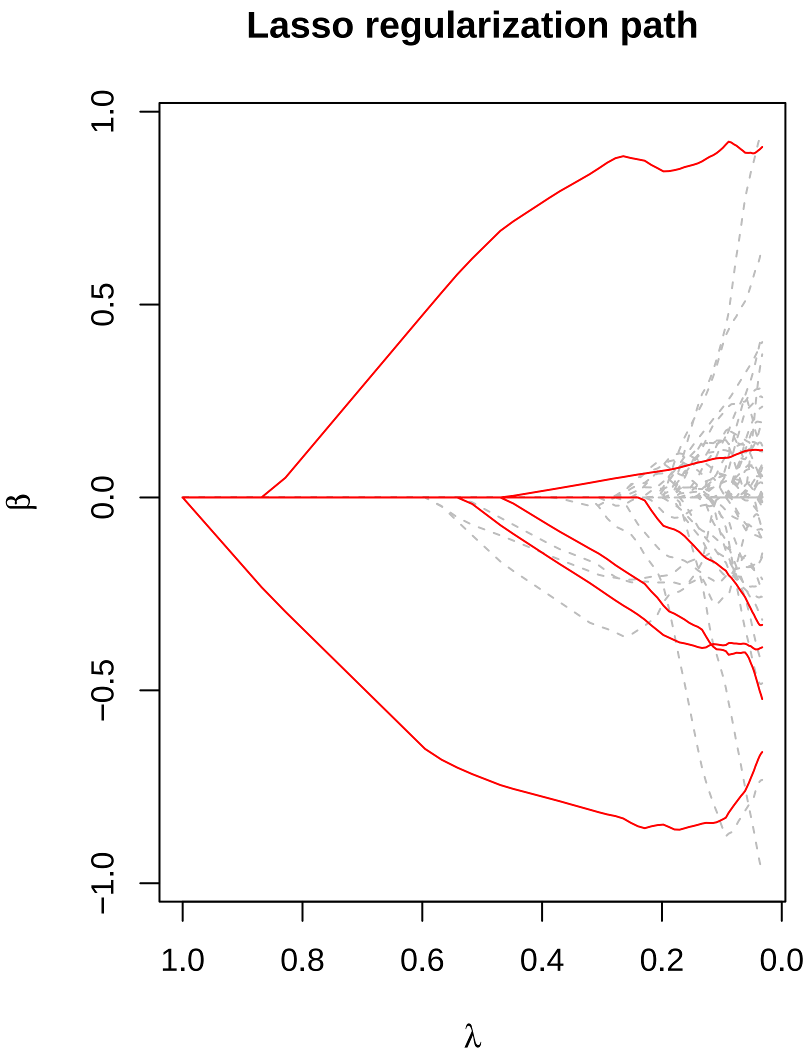
```r
fit_glm <- glmnet(X_sim, y, alpha = 1, standardize = TRUE)
```

```r
# error control to define regularization region of lambda
q <- sqrt(0.8*p)
coefs <- coef(fit_glm)[-1, ]  # remove intercept row
selected_counts <- apply(coefs, 2, function(col) sum(col != 0))
idx <- which(selected_counts >= q)[1]
lambda_region <- fit_glm$lambda[1:idx]  # from largest to chosen lambda

# cat("Variables selected at chosen lambda:", selected_counts[idx], "\n")
```

```r
# lambda rescale to [0, 1]
idx_null <- which(selected_counts == 0)[1]
lambda_null <- fit_glm$lambda[idx_null]      # null model lambda (max penalty)
lambda_min_full <- min(fit_glm$lambda)       # minimal lambda in full path
```

```
lambda_scaled <- (lambda_region - lambda_min_full)/(lambda_null - lambda_min_full)
lasso_lambda <- lambda_scaled

beta_mat <- as.matrix(fit_glm$beta)[, 1:idx] # p x l
beta_mat <- t(beta_mat) # transpose, l x p
combined_mat <- cbind(lasso_lambda, beta_mat)
```

```
plot(c(max(lasso_lambda), min(lasso_lambda)), range(beta_mat), type="n",
     xlab=expression(lambda), ylab=expression(beta),
     xlim = rev(range(lasso_lambda)),
     main = "Lasso regularization path")

for (i in noise_genes_id){
  lines(combined_mat[,1], beta_mat[,i], col=adjustcolor("grey"), lty =2)
}
for (i in true_genes_id){
  lines(combined_mat[,1], beta_mat[, i], col="red")
}
```

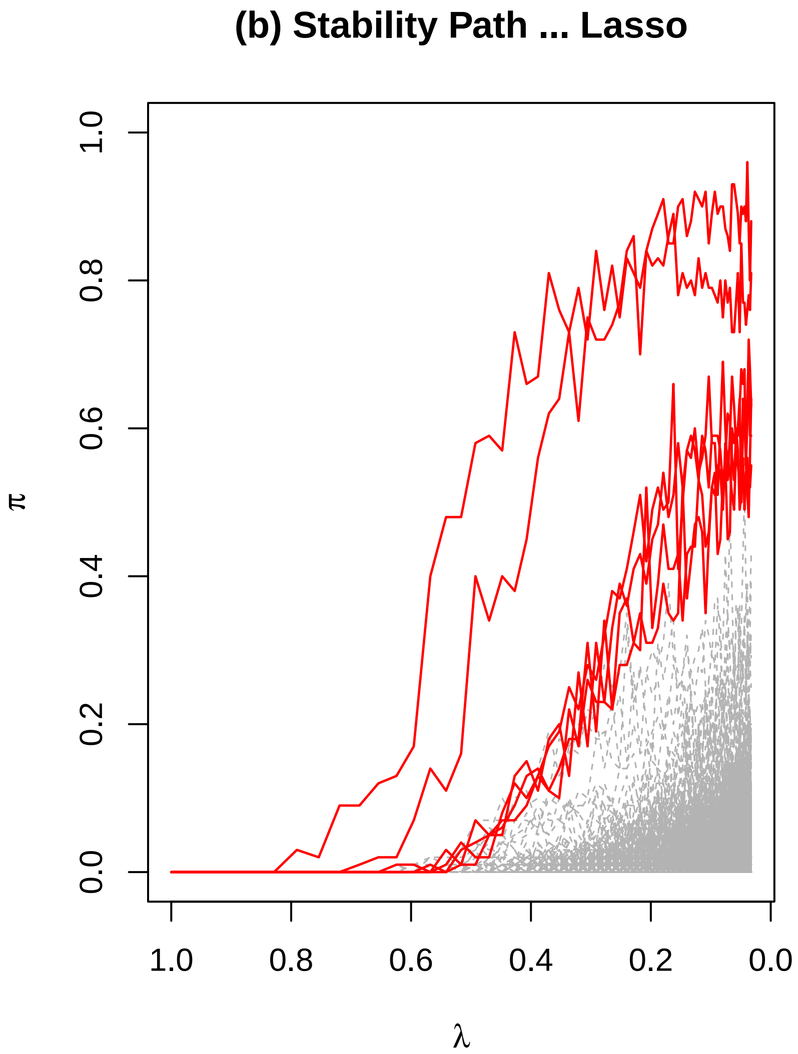


Stability Selection with LASSO

```
stability_lasso <- function(X, y, lambda_seq, B = 100) {
  # X: n x p
  n <- nrow(X)
  p <- ncol(X)
  stab <- matrix(0, nrow = p, ncol = length(lambda_seq))
  # p x l, each row is coef of each gene of a lambda
  for (j in seq_along(lambda_seq)) { #traverse through each lambda
    lam <- lambda_seq[j]
    for (b in 1:B) { # loop through all subsamples
      idx <- sample(1:n, n/2, replace = FALSE)
      fit <- glmnet(X[idx,], y[idx], alpha = 1, lambda = lam, standaridze= TRUE)
      stab[, j] <- stab[, j] + as.numeric(coef(fit)[-1] != 0)
      # update non-zero coef of all gene per lambda per sample
    }
  }
  return(stab / B)
}
```

? why the lambda generated from previous fit doesnt not result in convergence of percentage of unpermuted genes being selected out of samples to 0

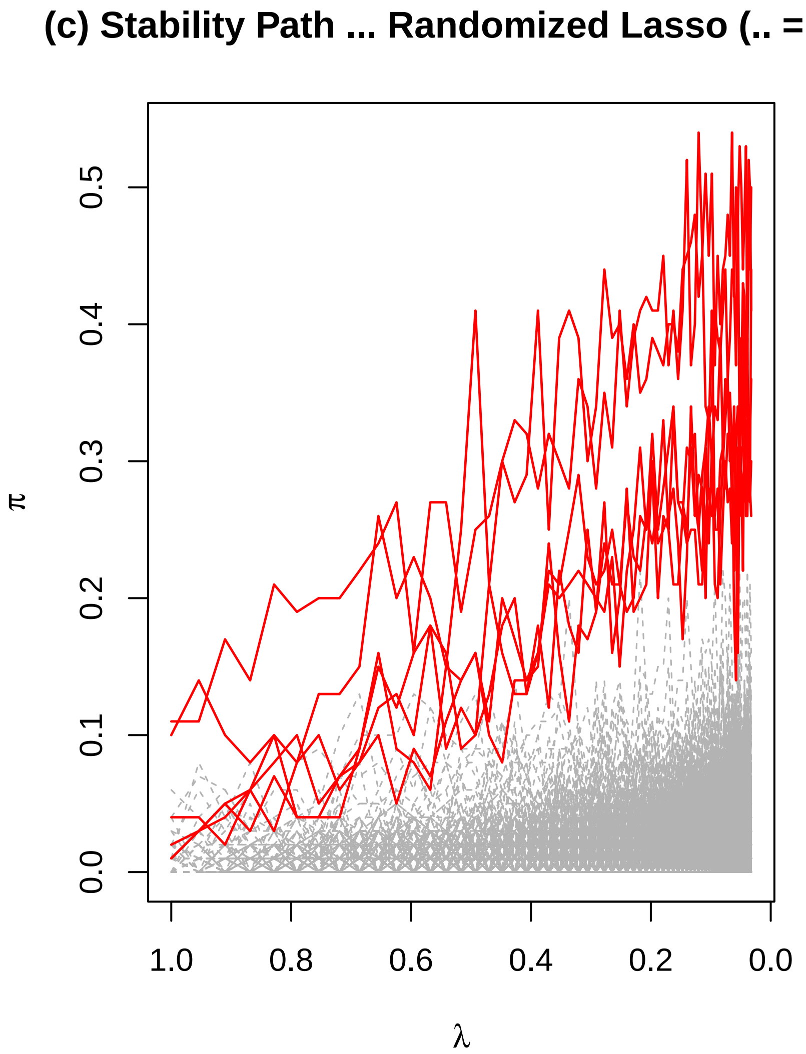
```
B_ <- 100
stab_lasso <- stability_lasso(X_sim, y, lasso_lambda, B = B_)

plot(range(lasso_lambda), c(0,1), type="n",
     xlab=expression(lambda), ylab=expression(pi),
     xlim = rev(range(lasso_lambda)),
     main="(b) Stability Path - Lasso")

# Noise genes
for (i in noise_genes_id) {
  lines(lasso_lambda, stab_lasso[i,], col="grey70", lty=2, lwd = 0.8)
}

# True genes
for (i in true_genes_id) {
  lines(lasso_lambda, stab_lasso[i,], col="red", lwd = 1.2)
}
```



randomized lasso

```
randomized_stability <- function(X, y, lambda_seq, B = 100, alpha = 0.2) {
  n <- nrow(X)
  p <- ncol(X)
  stab <- matrix(0, nrow = p, ncol = length(lambda_seq))
  for (j in seq_along(lambda_seq)) {
    lam <- lambda_seq[j]
    for (b in 1:B) {
      idx <- sample(1:n, n/2, replace = FALSE)

      # Random weights each predictors
      w <- runif(p, alpha, 1)
      # Xw <- sweep(X[idx,], 2, w, "/")

      fit <- glmnet(X[idx,], y[idx], alpha = 1, lambda = lam,
                    standardize = TRUE, penalty.factor = w)
      stab[, j] <- stab[, j] + as.numeric(coef(fit)[-1] != 0)
    }
  }
  return(stab / B)
}
```

```
stab_rand <- randomized_stability(X_sim, y, lasso_lambda, B = 100, alpha = 0.2)

plot(range(lasso_lambda), c(min(stab_rand), max(stab_rand)), type="n",
     xlab=expression(lambda), ylab=expression(pi),
     xlim = rev(range(lasso_lambda)),
     main="(c) Stability Path - Randomized Lasso (  = 0.2)")

# Noise genes
for (i in noise_genes_id) {
  lines(lasso_lambda, stab_rand[i,], col="grey70", lty=2, lwd = 0.8)
}

# True genes
for (i in true_genes_id) {
  lines(lasso_lambda, stab_rand[i,], col="red", lwd = 1.2)
}
```



(c) Stability Path ... Randomized Lasso (.. = 0.2

extension numerical results

in reality, it is nearly impossible to find perfect dataset that satisfy all the assumptions, such as iid samples that do not have any correlation and data set with perfect distribution; Thus, I want to examine the robustness of stability selection of lasso and randomized lasso on simulated data that is imperfect with correlated predictors. Correlated predictors can affect the regularization

Extension

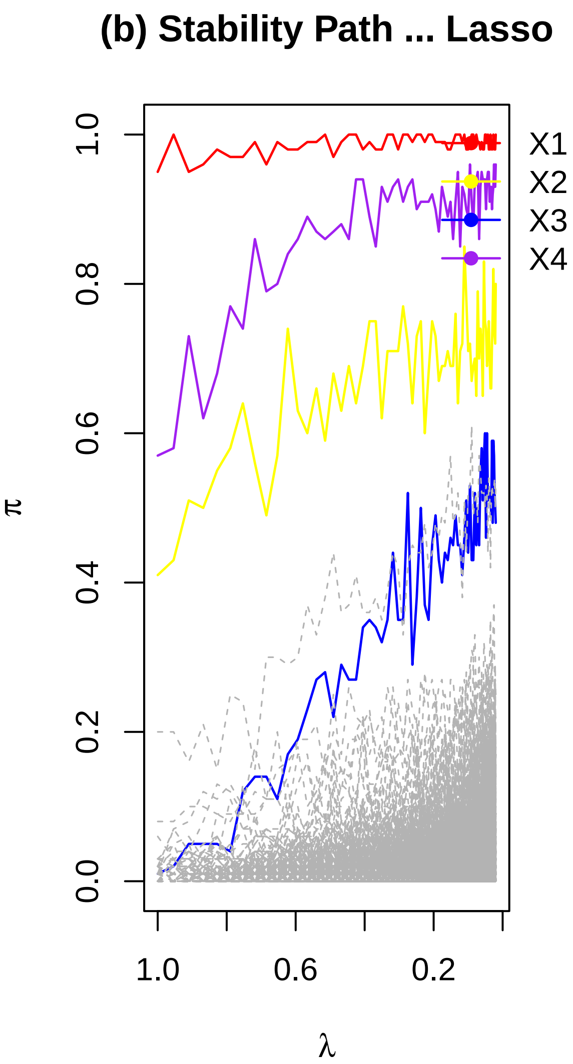
fit the stabilitys election on correlated dataset, lasso and randmoized lasso

```r
make_sigma <- function(p, rho, v1 = 1, v2 = 1){
  corr <- diag(p)
  corr[1,2] <- rho * sqrt(v1*v2)
  corr[2,1] <- corr[1,2]
  corr[1,1] <- v1
  corr[2,2] <- v2
  return(corr)
}
```

```r
simulate_data <- function(n, p, rho, var_x1, var_x2, signals){
    # epsilon, noise simulation, epsilon - N(0, 1/4), var = 1/2
    eps <- rnorm(n, 0, sd = 1/4)

    # beta simulation
    beta <- rep(0, p)
    for (i in 1:signals) { beta[i] = 1 }

    # predictor values simulation
    mns <- rep(0, p) # mean of each predictor
    var_x1 <- 1.2; var_x2 <- 0.5
    sigma <- make_sigma(p, rho = rho_) #, v1 = var_x1, v2 = var_x2 # covariance matrix of pre
    sim_X <- mvrnorm(n, mu = mns, Sigma = sigma) # n x p
    colnames(sim_X) <- paste0("X", 1:p)

    # response values simulation
    sim_y <- as.numeric(sim_X %*% beta + eps)

    # first fit
    fit_sim <- glmnet(sim_X, sim_y, alpha = 1, standardize = FALSE)

    # error control to define regularization region of lambda
    sim_q <- sqrt(0.8*p)
    sim_coefs <- coef(fit_sim)[-1, ]  # remove intercept row
    sim_selected_counts <- apply(sim_coefs, 2, function(col) sum(col != 0))
    sim_idx <- which(sim_selected_counts >= sim_q)[1]
    sim_lambda_region <- fit_sim$lambda[1:sim_idx]  # from largest to chosen lambda

    # lambda rescale to [0, 1]
    sim_idx_null <- which(sim_selected_counts == 0)[1]
    sim_lambda_null <- fit_sim$lambda[sim_idx_null]      # null model lambda (max penalty)
    sim_lambda_min_full <- min(fit_sim$lambda)       # minimal lambda in full path

    sim_lambda_scaled <- (sim_lambda_region - sim_lambda_min_full)/(sim_lambda_null - sim_lam
    sim_lambda_seq <- sim_lambda_scaled

    return(list(X = sim_X, y = sim_y, lambda_sq = sim_lambda_seq))

}
```

```
set.seed(123)
p_sim <- 1000 # num of predictors
n_sim <- 50 # num of samples
rho_ <- 0.8 # correlation value between x1 and x2, range from -1 to 1
signals <- 4 # number of signal varaible, aka relevant variable
var_x1 <- 1.2
var_x2 <- 0.5
results1 <- simulate_data(n_sim, p_sim, rho_, var_x1, var_x2, signals)
sim_X <- results1$X; sim_y <- results1$y
sim_lambda_seq <- results1$lambda_sq # regularization region
```

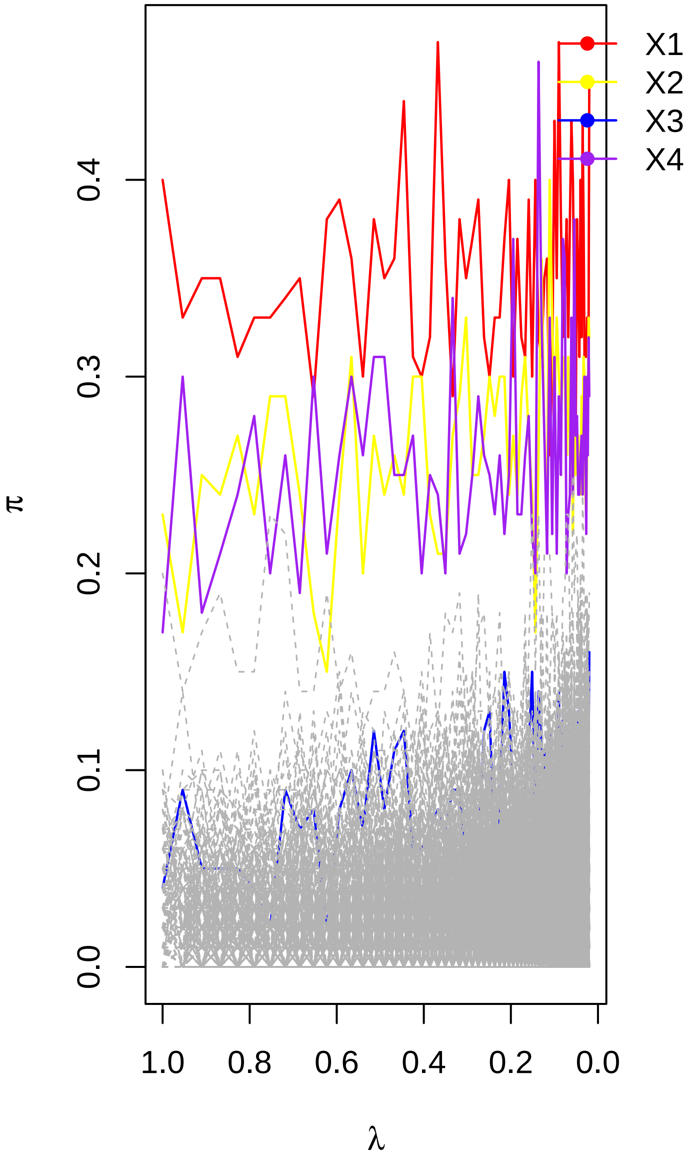
```
# stability selectin performance inspection
B_ <- 100
selection_threshold <- 0.9
sim_stab_lasso <- stability_lasso(sim_X, sim_y, sim_lambda_seq, B = B_)

par(mar = c(5, 5, 4, 8))
plot(c(max(sim_lambda_seq), min(sim_lambda_seq)), c(0,1), type="n",
     xlab=expression(lambda), ylab=expression(pi),
     xlim = rev(range(sim_lambda_seq)),
     main="(b) Stability Path - Lasso")

for (i in seq_len(nrow(sim_stab_lasso))) {
  if (i == 1){
    lines(sim_lambda_seq, sim_stab_lasso[i,], col="red", lwd = 1.2)
  }else if (i == 2){
    lines(sim_lambda_seq, sim_stab_lasso[i,], col="yellow", lwd = 1.2)
  }else if (i == 3){
    lines(sim_lambda_seq, sim_stab_lasso[i,], col="blue", lwd = 1.2)
  }else if(i == 4){
    lines(sim_lambda_seq, sim_stab_lasso[i,], col="purple", lwd = 1.2)
  }else{
    lines(sim_lambda_seq, sim_stab_lasso[i,], col="grey70", lty=2,lwd = 0.8)
  }
}

par(xpd = TRUE)
legend("topright",legend = c("X1", "X2", "X3", "X4"),col = c("red", "yellow", 'blue', 'purpl
        inset = c(-0.2, 0), lwd = 1.2, pch = 16, bty = "n")
```



```
par(xpd = FALSE)
```

```
# randomized lasso stability selection
B_ <- 100
sim_stab_rand <- randomized_stability(sim_X, sim_y, sim_lambda_seq, B = B_, alpha = 0.2)

par(mar = c(5, 5, 4, 8))
plot(range(sim_lambda_seq), c(min(sim_stab_rand), max(sim_stab_rand)), type="n",
     xlab=expression(lambda), ylab=expression(pi),
     xlim = rev(range(sim_lambda_seq)),
     main="(c) Stability Path - Randomized Lasso (  = 0.2)")

for (i in seq_len(nrow(sim_stab_lasso))) {
  if (i == 1){
    lines(sim_lambda_seq, sim_stab_rand[i,], col="red", lwd = 1.2)
  }else if (i == 2){
    lines(sim_lambda_seq, sim_stab_rand[i,], col="yellow", lwd = 1.2)
  }else if (i == 3){
    lines(sim_lambda_seq, sim_stab_rand[i,], col="blue", lwd = 1.2)
  }else if(i == 4){
    lines(sim_lambda_seq, sim_stab_rand[i,], col="purple", lwd = 1.2)
  }else{
    lines(sim_lambda_seq, sim_stab_rand[i,], col="grey70", lty=2, lwd = 0.8)
  }
}
par(xpd = TRUE)
legend("topright",legend = c("X1", "X2", "X3", "X4"),col = c("red", "yellow", 'blue', 'purpl
       inset = c(-0.2, 0), lwd = 1.2, pch = 16, bty = "n")
```

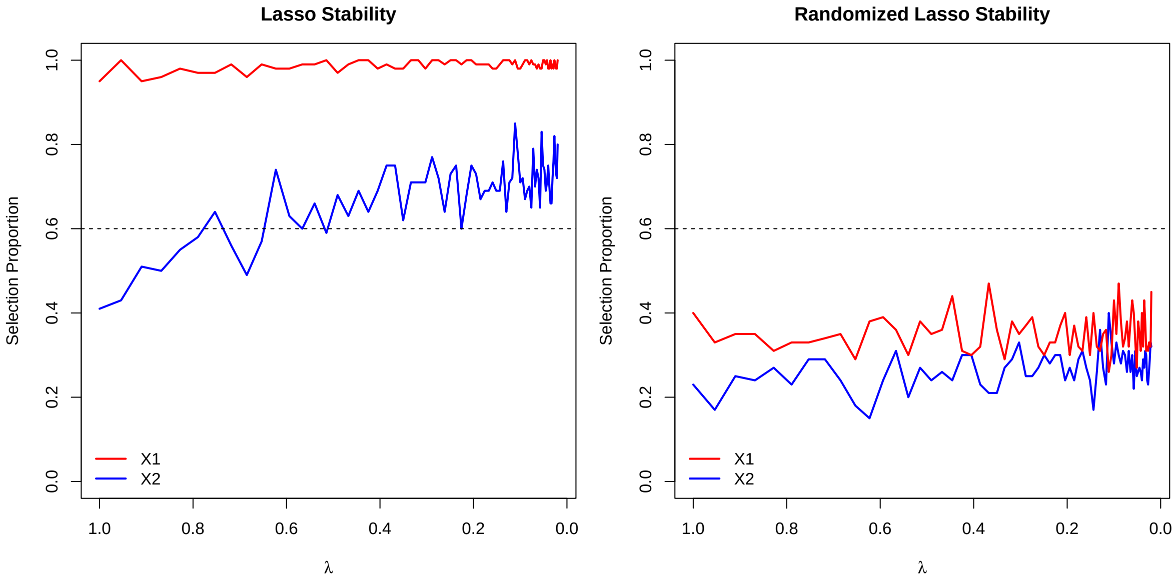
(c) Stability Path ... Randomized Lasso (.. = 0.2)



```r
par(xpd = FALSE)
```

proportion x1 selected vs x2 selected: lasso vs rand lasso

```r
par(mfrow = c(1, 2),mar = c(4, 4, 3, 1))
# proportion in standard lasso
x1_prop_lasso <- sim_stab_lasso[1,]; x2_prop_lasso <- sim_stab_lasso[2,]

# visualization
plot(sim_lambda_seq, x2_prop_lasso, type = "l",ylab = "Selection Proportion",
     ylim = c(0, 1), lwd = 2, col = "blue", xlab = expression(lambda),
     xlim = rev(range(sim_lambda_seq)),
     main = "Lasso Stability")

lines(sim_lambda_seq, x1_prop_lasso,lwd = 2, col = "red")
abline(h = 0.6, lty = 2)
legend("bottomleft", legend = c("X1", "X2"),col = c("red", "blue"),
       lwd = 2, bty = "n")

# proportion in randomized lasso
x1_prop_rand_lasso <- sim_stab_rand[1,]; x2_prop_rand_lasso <- sim_stab_rand[2,]

# visualization
plot(sim_lambda_seq, x2_prop_rand_lasso, type = "l",ylab = "Selection Proportion",
     ylim = c(0, 1), lwd = 2, col = "blue", xlab =expression(lambda),
     xlim = rev(range(sim_lambda_seq)),
     main = "Randomized Lasso Stability")

# Add X1 (flat line at 1)
lines(sim_lambda_seq, x1_prop_rand_lasso,lwd = 2, col = "red")
# Add stability threshold line
abline(h = 0.6, lty = 2)
legend("bottomleft", legend = c("X1", "X2"), col = c("red", "blue"),
       lwd = 2, bty = "n")
```



comparison of different rho (rand lasso)

```r
rho_vals <- c(0.2, 0.6, 1)
simulation_results <- list()

for (r in rho_vals) {
  cat("Running simulation for rho =", r, "\n")

  results <- simulate_data(n = n_sim, p = p_sim, rho = r, var_x1 = var_x1,
                           var_x2 = var_x2, signals = signals)

  X_sim <- results$X
  y_sim <- results$y
  lambda_seq0 <- results$lambda_sq

  stab_lasso <- stability_lasso(X_sim, y_sim, lambda_seq0)
  stab_random <- randomized_stability(X_sim, y_sim, lambda_seq0, alpha = 0.2)

  simulation_results[[paste0("rho_", r)]] <- list(
    X = X_sim,
    y = y_sim,
    lambda_seq1 = lambda_seq0,
    stab_lasso = stab_lasso,
    stab_random = stab_random
  )
}
```

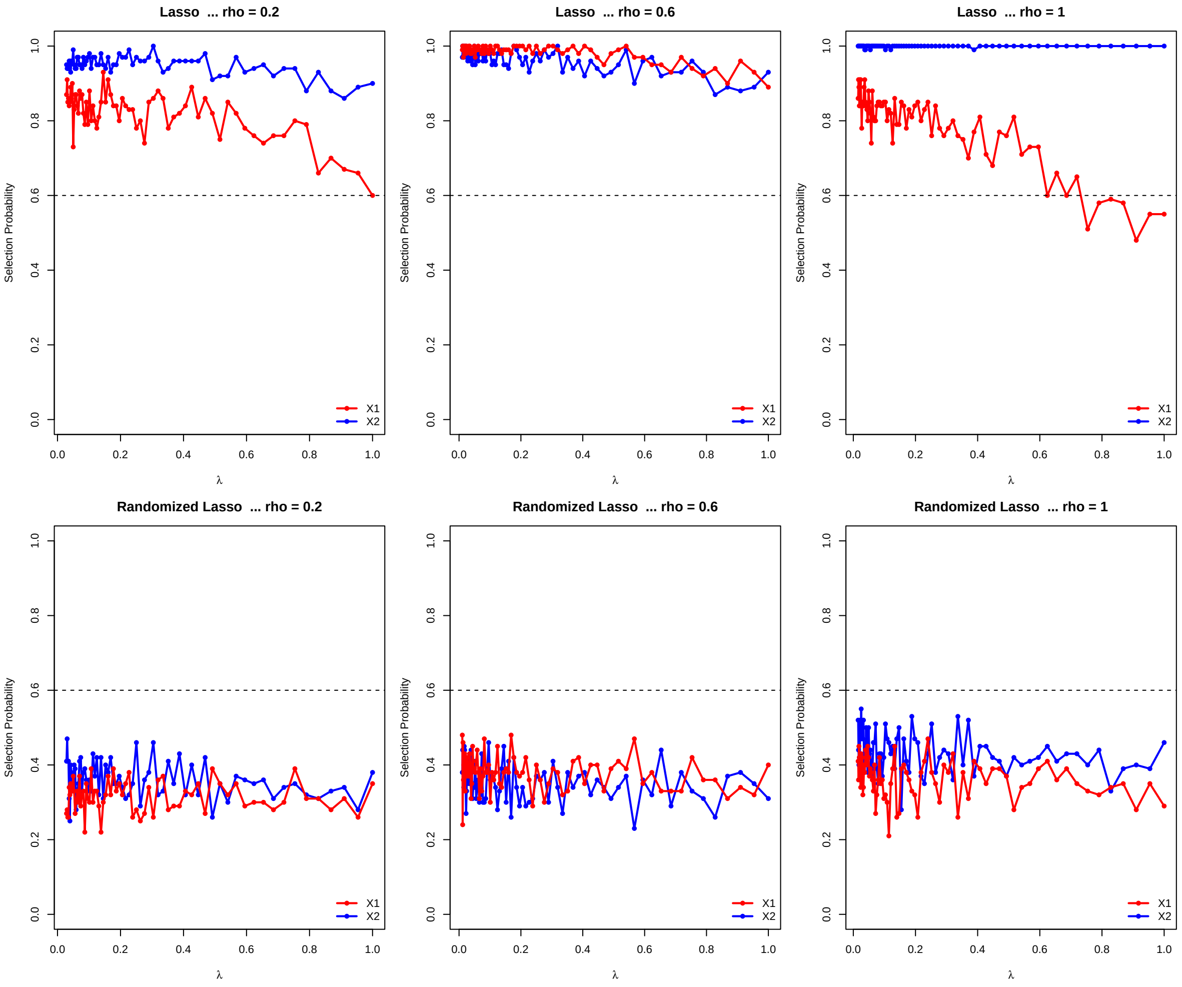
```
Running simulation for rho = 0.2
Running simulation for rho = 0.6
Running simulation for rho = 1
```

```
res02 <- simulation_results$rho_0.2
stab_lasso_02 <- res02$stab_lasso
stab_random_02 <- res02$stab_random

res06 <- simulation_results$rho_0.6
stab_lasso_06 <- res06$stab_lasso
stab_random_06 <- res06$stab_random

res1 <- simulation_results$rho_1
stab_lasso_1 <- res1$stab_lasso
stab_random_1 <- res1$stab_random
```

```
plot_stability_two_vars <- function(lambdas, stab, rho_value, method_name) {
  x1 <- stab[1, ]
  x2 <- stab[2, ]

  plot(lambdas, x2, type = "o", pch = 16, ylim = c(0,1),
       xlab = expression(lambda), ylab = "Selection Probability",
       main = paste(method_name, " - rho =", rho_value), col = "blue", lwd = 2)

  lines(lambdas, x1, type = "o", pch = 16, col = "red", lwd = 2)
  abline(h = 0.6, lty = 2)
  legend("bottomright",legend = c("X1", "X2"), col = c("red", "blue"),
         lwd = 2, pch = 16, bty = "n")
}
```

```
par(mfrow = c(2, 3), mar = c(4,4,3,1))

# lasso
plot_stability_two_vars(lambdas = res02$lambda_seq1, stab = stab_lasso_02,
                        rho_value = 0.2, method_name = "Lasso")
plot_stability_two_vars(lambdas = res06$lambda_seq1, stab = stab_lasso_06,
                        rho_value = 0.6,method_name = "Lasso")
```

```
plot_stability_two_vars(lambdas = res1$lambda_seq1, stab = stab_lasso_1,
                        rho_value = 1, method_name = "Lasso")
# randomized lasso
plot_stability_two_vars(lambdas = res02$lambda_seq1, stab = stab_random_02,
                        rho_value = 0.2, method_name = "Randomized Lasso")
plot_stability_two_vars(lambdas = res06$lambda_seq1,stab = stab_random_06,
                        rho_value = 0.6, method_name = "Randomized Lasso")
plot_stability_two_vars(lambdas = res1$lambda_seq1, stab = stab_random_1,
                        rho_value = 1,method_name = "Randomized Lasso")
```

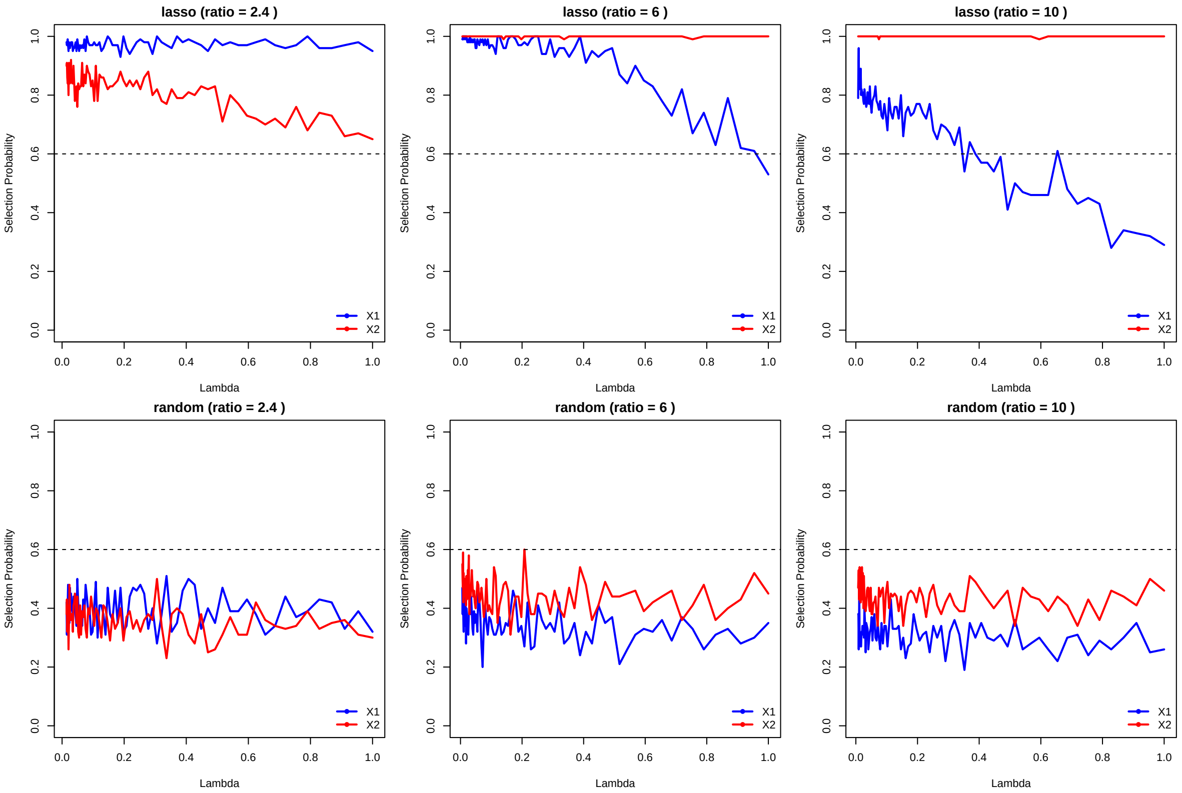


comparison of magnitude of variances

```
#specify different variances to x1 and x2; focus on the ratio x1/x2 only
var_set <- list(c(1.2, 0.5), c(3.0, 0.5), c(5.0, 0.5))
results_var <- list()
rho1 <- 0.8
i <- 1
for (vars in var_set) {
  vx1 <- vars[1]; vx2 <- vars[2]
  res <- simulate_data(n = n_sim, p = p_sim, rho = rho1, var_x1 = vx1,
                       var_x2 = vx2, signals = signals)
  X_sim <- res$X
  y_sim <- res$y
  lambda_seq2 <- res$lambda_sq
  stab_lasso <- stability_lasso(X_sim, y_sim, lambda_seq2)
  stab_random <- randomized_stability(X_sim, y_sim, lambda_seq2, alpha = 0.2)

  results_var[[i]] <- list(lambda_ = lambda_seq2, var_ratio = vx1/vx2,
                           stab_lasso = stab_lasso, stab_random = stab_random)
  i <- i + 1
}
```

```
# Build a list for easy plotting: results_plot[[ratio]][[method]]
results_plot <- list()

for (i in seq_along(results_var)) {

  ratio <- results_var[[i]]$var_ratio
  lambda_seq_ <- results_var[[i]]$lambda_

  # extract only X1 and X2 rows
  prob_lasso <- results_var[[i]]$stab_lasso[1:2, , drop = FALSE]
  prob_rand  <- results_var[[i]]$stab_random[1:2, , drop = FALSE]

  results_plot[[as.character(ratio)]] <-
    list(lasso = list(lambda__ = lambda_seq_, X1 = prob_lasso[1, ],
                      X2 = prob_lasso[2, ]),
         random = list(lambda__ = lambda_seq_, X1 = prob_rand[1, ],
                       X2 = prob_rand[2, ]))
}
```

```
par(mfrow = c(2, 3), mar=c(4,4,2,1))

methods <- c("lasso", "random")
ratios  <- names(results_plot)

for (method in methods) {
  for (ratio in ratios) {

    obj <- results_plot[[ratio]][[method]]
    lambda3 <- obj$lambda__

    # Plot X1 line
    plot(lambda3, obj$X1, type = 'l', col = "blue", lwd = 2, ylim = c(0, 1),
         xlab = "Lambda", ylab = "Selection Probability",
         main = paste(method, "(ratio =", ratio, ")"))

    # x2
    lines(lambda3, obj$X2,col = "red",lwd = 2)

    abline(h = 0.6, lty = 2)
    # legend
    legend("bottomright",legend = c("X1", "X2"),col = c("blue", "red"),
           lwd = 2, pch = 16, bty = "n")
  }
}
```


lasso (ratio = 2.4)
lasso (ratio = 6)
lasso (ratio = 10)
random (ratio = 2.4)
random (ratio = 6)
random (ratio = 10)
Selection Probability
Lambda
0.0
0.2
0.4
0.6
0.8
1.0
X1
X2