Computational Statistics : Problem Set 5

Gewei Cao, Andreas Koundouros, Raúl Luyando, Erik Covarrubias

24.05.2022

Exercise 1

The data generating process is as follows:

$$Y = X \cdot \beta + \varepsilon$$

We consider $n = 100$, and $p = 3$ covariates, $X \sim \mathcal{N}_p(0, \Sigma)$ and, $\varepsilon \sim \mathcal{N}(0, 10)$.

In order to preform our study, we program a data generating functions with an arbitrary Σ. This function stores both the standardized and un-standardized covariates, as the latter ones are needed to run the ridge regression.

The standarization was done as follows:

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_j}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x_j})^2}}$$

```
set.seed(777)

library(MASS)

# Parameters

n <- 100
beta.true <- c(0.5, 0.5, -0.5)
sigma <- matrix(c(2,0.1,0.1,0.1,3,0.1,0.1,0.1,4) ,nrow= 3, ncol= 3, byrow=TRUE)
mu <- rep(0,3)

# Data generator

data.generator <- function(n, sigma, mu, beta){
  x <- mvrnorm(n, mu, sigma)
  e <- rnorm(n, 0, sqrt(10))
  y <- x %*% beta + e
  x.1 <- (x[,1] - mean(x[,1])) / sd(x[,1])
  x.2 <- (x[,2] - mean(x[,2]))/ sd(x[,2])
  x.3 <- (x[,3] - mean(x[,3]))/ sd(x[,3])
  x.sd <- cbind(x.1, x.2, x.3)
  data <- data.frame ("y" = y, "x.sd" = x.sd, "x" = x)
  return(data)
}

# Training data for the Exercises
```

```
data <- data.generator(n, sigma, mu, beta.true)
head(data)
```

```
##            y   x.sd.x.1   x.sd.x.2   x.sd.x.3        x.1        x.2        x.3
## 1 -0.2747329 -0.2610935 -0.8432409 -0.4351033 -0.2488734 -1.5676152 -0.8118595
## 2  3.6247862  1.2148685 -0.2126859  0.3793337  2.1922599 -0.4131343  0.7269980
## 3 -3.9744011 -0.2912272 -0.4786426 -0.4928201 -0.2987124 -0.9000735 -0.9209139
## 4  4.7485313 -0.2145335  1.1195675  0.3140972 -0.1718666  2.0260839  0.6037353
## 5 -3.7068684 -0.9604029 -1.3201498 -1.5804828 -1.4054801 -2.4407862 -2.9760241
## 6 -5.2451449 -1.1601933 -0.9927257 -0.5173811 -1.7359188 -1.8413065 -0.9673212
```

```
# Storing the centered and uncentered covariates in a matrix for later convenience

x.sd <- cbind(data$x.sd.x.1, data$x.sd.x.2, data$x.sd.x.3)
x <- cbind(data$x.1, data$x.2, data$x.3)
```

a)

The ridge coefficients minimize a penalized residual sum of squares:

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} x_{ij}\beta_j^2 \right\}$$

$$\Leftrightarrow \hat{\beta}^{\text{ridge}} = \left(X^T X + \lambda I \right)^{-1} X^T y$$

Using the latter expression we can program a function that computes $\hat{\beta}$ given covariates X, the explanatory variable y and a given λ.

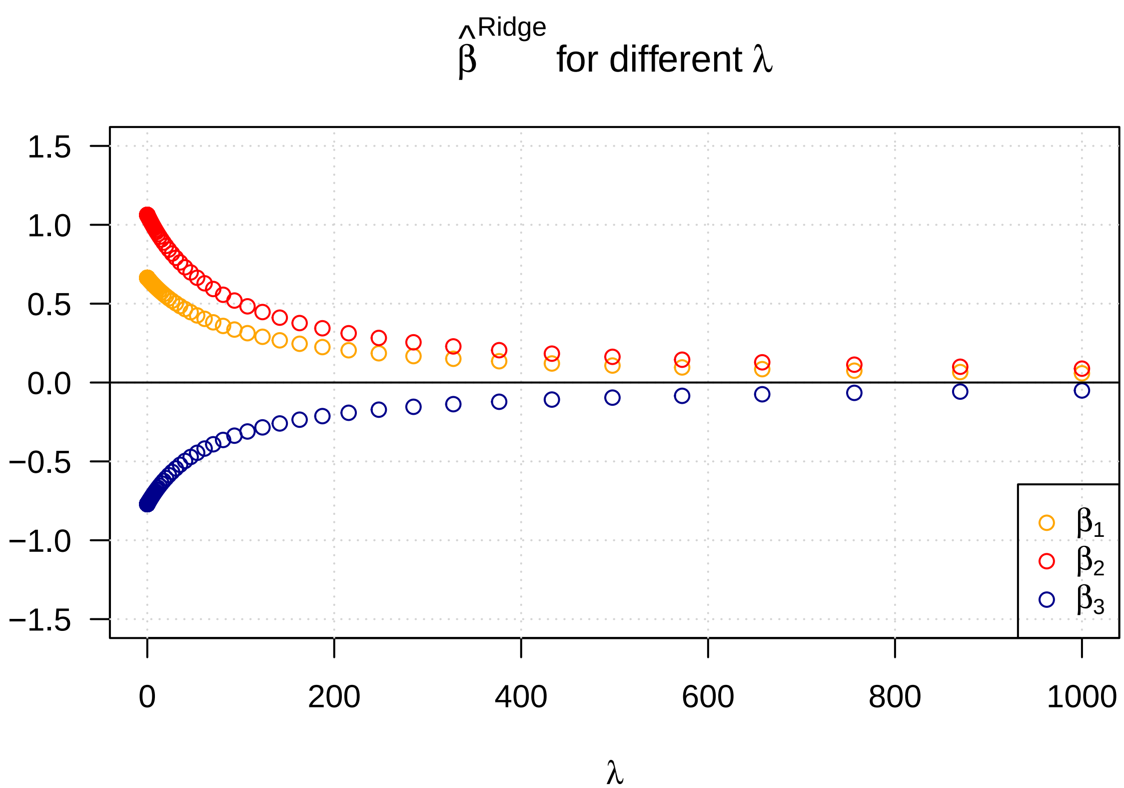
```
# Ridge-regression function

ridge <- function(x,y,lambda, p = 3){
  beta.ridge <- solve(t(x) %*% x + diag(lambda, nrow = p, ncol = p)) %*% t(x) %*% y
  return(beta.ridge)
}
```

b)

Estimating $\hat{\beta}$ for $\lambda \in [10^{-3}, 10^{3}]$. As λ goes to zero, the estimates converge to the ones predicted by OLS. On the other hand, larger penalization terms *shrink* the estimated coefficients to zero.

```
# Grid for the lambdas

grid <- 10^ seq (3, -3, length = 100)

# Storage matrix for the regression outputs

beta.ridge <- matrix(NA, nrow = 3, ncol= length(grid))

# Ridge regression over the grid

for (i in 1:length(grid)){
```

```r
  lam <- grid[i]
  beta.ridge[,i] <- ridge(x.sd, data$y, lam)
}

# Plot

par(las = 1)
ylimits = c(-1.5, 1.5)
plot(x=grid, y=beta.ridge[1,], col = "red", ylim = ylimits,
     xlab = expression(lambda), ylab = "",
     main = expression(hat(beta)^Ridge ~ "for different" ~ lambda), type = "n")
grid()
points(x=grid, y=beta.ridge[1,], col = "orange", lwd = 1)
points(x=grid, y=beta.ridge[2,], col = "red", lwd = 1)
points(x=grid, y=beta.ridge[3,], col = "darkblue", lwd = 1)
abline(h = 0, col = "black")
legend("bottomright", c(expression(beta[1]), expression(beta[2]), expression(beta[3])),
       col = c("orange", "red", "darkblue"), pch = 1)
```



c)

Comparing the performance of the Ridge regression and OLS in a one-run simulation by calculating each estimators **train- and test MSE** as follows:

The average training error error:

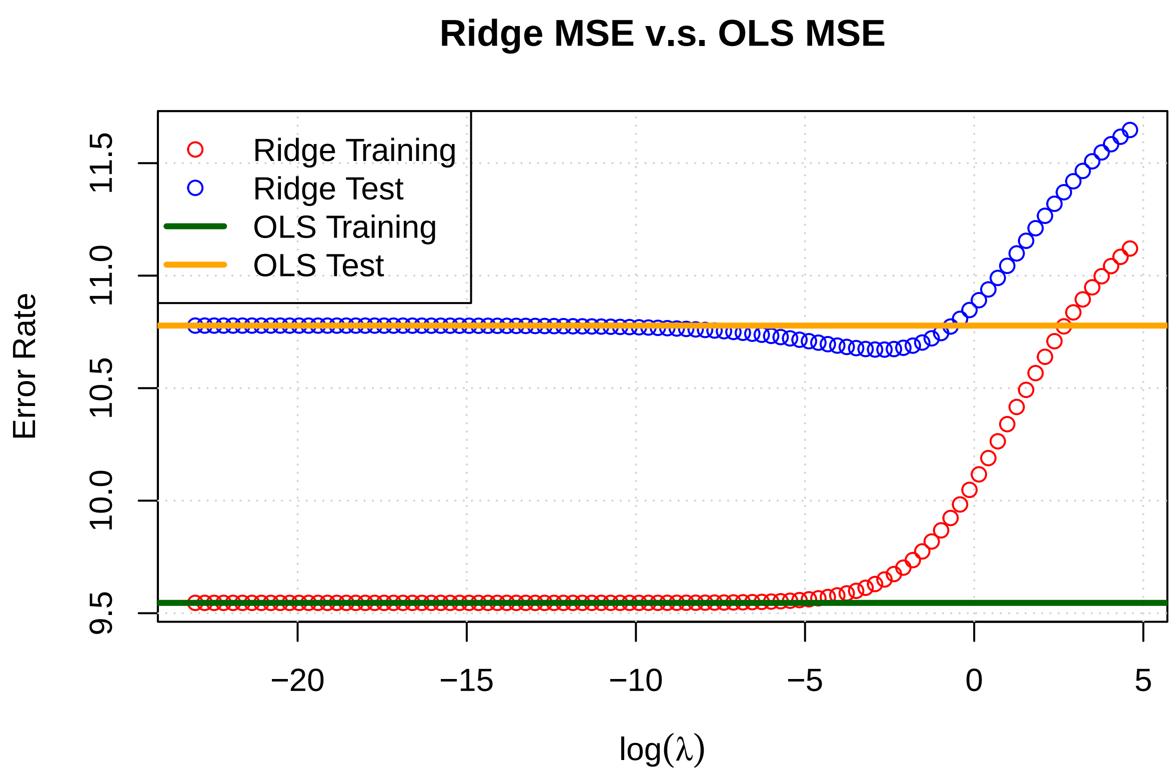
$$\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

The average prediction error:

$$\frac{1}{n}\sum_{i=1}^{n}(y_0-\hat{y}_0)^2$$

```
set.seed(123)

# Creating a new grid for the values of lambda

grid <- 10^ seq (2, -10, length = 100)

# Generating test data

test.data <- data.generator(n, sigma, mu, beta.true)

# Uncentered test data
x.t <- cbind(test.data$x.1, test.data$x.2, test.data$x.3)

# Centered test data

x.sd.t <- cbind(test.data$x.sd.x.1, test.data$x.sd.x.2, test.data$x.sd.x.3)

# Containers for the train- and test errors of the ridge regression

train.error <- c()
test.error <- c()

for (i in 1:length(grid)){
  y.hat <- x.sd %*% beta.ridge[,i]
  train.error[i] <- mean((data$y - y.hat)^2)
  y.hat.t <- x.sd.t %*% beta.ridge[,i]
  test.error[i] <- mean((test.data$y - y.hat.t)^2)
}

# Train - and test errors for the OLS function

lm.obj <- lm(data$y ~ x.sd.x.1 + x.sd.x.2 + x.sd.x.3 -1, data = data)
lm.obj.test <- x.sd.t %*% lm.obj$coefficients


ols.train.error <- mean((lm.obj$fitted.values - data$y)^2)
ols.test.error <- mean((lm.obj.test - test.data$y)^2)




# Plot

plot(x = log(grid), y = train.error, col = "red", main = "Ridge MSE v.s. OLS MSE", ylim = c(min(train.e
     type = "n",
```

```
     xlab = expression(log(lambda)), ylab = "Error Rate")
grid()
points(x = log(grid),  y = test.error, col = "blue")
points(x = log(grid),  y = train.error, col = "red")
abline(h = ols.train.error, col = "darkgreen", lwd = 3)
abline(h = ols.test.error, col = "orange", lwd = 3)
legend("topleft", legend = c("Ridge Training", "Ridge Test", "OLS Training",
                             "OLS Test"), pch = c(1,1, NA, NA),
       lty = c(NA, NA, 1, 1), lwd = c(NA, NA, 3, 3),
       col = c("red", "blue", "darkgreen", "orange"),
       cex = 1)
```

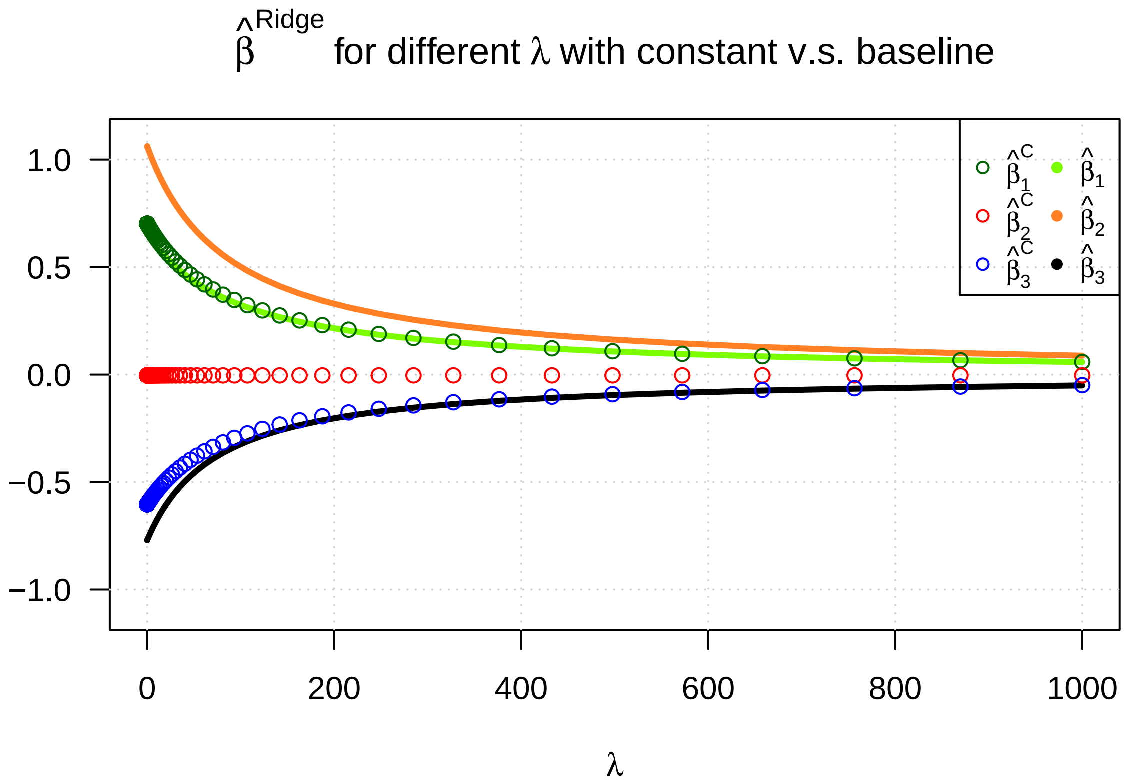


d)

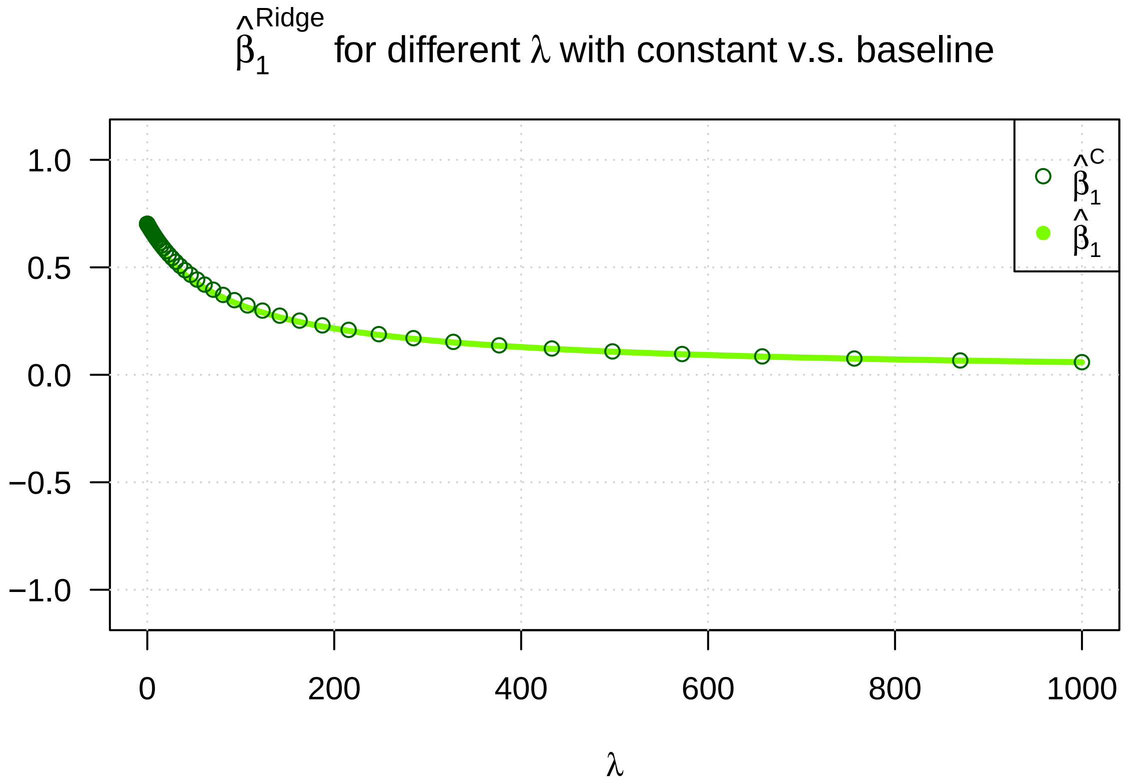
Replacing X_2 with a constant with a value of -10.

```
x.sd2 <- x.sd
x.sd2[,2] <- rep(-10, n)

grid <- 10^ seq (3, -3, length = 100)

# Storage matrix for the regression outputs

beta.ridge.d <- matrix(NA, nrow = 3, ncol= length(grid))

# Ridge regression over the grid
```

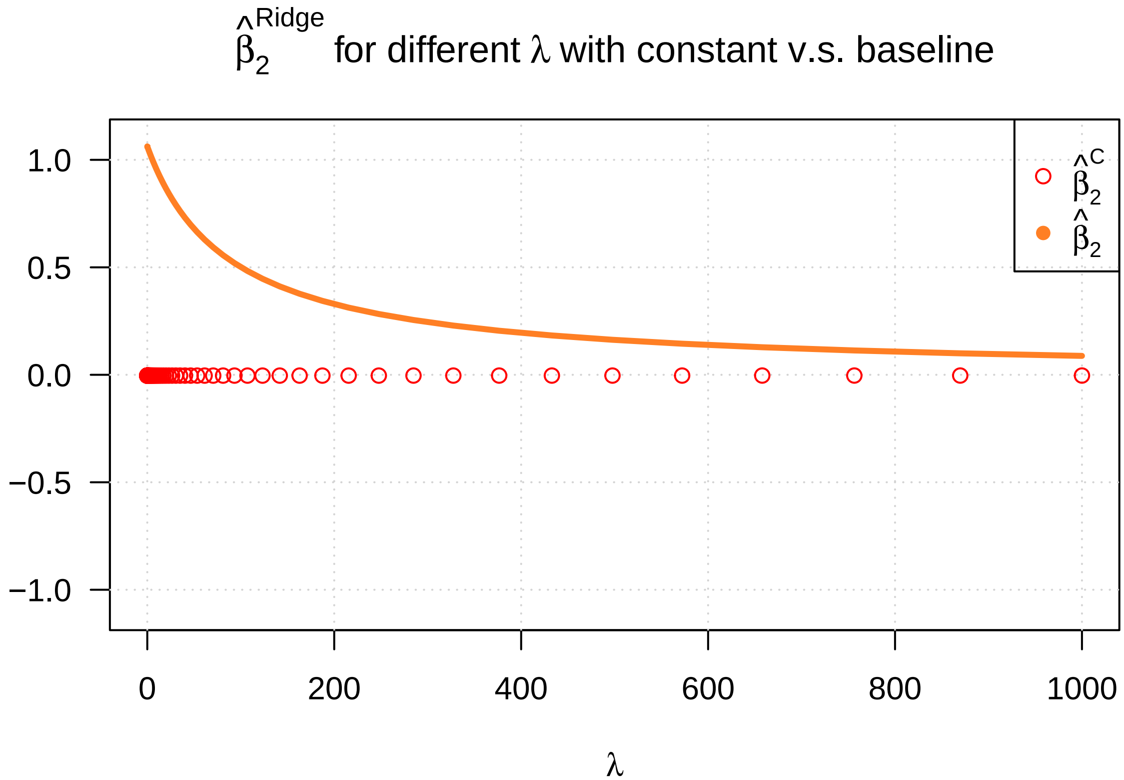
```r
for (i in 1:length(grid)){
  lam <- grid[i]
  beta.ridge.d[,i] <- ridge(x.sd2, data$y, lam)
}

# OLS regression


lm.obj <- lm(data$y ~ x.sd2 -1)


# Code for plot in 1d)

# Common plot of all coefficient estimates
par(las = 1)
ylimits = c(-1.1, 1.1)
plot(x=grid, y=beta.ridge.d[1,], col = "red", ylim = ylimits,
     xlab = expression(lambda), ylab = "",
     main = expression(hat(beta)^Ridge ~ "for different" ~ lambda ~ "with constant v.s. baseline"),
     type = "n")
grid()
lines(x=grid, y=beta.ridge[1,], col = "lawngreen", lwd = 3)
lines(x=grid, y=beta.ridge[2,], col = "chocolate1", lwd = 3)
lines(x=grid, y=beta.ridge[3,], col = "black", lwd = 3)
points(x=grid, y=beta.ridge.d[1,], col = "darkgreen", pch = 1)
points(x=grid, y=beta.ridge.d[2,], col = "red", pch = 1)
points(x=grid, y=beta.ridge.d[3,], col = "blue", pch = 1)
lgd.ordering <- matrix(c(1:6), ncol = 2, nrow = 3, byrow = F)
legend("topright",
       c(expression(hat(beta)[1]^C), expression(hat(beta)[2]^C), expression(hat(beta)[3]^C),
         expression(hat(beta)[1]), expression(hat(beta)[2]), expression(hat(beta)[3]))[lgd.ordering],
       pch = c(1, 1, 1, 16, 16, 16)[lgd.ordering],
       ncol = 2,
       col = c("darkgreen", "red", "blue", "lawngreen", "chocolate1", "black")[lgd.ordering],
       cex = 0.8)
```



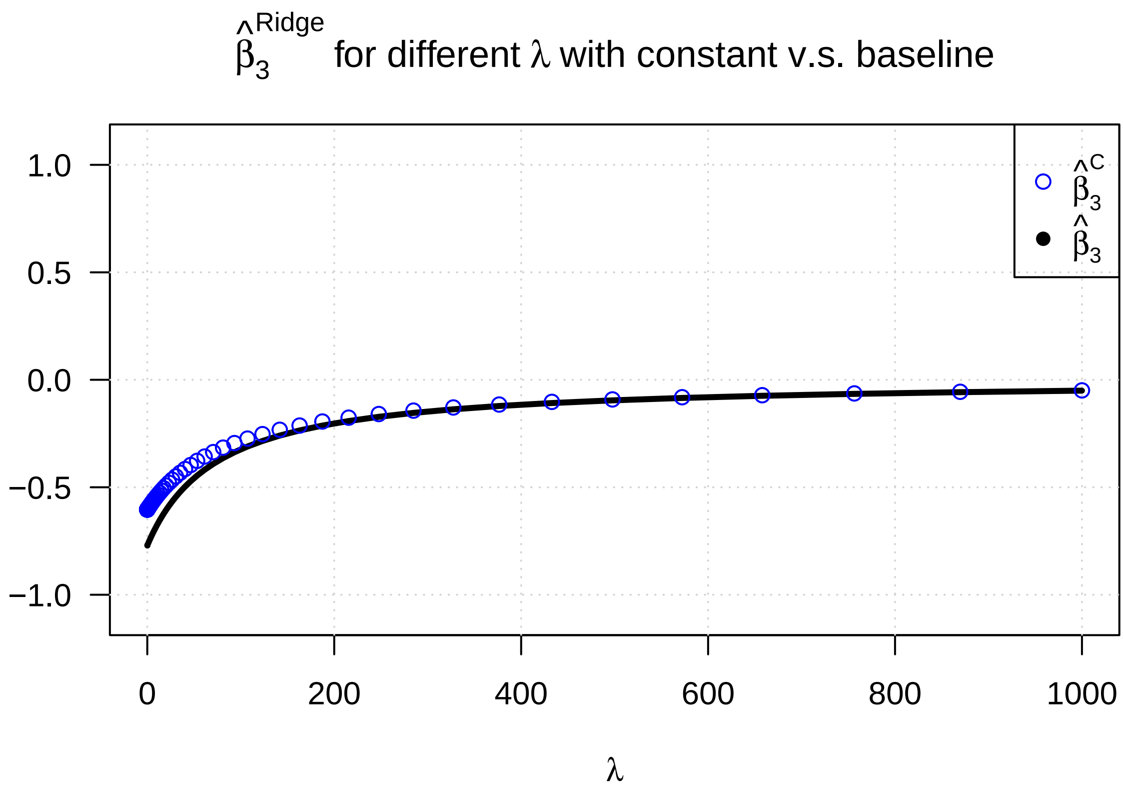

```r
# Plots of individual coefficient estimates
plot(x=grid, y= beta.ridge.d[1,], col = "darkgreen", ylim = ylimits, type = "n",
     main = expression(hat(beta)[1]^Ridge ~ "for different" ~ lambda ~ "with constant v.s. baseline"),
     xlab = expression(lambda),
     ylab = "")
grid()
lines(x=grid, y= beta.ridge[1, ], col = "lawngreen", lwd = 3)
points(x=grid, y= beta.ridge.d[1, ], col = "darkgreen")
legend("topright", legend = c(expression(hat(beta)[1]^C), expression(hat(beta)[1])),
       col = c("darkgreen", "lawngreen"), pch = c(1, 16))
```




```
plot(x=grid, y= beta.ridge.d[2,], col = "red", ylim = ylimits, type = "n",
     main = expression(hat(beta)[2]^Ridge ~ "for different" ~ lambda ~ "with constant v.s. baseline"),
     xlab = expression(lambda),
     ylab = "")
grid()
lines(x=grid, y= beta.ridge[2, ], col = "chocolate1", lwd = 3)
points(x=grid, y= beta.ridge.d[2, ], col = "red")
legend("topright", legend = c(expression(hat(beta)[2]^C), expression(hat(beta)[2])),
       col = c("red", "chocolate1"), pch = c(1, 16))
```



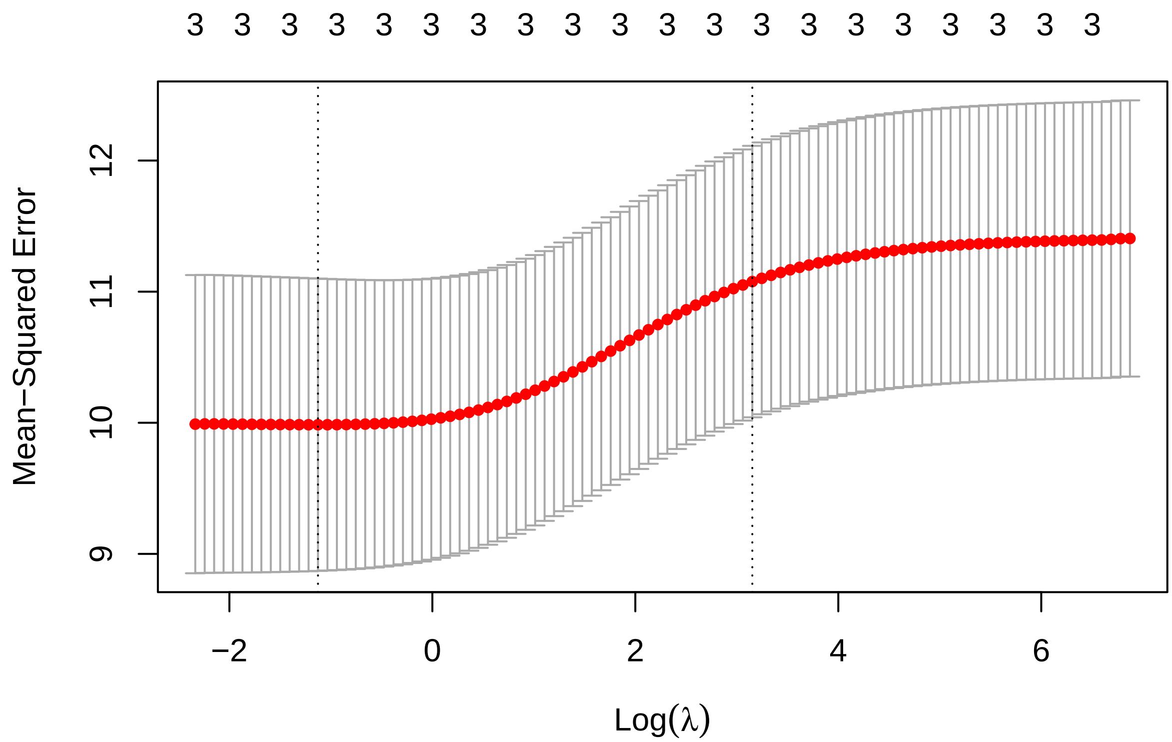

```
plot(x=grid, y= beta.ridge.d[3,], col = "blue", ylim = ylimits, type = "n",
     main = expression(hat(beta)[3]^Ridge ~ "for different" ~ lambda ~ "with constant v.s. baseline"),
     xlab = expression(lambda),
     ylab = "")
grid()
lines(x=grid, y= beta.ridge[3, ], col = "black", lwd = 3)
points(x=grid, y= beta.ridge.d[3, ], col = "blue")
legend("topright", legend = c(expression(hat(beta)[3]^C), expression(hat(beta)[3])),
       col = c("blue", "black"), pch = c(1, 16))
```



e)

We calculate the optimal value for λ using the `glmnet` package. To do this, we utilize the standard seting for the `cv.glmnet` function of the package, which preforms a 10-fold cross-validation procedure.

```
library(glmnet)


ridge.mod <- glmnet(x.sd, data$y, alpha = 0, lambda = grid, tresh = 1e-12, intercept = FALSE)

# Set seed for the cross validation procedure
set.seed(1)

cv.ridge <- cv.glmnet(x.sd, data$y, alpha = 0, intercept = FALSE)
plot(cv.ridge)
```




```
lam.star <- cv.ridge$lambda.min
lam.star
```

```
## [1] 0.3240574
```

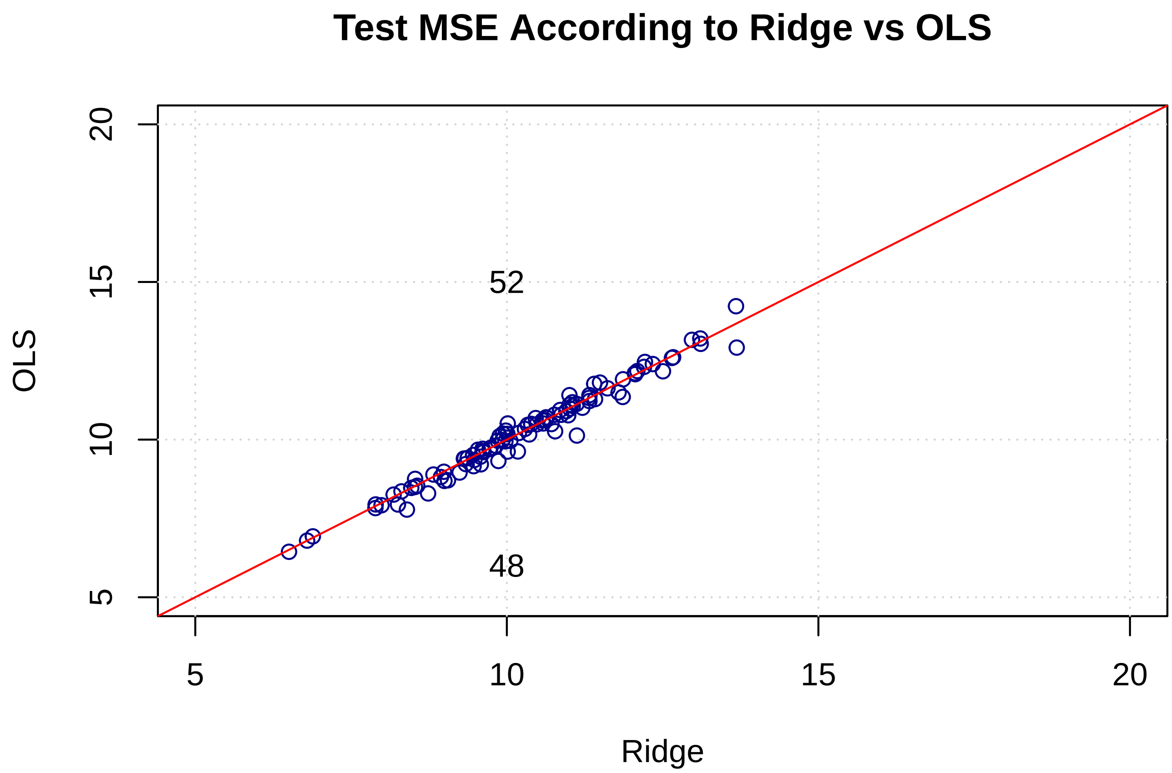
Exercise 2

a)

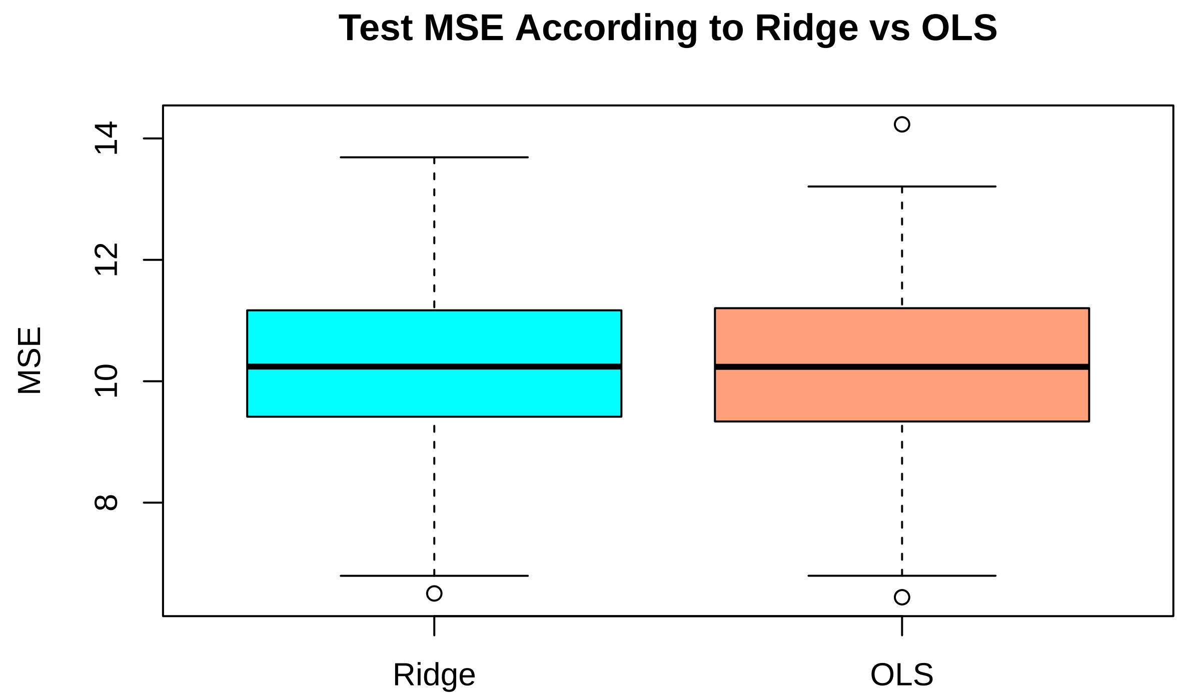
Re-runing the performance test for the Ridge regression 100 times.

```
set.seed(1)

rep <- 100


MSE <- matrix(NA,rep,2)
colnames(MSE) <- c("Ridge", "OLS")

for (i in 1:rep){

  # Data generating process
  train.data <- data.generator(n, sigma, mu, beta.true)
  test.data <- data.generator(n,sigma, mu, beta.true)

  # Storing the x variables into matrixes for convinience
```

```r
    # Train covariates
  x <- cbind(train.data$x.1, train.data$x.2, train.data$x.3)
  x.sd <- cbind(train.data$x.sd.x.1, train.data$x.sd.x.2, train.data$x.sd.x.3)

    #Test covariates

  x.t <- cbind(test.data$x.1, test.data$x.2, test.data$x.3)
  x.sd.t <- cbind(test.data$x.sd.x.1, test.data$x.sd.x.2, test.data$x.sd.x.3)

  # Ridge regession and prediction of the optimal lambda

  ridge.mod <- glmnet(x.sd, train.data$y, alpha = 0,
                      lambda = grid, tresh = 1e-12, intercept = FALSE)

  cv.ridge <- cv.glmnet(x.sd, train.data$y, alpha = 0, intercept = FALSE)

  lam.star <- cv.ridge$lambda.min

  ridge.test <- predict(ridge.mod, s= lam.star, newx = x.sd.t)

  # Ridge MSE

  MSE[i,1] <- mean(( ridge.test - test.data$y)^2)

  # OLS fit

  lm.obj <- lm(y ~ x.sd.x.1 + x.sd.x.2 + x.sd.x.3 -1, data = train.data)

  lm.obj.test <- x.sd.t %*% lm.obj$coefficients

  #OLS MSE

  MSE[i,2] <- mean( (lm.obj.test - test.data$y)^2)
}

# Plot OLS MSE vs. Ridge MSE

# Percentage of values above the 45 degree line

above <- mean(MSE[,1]<=MSE[,2])*100


plot(MSE[,1],MSE[,2], xlim = c(5,20), ylim = c(5,20),
     main = "Test MSE According to Ridge vs OLS",
     xlab = "Ridge", ylab = "OLS", type = "n")
grid()
points(MSE[,1],MSE[,2], col = "darkblue")
abline(a = 0, b = 1, col = "red")
text(10, 15, above)
text(10, 6, 100-above)
```




```r
# Boxplot

boxplot(MSE, ylab = "MSE", col = c("cyan1", "lightsalmon"),  main = "Test MSE According to Ridge vs OLS
```



b)

We propose three scenarios under which the Ridge approach provides an advantage over the OLS estimation:

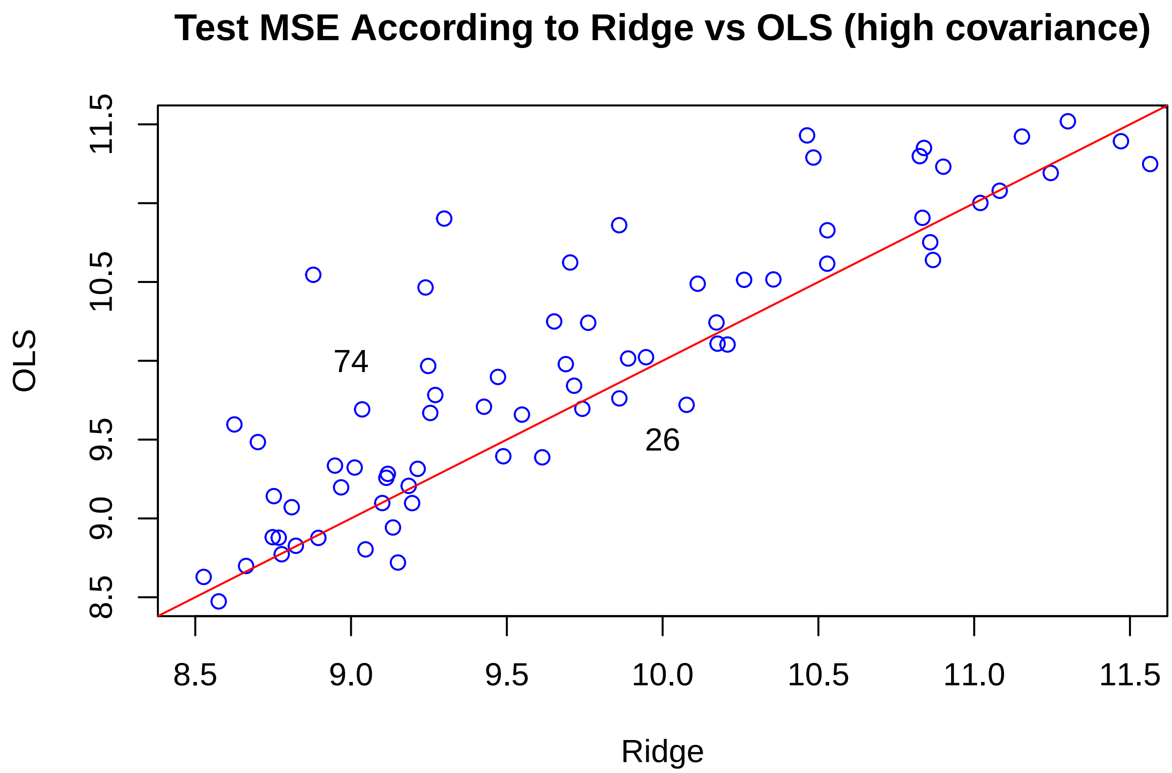
1. Altering the data generating process to have highly correlated regressors.

2. Introducing noise variables into the estimation.

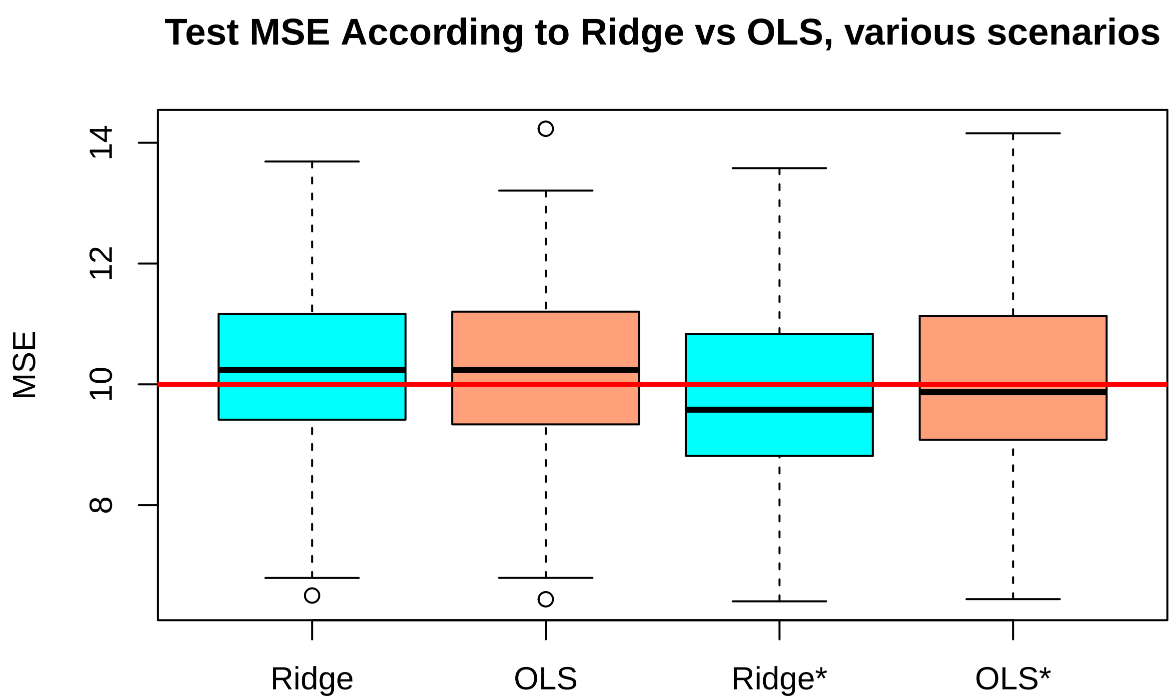
3. Lowering the number of observations as to have $p > n$.

```
##### 1

set.seed(527)
n <- 100


beta.true <- c(0.5, 0.5, -0.5)

# a,b,c are the elements in the diagonal of the VCovar Matrix

covmatrix <- function(a,b,c,d){
  matrix(c(a,sqrt(a*b)-d,sqrt(a*c)-3*d,
           sqrt(a*b)-d,b,sqrt(b*c)-d,
           sqrt(a*c)-3*d,sqrt(b*c)-d,c),
         nrow= 3, ncol= 3, byrow=TRUE)
}
```

```r
sigma2 <- covmatrix(11,15,11,0.01)

mu <- rep(0,3)
rep <- 100
grid <- 10^ seq (5, -2, length = 100)

MSE2 <- matrix(NA,rep,2)
colnames(MSE2) <- c("Ridge*", "OLS*")

for (i in 1:rep){
  train.data <- data.generator(n, sigma2, mu, beta.true)
  test.data <- data.generator(n,sigma2, mu, beta.true)

  x <- cbind(train.data$x.1, train.data$x.2, train.data$x.3)
  x.sd <- cbind(train.data$x.sd.x.1, train.data$x.sd.x.2, train.data$x.sd.x.3)

  x.t <- cbind(test.data$x.1, test.data$x.2, test.data$x.3)
  x.sd.t <- cbind(test.data$x.sd.x.1, test.data$x.sd.x.2, test.data$x.sd.x.3)

  ridge.mod <- glmnet(x.sd, train.data$y, alpha = 0,
                      lambda = grid, tresh = 1e-12,intercept = F
                      )
  cv.ridge <- cv.glmnet(x.sd, train.data$y, alpha = 0,intercept = F)

  lam.star <- cv.ridge$lambda.min

  ridge.test <- predict(ridge.mod, s= lam.star, newx = x.sd.t)

  lm.obj <- lm(train.data$y ~ train.data$x.sd.x.1 + train.data$x.sd.x.2 +
                 train.data$x.sd.x.3 -1)

  lm.obj.test <- x.sd.t %*% lm.obj$coefficients

  MSE2[i,1] <- mean(( ridge.test - test.data$y)^2)
  MSE2[i,2] <- mean( (lm.obj.test - test.data$y)^2)
}

# Plot OLS MSE vs. Ridge MSE


above2 <- mean(MSE2[,1]<=MSE2[,2])*100

plot(MSE2[,1],MSE2[,2], col="blue",xlim = c(8.5,11.5),ylim = c(8.5,11.5),
     xlab = "Ridge",ylab = "OLS", main = "Test MSE According to Ridge vs OLS (high covariance)")
lines(c(0:20),c(0:20), col = "red")
text(9, 10, above2)
text(10, 9.5, 100-above2)
```

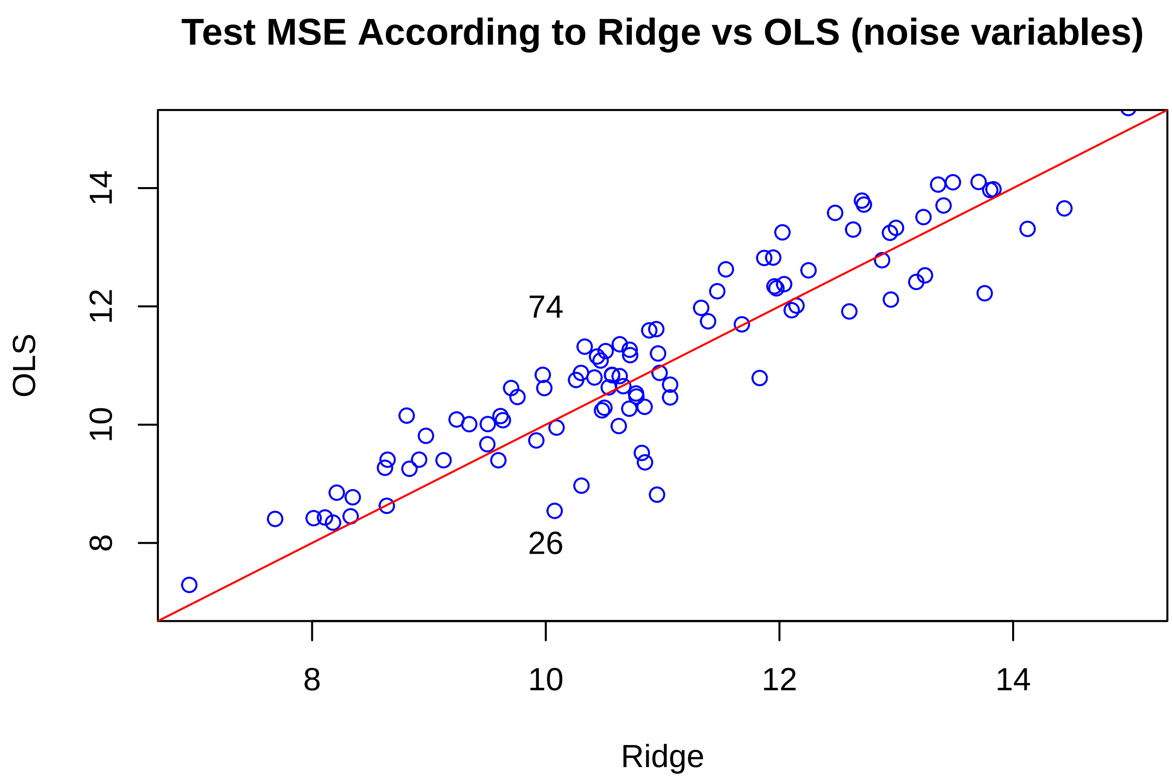



```
# Boxplot

boxplot(cbind(MSE,MSE2),  ylab = "MSE", col = c("cyan1", "lightsalmon"), main = "Test MSE According to I
lines(0:7,rep(10,8),col="red", lwd= 2.5)
```

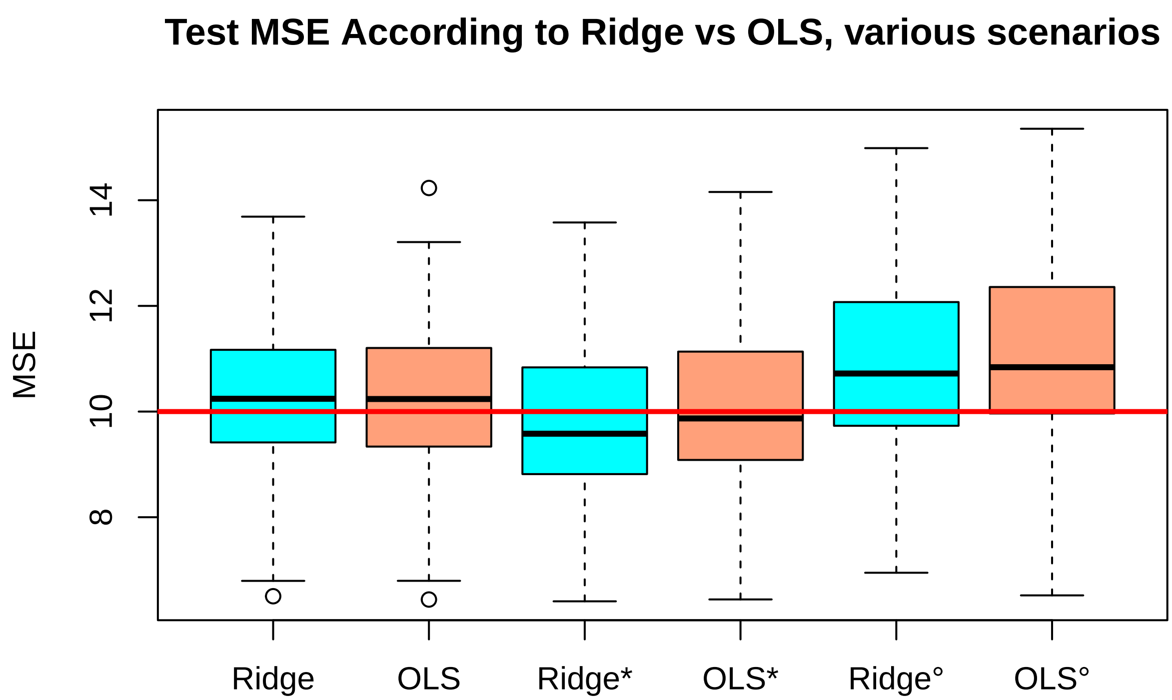



```
##### 2

set.seed(527)
n <- 100
beta.true <- c(0.5, 0.5, -0.5)
mu <- rep(0,3)
rep <- 100
grid <- 10^ seq (5, -2, length = 100)

MSE3 <- matrix(NA,rep,2)

colnames(MSE3) <- c("Ridge°", "OLS°")

sigma <- matrix(c(2,0.1,0.1,0.1,3,0.1,0.1,0.1,4) ,
                 nrow= 3, ncol= 3, byrow=TRUE)

for (i in 1:rep){

  train.data <- data.generator(n, sigma, mu, beta.true)
  test.data <- data.generator(n,sigma, mu, beta.true)

  # Noise variables are already centered
  x4 <- rnorm(n,2,1)
  x5 <- runif(n,1-sqrt(3),1+sqrt(3))
  x6 <- runif(n,2-sqrt(3),2+sqrt(3))
  x7 <- rnorm(n,4,1)
```

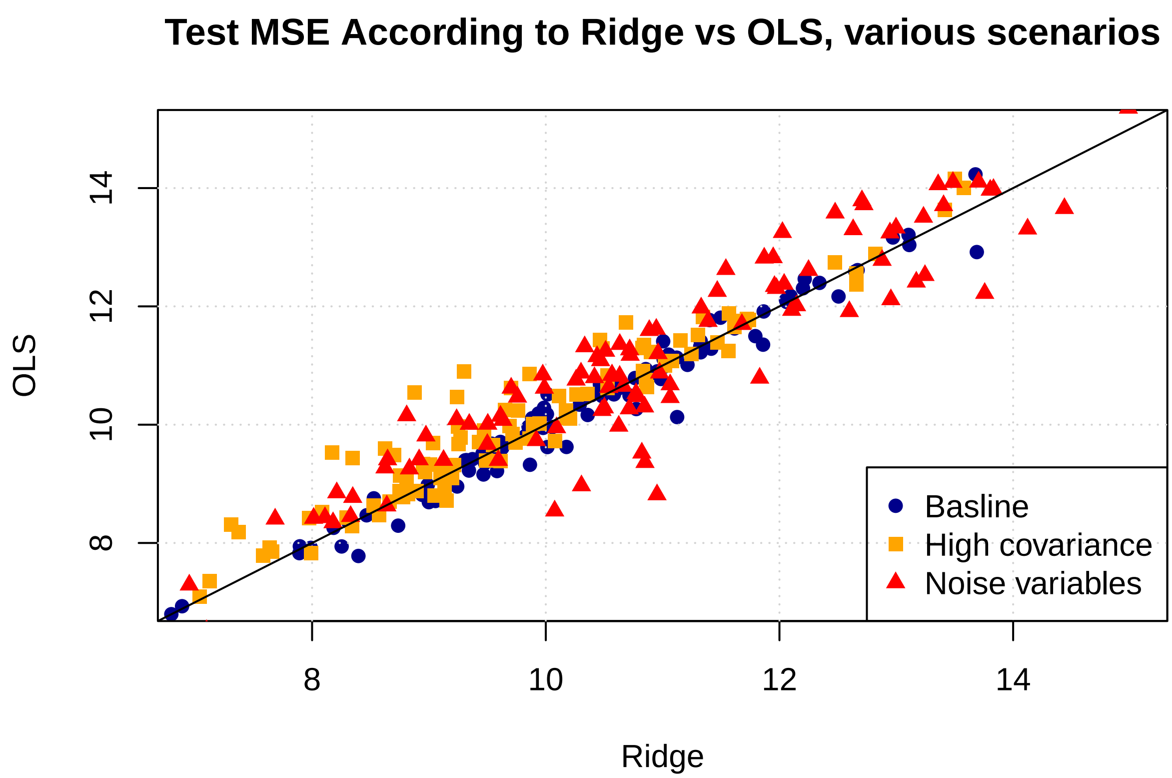
```r
  x8 <- runif(n,-1-sqrt(3),-1+sqrt(3))
  x9 <- rnorm(n,-2,1)
  x10 <- runif(n,-sqrt(3),sqrt(3))
  x4t <- rnorm(n,2,1)
  x5t <- runif(n,1-sqrt(3),1+sqrt(3))
  x6t <- runif(n,2-sqrt(3),2+sqrt(3))
  x7t <- rnorm(n,4,1)
  x8t <- runif(n,-1-sqrt(3),-1+sqrt(3))
  x9t <- rnorm(n,-2,1)
  x10t <- runif(n,-sqrt(3),sqrt(3))


  x.sd <- cbind(train.data$x.sd.x.1, train.data$x.sd.x.2,
        train.data$x.sd.x.3,x4,x5,x6,x7,x8,x9,x10)
  x.sd.t <- cbind(test.data$x.sd.x.1, test.data$x.sd.x.2,
        test.data$x.sd.x.3,x4t,x5t,x6t,x7t,x8t,x9t,x10t)


  ridge.mod <- glmnet(x.sd, train.data$y, alpha = 0,
                      lambda = grid, tresh = 1e-12,intercept = F)

  cv.ridge <- cv.glmnet(x.sd, train.data$y, alpha = 0,intercept = F)

  lam.star <- cv.ridge$lambda.min

  ridge.test <- predict(ridge.mod, s= lam.star, newx = x.sd.t)

  lm.obj <- lm(train.data$y ~ train.data$x.sd.x.1 + train.data$x.sd.x.2 +
               train.data$x.sd.x.3+x4+x5+x6+x7+x8+x9+x10 -1)
  lm.obj.test <- x.sd.t %*% lm.obj$coefficients

  MSE3[i,1] <- mean(( ridge.test - test.data$y)^2)
  MSE3[i,2] <- mean( (lm.obj.test - test.data$y)^2)
}

# Plot OLS MSE vs. Ridge MSE

above3 <- mean(MSE3[,1]<=MSE3[,2])*100

plot(MSE3[,1],MSE3[,2],col="blue",
     xlab = "Ridge",ylab = "OLS",xlim = c(7,15),
     ylim = c(7,15), main = "Test MSE According to Ridge vs OLS (noise variables)")
lines(c(0:20),c(0:20), col = "red")
text(10,12, above2)
text(10,8, 100-above2)
```




```
# Boxplot

boxplot(cbind(MSE,MSE2,MSE3), ylab = "MSE", col = c("cyan1", "lightsalmon"), main = "Test MSE According
lines(0:7,rep(10,8),col="red", lwd= 2.5)
```




```
plot(MSE[,1],MSE[,2],col="darkblue",
     xlab = "Ridge",ylab = "OLS",xlim = c(7,15),
     ylim = c(7,15), main = "Test MSE According to Ridge vs OLS, various scenarios", pch= 16)
grid()
points(MSE2[,1],MSE2[,2],col="orange", pch = 15)
points(MSE3[,1],MSE3[,2],col="red", pch = 17)
lines(c(0:20),c(0:20), col = "black")
legend("bottomright", c("Basline", "High covariance", "Noise variables"),
       col = c("darkblue", "orange", "red"), pch = c(16,15,17))
```




```
####### 3

n_low <- 2
set.seed(55) # same as above

data_n_low <- data.generator(n = n_low, sigma, mu, beta = beta.true)
x.sd_n_low <- cbind(data_n_low$x.sd.x.1, data_n_low$x.sd.x.2, data_n_low$x.sd.x.3)
x_n_low <- cbind(data_n_low$x.1, data_n_low$x.2, data_n_low$x.3)

# OLS fails:

#beta.OLS_n_low <- solve(t(x_n_low) %% x_n_low) %% t(x_n_low) %*% y

# Ridge works:

beta.ridge_n_low <- matrix(NA, ncol = length(grid), nrow = 3)

for (i in 1:length(grid)){
  lam <- grid[i]
  beta.ridge_n_low[,i] <- ridge(x.sd_n_low, data_n_low$y, lam)
}

head(t(beta.ridge_n_low))
```

```
##                [,1]         [,2]         [,3]
## [1,] 1.245860e-05 1.245860e-05 1.245860e-05
```

```
## [2,] 1.466135e-05 1.466135e-05 1.466135e-05
## [3,] 1.725354e-05 1.725354e-05 1.725354e-05
## [4,] 2.030403e-05 2.030403e-05 2.030403e-05
## [5,] 2.389381e-05 2.389381e-05 2.389381e-05
## [6,] 2.811824e-05 2.811824e-05 2.811824e-05
```