MATH 680: Assignment 1

David Fleischer – 260396047

Last Update: 23 January, 2018

Question 1

From our definitions of $\tilde{X}$ and $\tilde{Y}$

$$\begin{aligned}\tilde{X} &= X_{-1} - \mathbf{1}_n \bar{x}^T \\ \tilde{Y} &= Y - \mathbf{1}_n^T \bar{Y},\end{aligned}$$

we find

$$\begin{aligned}\hat{\beta}_{-1} &= \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min} \; \|\tilde{Y} - \tilde{X}\beta\|_2^2 \\ &= \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min} \; \|Y - \mathbf{1}_n \bar{Y} - \left(X_{-1} - \mathbf{1}_n \bar{x}^T\right) \beta_{-1}\|_2^2 \\ &= \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min} \; \|Y - X_{-1}\beta_{-1} - \mathbf{1}_n \left(\bar{Y} - \bar{x}^T \beta_{-1}\right)\|_2^2 \\ &= \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min} \; \|Y - X_{-1}\beta_{-1} - \mathbf{1}_n \beta_1\|_2^2 \quad \text{(by definition of } \beta_1 \text{ above)} \\ &= \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min} \; \|Y - [\mathbf{1}_n, \, X_{-1}] \, [\beta_1, \, \beta_{-1}]\|_2^2 \\ &\equiv \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min} \; \|Y - X\beta\|_2^2.\end{aligned}$$

Therefore, if $\hat{\beta} = \left(\hat{\beta}_1, \, \hat{\beta}_{-1}^T\right)^T \in \mathbb{R}^p$ and

$$\hat{\beta}_1 = \bar{Y} - \bar{x}^T \hat{\beta}_{-1},$$

then $\hat{\beta}$ also solves the uncentered problem

$$\hat{\beta} \equiv \left(\hat{\beta}_1, \, \hat{\beta}_{-1}^T\right)^T = \underset{\beta \in \mathbb{R}^p}{\arg\min} \; \|Y - X\beta\|_2^2,$$

as desired.

Question 2

(a)

Define our objective function $f : \mathbb{R}^p \to \mathbb{R}$ by

$$
\begin{aligned}
f(\beta) &= \|\tilde{Y} - \tilde{X}\beta\|_2^2 + \lambda\|\beta\|_2^2 \\
&= \left(\tilde{Y} - \tilde{X}\beta\right)^T \left(\tilde{Y} - \tilde{X}\beta\right)^T + \lambda\beta^T\beta \\
&= \tilde{Y}^T\tilde{Y} - \tilde{Y}^T\tilde{X}\beta - \beta^T\tilde{X}^T\tilde{Y} + \beta^T\tilde{X}^T\tilde{X}\beta + \lambda\beta^T\beta \\
&= \tilde{Y}^T\tilde{Y} - 2\beta^T\tilde{X}^T\tilde{Y} + \beta^T\tilde{X}^T\tilde{X}\beta + \lambda\beta^T\beta.
\end{aligned}
$$

Therefore

$$
\nabla f(\beta) = -2\tilde{X}^T\tilde{Y} + 2\tilde{X}^T\tilde{X}\beta + 2\lambda\beta,
$$

as desired.

(b)

The Hessian $\nabla^2 f(\beta)$ is given by

$$
\nabla^2 f(\beta) = 2\tilde{X}^T\tilde{X} + 2\lambda\mathbb{I}_{p-1},
$$

where $\mathbb{I}_{p-1}$ is the $(p-1) \times (p-1)$ identity matrix. Note that $2\tilde{X}^T\tilde{X} \in \mathbb{S}_+^{p-1}$ (positive semi-definite) and, for $\lambda > 0$, we have $2\lambda\mathbb{I}_{p-1} \in \mathbb{S}_{++}^{p-1}$ (positive definite). Therefore, for all nonzero vectors $v \in \mathbb{R}^{p-1}$,

$$
\begin{aligned}
v^T\nabla^2 f(\beta)v &= v^T\left(2\tilde{X}^T\tilde{X} + 2\lambda\mathbb{I}_{p-1}\right)v \\
&= 2v^T\tilde{X}^T\tilde{X}v + 2\lambda v^T\mathbb{I}_{p-1}v \\
&= 2\left(\underbrace{\|\tilde{X}v\|_2^2}_{\geq 0} + \underbrace{\lambda\|v\|_2^2}_{>0 \text{ when } \lambda>0}\right) \\
&> 0.
\end{aligned}
$$

Hence,

$$
\nabla^2 f(\beta) = 2\tilde{X}^T\tilde{X} + 2\lambda\mathbb{I}_{p-1} \in \mathbb{S}_{++}^{p-1},
$$

and so f must be strictly convex in β, as desired.

(c)

Suppose a strictly convex function f is globally minimized at distinct points x and y. By the definition of strict convexity

$$
\forall\, t \in (0,1) \quad f(tx + (1-t)y) < tf(x) + (1-t)f(y).
$$

Then, since f is minimized at both x and y, we have $f(x) = f(y)$, so

$$
f(tx + (1-t)y) < tf(x) + (1-t)f(x) = f(x).
$$

However, this implies that the point $z = tx + (1-t)y$ admits a value of f even *smaller* than at x, contradicting our assumption that x is a global minimizer, implying strict convexity is a sufficient condition for a unique global minimizer. Hence, for $\lambda > 0$, we are guaranteed that the above solution will be the unique solution to our penalized least squares problem.

(d)

To write our function computing the ridge coefficients we first set $\nabla f(\beta) = 0$

$$\hat{\beta}_{-1}^{(\lambda)} = \left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\tilde{Y}.$$

For the purpose of computational efficiency we make use of the singular value decomposition of $\tilde{X}$

$$\tilde{X} = UDV^T,$$

for $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{(p-1)\times(p-1)}$ both orthogonal matrices, $U^TU = \mathbb{I}_n$, $V^TV = \mathbb{I}_{p-1}$, and $D \in \mathbb{R}^{n\times(p-1)}$ a diagonal matrix with entries $\{d_j\}_{j=1}^{\min(n,\,p-1)}$ along the main diagonal and zero elsewhere. Hence,

$$\begin{aligned}
\hat{\beta}_{-1}^{(\lambda)} &= \left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\tilde{Y} \\
&= \left(\left(UDV^T\right)^T UDV^T + \lambda VV^T\right)^{-1}\left(UDV^T\right)^T \\
&= V\left(D^TD + \lambda\mathbb{I}_{p-1}\right)^{-1}D^TU^T\tilde{Y}.
\end{aligned}$$

Note that $\left(D^TD + \lambda\mathbb{I}_{p-1}\right)^{-1}D^T$ is diagonal with entries $\frac{d_j}{d_j^2+\lambda}$, $j = 1, ..., p-1$. We exploit this to avoid performing an explicit matrix inversion in our function.

```
ridge_coef <- function(X, y, lam) {
  # Commented-out scaling parameters represent the transformations
  # used to make the output identical to that of
  # coef(MASS::lm.ridge(y ~ X[,-1], lambda = lam))
  Xm1 <- X[,-1]; xbar <- colMeans(Xm1); ytilde <- y - mean(y)

  # center each predictor according to its mean
  Xtilde <- Xm1 - tcrossprod(rep(1, nrow(Xm1)), xbar) # * sqrt(n/(n - 1))

  # compute the SVD on the centered design matrix
  Xtilde_svd <- svd(Xtilde)
  U <- Xtilde_svd$u; d <- Xtilde_svd$d; V <- Xtilde_svd$v

  # compute the inverse (D^T D + lambda I_{p-1})^{-1} D^T
  Dstar <- diag(d/(d^2 + lam))

  # compute ridge coefficients
  b <- V %*% (Dstar %*% crossprod(U, ytilde)) # * 1/xbar * sqrt(n/(n - 1))
  b1 <- mean(y) - crossprod(xbar, b)
  list(b1 = b1, b = b)
}
```

(e)

We first take the expectation of $\hat{\beta}_{-1}^{(\lambda)}$

$$\begin{aligned}\mathbb{E}\left[\hat{\beta}_{-1}^{(\lambda)}\right] &= \mathbb{E}\left[\left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\tilde{Y}\right] \\ &= \left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\mathbb{E}\left[\tilde{Y}\right] \\ &= \left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\tilde{X}\beta_{-1}.\end{aligned}$$

Once again using the SVD of $\tilde{X}$ yields the more elegant expression

$$\begin{aligned}\mathbb{E}\left[\hat{\beta}_{-1}^{(\lambda)}\right] &= \left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\tilde{X}\beta_{-1} \\ &= V\left(D^TD + \lambda\mathbb{I}_{p-1}\right)^{-1}D^TDV^T\beta_{-1} \\ &= VD^*V^T\beta_{-1},\end{aligned}$$

where D^* is a diagonal matrix with entries $\left\{\frac{d_j^2}{d_j^2+\lambda}\right\}_{j=1}^{\min(n,p-1)}$ along the main diagonal and zero elsewhere. We next take the he variance of our centered ridge estimates

$$\begin{aligned}\text{Var}\left(\hat{\beta}_{-1}^{(\lambda)}\right) &= \text{Var}\left(\left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\tilde{Y}\right) \\ &= \left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\text{Var}\left(\tilde{Y}\right)\left(\left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\right)^T \\ &= \left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\text{Var}\left(\tilde{Y}\right)\tilde{X}\left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1} \\ &= \sigma_*^2\left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\tilde{X}\left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\end{aligned}$$

We may yet again apply the SVD of $\tilde{X}$, yielding the more concise expression

$$\text{Var}\left(\hat{\beta}_{-1}^{(\lambda)}\right) = VD^{**}V^T,$$

where D^{**} is a diagonal matrix with entries $\left\{\frac{d_j^2}{\left(d_j^2+\lambda\right)^2}\right\}_{j=1}^{\min(n,p-1)}$ along the main diagonal and zero elsewhere.

We now perform a simulation study to compare sample estimates with their theoretical values $\mathbb{E}\left[\hat{\beta}_{-1}^{(\lambda)}\right]$ and $\text{Var}\left(\hat{\beta}_{-1}^{(\lambda)}\right)$. We first define functions computing the theoretical mean and variance according to our above expressions.

```
ridge_coef_true <- function(X, lam, beta, sigma) {
  n <- nrow(X); p <- ncol(X)
  betam1 <- beta[-1]; Xm1 <- X[,-1]; xbar <- colMeans(Xm1)

  # center each predictor according to its mean
  Xtilde <- Xm1 - tcrossprod(rep(1, nrow(Xm1)), xbar)

  # compute theoretical expectation and variance/covariance matrix
  inv <- solve(crossprod(Xtilde) + lam * diag(p - 1))
```

```r
  b <- inv %*% (crossprod(Xtilde) %*% betam1)
  vcv <- sigma^2 * inv %*% crossprod(Xtilde) %*% inv
  list(b = b, vcv = vcv)
}
```

We may now perform our simulation.

```r
set.seed(124)

# set parameters
nsims <- 1e3
n <- 100
p <- 5
lam <- 4
beta_star <- (-1)^(1:p) * (1:p)
sigma_star <- 1

# generate fixed design matrix
X <- cbind(1, matrix(rnorm(n * (p - 1)), nrow = n))

# compute theoretical mean and variance
true_vals <- ridge_coef_true(X, lam, beta_star, sigma_star)
b_true <- as.vector(true_vals$b)
vcv_true <- true_vals$vcv

# simulate ridge coefficients nsims times
# outputs a matrix with rows corresponding to coefficients
# and columns correspond to simulation number
pt <- proc.time()
b_hat <- replicate(nsims, {
  y <- X %*% beta_star + rnorm(n, 0, sigma_star)
  as.vector(ridge_coef(X, y, lam)$b)
})
proc.time() - pt
```

```
##    user  system elapsed
##   0.152   0.006   0.159
```

```r
# estimate variance of b1, ..., b_p estimates
vcv_hat <- var(t(b_hat))

# print estimated values vs. expected values
b <- rbind(rowMeans(b_hat), b_true, abs(rowMeans(b_hat) - b_true))
rownames(b) <- c("b_hat", "b_true", "abs_err")
round(b, 4)
```

```
##           [,1]    [,2]   [,3]    [,4]
## b_hat   1.8440 -2.9266 3.8250 -4.7900
## b_true  1.8502 -2.9270 3.8271 -4.7904
## abs_err 0.0061  0.0004 0.0022  0.0004
```

```r
# print absolute error between estimated and true variances
round(abs(vcv_true - vcv_hat), 4)
```

```
##       [,1]  [,2]  [,3]  [,4]
## [1,] 1e-04 7e-04 1e-04 2e-04
```

```
## [2,] 7e-04 3e-04 5e-04 5e-04
## [3,] 1e-04 5e-04 1e-04 1e-04
## [4,] 2e-04 5e-04 1e-04 7e-04
```

We see that the empirical sample estimates are very close to their theoretical values, as expected.

Question 3

Note that for the purpose of improved performance we *do not* call our previous `ridge_coef` function. Calling our previous implementation would compute the SVD for each value of λ. This is unnecessary since the SVD is independent of λ (for a fixed design matrix X and response vector y). Instead, we perform the SVD prior to separating the vector of tuning parameters.

```r
ridge_cv <- function(X, y, lam.vec, K) {
  # perform K-fold cross-validation on the ridge regression
  # estimation problem over tuning parameters given in lam.vec
  n <- nrow(X); p <- ncol(X); L <- length(lam.vec)

  # groups to cross-validate over
  folds <- cut(1:n, breaks = K, labels = F)
  # get indices of training subset
  train_idxs <- lapply(1:K, function(i) !(folds %in% i))

  # preallocate empty data structure to store our CV errors
  # for each lambda & fold
  cv_errs_lams_folds <- matrix(0, nrow = L, ncol = K)
  cv_errs_lams_folds <- sapply(train_idxs, function(trn_idx) {
    tst_idx <- !trn_idx # find test subset indices

    # subset data and remove intercept column in the design matrix
    Xm1_trn <- X[trn_idx, -1]; y_trn <- y[trn_idx]
    xbar_trn <- colMeans(Xm1_trn); ytilde_trn <- y_trn - mean(y_trn)

    # center each predictor according to its mean
    Xtilde_trn <- Xm1_trn - tcrossprod(rep(1, nrow(Xm1_trn)), xbar_trn)

    # compute the SVD on the centered design matrix
    # Note: This is done before computing our coefficients for each
    # lambda since the SVD will be identical across the vector of
    # tuning parameters
    Xtilde_trn_svd <- svd(Xtilde_trn)
    U <- Xtilde_trn_svd$u; d <- Xtilde_trn_svd$d; V <- Xtilde_trn_svd$v
    d2 <- d^2

    # preallocate empty data structure to store CV errors
    # for each value of lambda
    cv_errs_lams <- vector(mode = 'numeric', length = L)
    cv_errs_lams <- sapply(lam.vec, function(lam) {
      # compute the inverse (D^T D + lambda I_{p-1})^{-1} D^T
      Dstar <- diag(d/(d2 + lam))
      # compute ridge coefficients
      b_trn <- V %*% (Dstar %*% crossprod(U, ytilde_trn))
      b1_trn <- mean(y_trn) - crossprod(xbar_trn, b_trn)
```

```r
      # fit test data
      yhat_tst <- X[tst_idx,] %*% c(b1_trn, b_trn)
      # compute test error
      sum((y[tst_idx] - yhat_tst)^2)
    })
    cv_errs_lams
  })

  #  weighted average according to group size (some groups may have
  # +/- 1 member depending on whether K can't divide sizes evenly) of
  # cross validation error for each value of lambda
  cv.error <- apply(cv_errs_lams_folds, 1,
                    function(cv_errs_folds) {
                      sum(cv_errs_folds * tabulate(folds))
                    })/n

  # extract the optimal value of our tuning parameter lambda
  # and (re)compute the corresponding coefficient estimates
  best.lam <- lam.vec[cv.error == min(cv.error)]
  best.fit <- ridge_coef(X, y, best.lam)
  b1 <- best.fit$b1
  b <- best.fit$b
  list(b1 = b1, b = b, best.lam = best.lam, cv.error = cv.error)
}
```

Question 4

For this problem we first set some global libraries/functions

```r
library(doParallel)

rmvn <- function(n, p, mu = 0, S = diag(p)) {
  # generates n (potentially correlated) p-dimensional normal deviates
  # given mean vector mu and variance-covariance matrix S
  # NOTE: S must be a positive-semidefinite matrix
  Z <- matrix(rnorm(n * p), nrow = n, ncol = p) # generate iid normal deviates
  C <- chol(S)
  mu + Z %*% C # compute our correlated deviates
}
loss1 <- function(beta, b) sum((b - beta)^2, na.rm = T)
loss2 <- function(X, beta, b) sum((X %*% (beta - b))^2, na.rm = T)
se <- function(x) sd(x)/length(x)
```

and global parameters which remain constant across (a)-(d)

```r
set.seed(124)

# global parameters
nsims <- 50
n <- 100
Ks <- c(5, 10, n)
lams <- 10^seq(-8, 8, 0.5)
```

```r
sigma_star <- sqrt(1/2)

# preallocate empty data structure to store our results
coef_list <- vector(mode = 'list', length = length(Ks) + 1)
names(coef_list) <- c("OLS", "K5", "K10", "Kn")
```

(a)

```r
# set parameters
p <- 50
theta <- 0.5

# generate data
beta_star <- rnorm(p, 0, sigma_star)
SIGMA <- outer(1:(p - 1), 1:(p - 1), FUN = function(a, b) theta^abs(a - b))
X <- cbind(1, rmvn(n, p - 1, 0, SIGMA))

# simulation
pt <- proc.time(); registerDoParallel(cores = 4)
sim <- foreach(1:nsims, .combine = cbind) %dopar% {
  # generate response
  y <- X %*% beta_star + rnorm(n, 0, sigma_star)

  # compute OLS estimates (lambda = 0)
  ols_fit <- ridge_coef(X, y, 0)
  coef_list[[1]] <- c(ols_fit$b1, ols_fit$b)

  # compute the cross-validated ridge estimates for each K
  coef_list[2:(length(Ks) + 1)] <- sapply(Ks, function(k) {
    rcv <- ridge_cv(X, y, lam.vec = lams, K = k)
    list(coefs = c(rcv$b1, rcv$b))
  })

  # compute losses
  l1 <- sapply(coef_list, function(b) loss1(beta_star, b))
  l2 <- sapply(coef_list, function(b) loss2(X, beta_star, b))
  list(l1, l2)
}
proc.time() - pt
```

```
##    user  system elapsed
##  18.131   0.267  11.122
```

```r
# restructure losses for readability
sim_loss <- lapply(1:nrow(sim), function(i) sapply(sim[i,], function(s) s))
names(sim_loss) <- c("Loss 1", "Loss 2")

# compute loss mean across our sims
sim_means <- t(sapply(sim_loss, function(s) rowMeans(s)))
# compute loss std error across our sims
sim_se <- t(sapply(sim_loss, function(s) apply(s, 1, function(x) se(x))))

# report results
```

```r
round(sim_means, 4)
```

```
##             OLS      K5     K10      Kn
## Loss 1  0.7863  0.7222  0.7196  0.7183
## Loss 2 24.5556 23.6037 23.5521 23.5389
```

```r
round(sim_se, 4)
```

```
##           OLS     K5    K10     Kn
## Loss 1 0.0045 0.0038 0.0037 0.0037
## Loss 2 0.0943 0.0913 0.0905 0.0919
```

(b)

```r
# set parameters
p <- 50
theta <- 0.9

# generate data
beta_star <- rnorm(p, 0, sigma_star)
SIGMA <- outer(1:(p - 1), 1:(p - 1), FUN = function(a, b) theta^abs(a - b))
X <- cbind(1, rmvn(n, p - 1, 0, SIGMA))

# simulation
pt <- proc.time(); registerDoParallel(cores = 4)
sim <- foreach(1:nsims, .combine = cbind) %dopar% {
  # generate response
  y <- X %*% beta_star + rnorm(n, 0, sigma_star)

  # compute OLS estimates (lambda = 0)
  ols_fit <- ridge_coef(X, y, 0)
  coef_list[[1]] <- c(ols_fit$b1, ols_fit$b)

  # compute the cross-validated ridge estimates for each K
  coef_list[2:(length(Ks) + 1)] <- sapply(Ks, function(k) {
    rcv <- ridge_cv(X, y, lam.vec = lams, K = k)
    list(coefs = c(rcv$b1, rcv$b))
  })

  # compute losses
  l1 <- sapply(coef_list, function(b) loss1(beta_star, b))
  l2 <- sapply(coef_list, function(b) loss2(X, beta_star, b))
  list(l1, l2)
}
proc.time() - pt
```

```
##    user  system elapsed
##  45.967   0.644  10.947
```

```r
# restructure losses for readability
sim_loss <- lapply(1:nrow(sim), function(i) sapply(sim[i,], function(s) s))
names(sim_loss) <- c("Loss 1", "Loss 2")

# compute loss mean across our sims
```

```r
sim_means <- t(sapply(sim_loss, function(s) rowMeans(s)))
# compute loss std error across our sims
sim_se <- t(sapply(sim_loss, function(s) apply(s, 1, function(x) se(x))))

# report results
round(sim_means, 4)
```

```
##             OLS      K5     K10      Kn
## Loss 1  4.5567  2.8896  2.8670  2.8986
## Loss 2 24.7500 20.7034 20.5895 20.7305
```

```r
round(sim_se, 4)
```

```
##           OLS     K5    K10     Kn
## Loss 1 0.0291 0.0139 0.0141 0.0137
## Loss 2 0.0959 0.0819 0.0817 0.0808
```

(c)

```r
# set parameters
p <- 200
theta <- 0.5

# generate data
beta_star <- rnorm(p, 0, sigma_star)
SIGMA <- outer(1:(p - 1), 1:(p - 1), FUN = function(a, b) theta^abs(a - b))
X <- cbind(1, rmvn(n, p - 1, 0, SIGMA))

# simulation
pt <- proc.time(); registerDoParallel(cores = 4)
sim <- foreach(1:nsims, .combine = cbind) %dopar% {
  # generate response
  y <- X %*% beta_star + rnorm(n, 0, sigma_star)

  # compute OLS estimates (lambda = 0)
  ols_fit <- ridge_coef(X, y, 0)
  coef_list[[1]] <- c(ols_fit$b1, ols_fit$b)

  # compute the cross-validated ridge estimates for each K
  coef_list[2:(length(Ks) + 1)] <- sapply(Ks, function(k) {
    rcv <- ridge_cv(X, y, lam.vec = lams, K = k)
    list(coefs = c(rcv$b1, rcv$b))
  })

  # compute losses
  l1 <- sapply(coef_list, function(b) loss1(beta_star, b))
  l2 <- sapply(coef_list, function(b) loss2(X, beta_star, b))
  list(l1, l2)
}
proc.time() - pt
```

```
##    user  system elapsed
## 126.625   1.705  54.884
```

```r
# restructure losses for readability
sim_loss <- lapply(1:nrow(sim), function(i) sapply(sim[i,], function(s) s))
names(sim_loss) <- c("Loss 1", "Loss 2")

# compute loss mean across our sims
sim_means <- t(sapply(sim_loss, function(s) rowMeans(s)))
# compute loss std error across our sims
sim_se <- t(sapply(sim_loss, function(s) apply(s, 1, function(x) se(x))))

# report results
round(sim_means, 4)
```

```
##            OLS      K5     K10      Kn
## Loss 1 47.5690 47.1207 47.1997 47.5562
## Loss 2 48.7681 48.5443 51.5463 63.9896
```

```r
round(sim_se, 4)
```

```
##           OLS     K5    K10     Kn
## Loss 1 0.0173 0.0043 0.0053 0.0076
## Loss 2 0.1092 0.1097 0.1878 0.2977
```

(d)

```r
# set parameters
p <- 200
theta <- 0.9

# simulation
pt <- proc.time(); registerDoParallel(cores = 4)
sim <- foreach(1:nsims, .combine = cbind) %dopar% {
  # generate response
  y <- X %*% beta_star + rnorm(n, 0, sigma_star)

  # compute OLS estimates (lambda = 0)
  ols_fit <- ridge_coef(X, y, 0)
  coef_list[[1]] <- c(ols_fit$b1, ols_fit$b)

  # compute the cross-validated ridge estimates for each K
  coef_list[2:(length(Ks) + 1)] <- sapply(Ks, function(k) {
    rcv <- ridge_cv(X, y, lam.vec = lams, K = k)
    list(coefs = c(rcv$b1, rcv$b))
  })

  # compute losses
  l1 <- sapply(coef_list, function(b) loss1(beta_star, b))
  l2 <- sapply(coef_list, function(b) loss2(X, beta_star, b))
  list(l1, l2)
}
proc.time() - pt
```

```
##    user  system elapsed
## 155.815   1.970  49.503
```

```r
# restructure losses for readability
sim_loss <- lapply(1:nrow(sim), function(i) sapply(sim[i,], function(s) s))
names(sim_loss) <- c("Loss 1", "Loss 2")

# compute loss mean across our sims
sim_means <- t(sapply(sim_loss, function(s) rowMeans(s)))
# compute loss std error across our sims
sim_se <- t(sapply(sim_loss, function(s) apply(s, 1, function(x) se(x))))

# report results
round(sim_means, 4)
```

```
##             OLS      K5     K10      Kn
## Loss 1 47.7104 47.1484 47.2079 47.5876
## Loss 2 51.3020 51.2726 52.6906 65.2708
```

```r
round(sim_se, 4)
```

```
##            OLS     K5    K10     Kn
## Loss 1 0.0161 0.0044 0.0056 0.0077
## Loss 2 0.1493 0.1492 0.1972 0.2638
```

Question 5

(a)

Taking the gradient of our objective function g with respect to coefficient vector β yields

$$\begin{aligned}\nabla_\beta g(\beta, \sigma^2) &= \nabla_\beta \left(\frac{n}{2} \log \sigma^2 + \frac{1}{2\sigma^2} \|\tilde{Y} - \tilde{X}\beta\|_2^2 + \frac{\lambda}{2} \|\beta\|_2^2 \right) \\ &= \frac{1}{\sigma^2} \left(-\tilde{X}^T \tilde{Y} + \tilde{X}^T \tilde{X} \beta \right) + \lambda \beta,\end{aligned}$$

while the gradient of g with respect to σ^2 is given by

$$\begin{aligned}\nabla_{\sigma^2} g(\beta, \sigma^2) &= \nabla_\beta \left(\frac{n}{2} \log \sigma^2 + \frac{1}{2\sigma^2} \|\tilde{Y} - \tilde{X}\beta\|_2^2 + \frac{\lambda}{2} \|\beta\|_2^2 \right) \\ &= \frac{n}{2\sigma^2} - \frac{1}{2\sigma^4} \|\tilde{Y} - \tilde{X}\beta\|_2^2.\end{aligned}$$

as desired.

(b)

We first consider the objective function in terms of β. We find the Hessian with respect to β

$$
\begin{aligned}
\nabla^2_\beta g\left(\beta, \sigma^2\right) &= \nabla^2_\beta \left(\frac{n}{2}\log\sigma^2 + \frac{1}{2\sigma^2}\|\tilde{Y} - \tilde{X}\beta\|_2^2 + \frac{\lambda}{2}\|\beta\|_2^2\right) \\
&= \nabla_\beta \left(\frac{1}{\sigma^2}\tilde{X}^T\left(-\tilde{Y} + \tilde{X}\beta\right) + \lambda\beta\right) \\
&= \tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}.
\end{aligned}
$$

The symmetric matrix $\tilde{X}^T\tilde{X}$ is always positive semi-definite, and for $\lambda \geq 0$, $\lambda\mathbb{I}_{p-1}$ will also be positive semi-definite (and strictly positive definite when $\lambda > 0$). Thus, the Hessian with respect to β must be positive semi-definite

$$
\nabla^2_\beta g\left(\beta, \sigma^2\right) = \tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1} \in \mathbb{S}_+^{p-1},
$$

and so our objective function $g\left(\beta, \sigma^2\right)$ is convex in β. Now, considering the Hessian with respect to σ^2,

$$
\begin{aligned}
\nabla^2_{\sigma^2} g\left(\beta, \sigma^2\right) &= \nabla^2_{\sigma^2} \left(\frac{n}{2}\log\sigma^2 + \frac{1}{2\sigma^2}\|\tilde{Y} - \tilde{X}\beta\|_2^2 + \frac{\lambda}{2}\|\beta\|_2^2\right) \\
&= \nabla_{\sigma^2} \left(\frac{n}{2\sigma^2} - \frac{1}{2\sigma^4}\|\tilde{Y} - \tilde{X}\beta\|_2^2\right) \\
&= -\frac{n}{2\sigma^4} + \frac{1}{\sigma^6}\|\tilde{Y} - \tilde{X}\beta\|_2^2.
\end{aligned}
$$

For g to be convex in σ^2 we require $\nabla^2_{\sigma^2} g\left(\beta, \sigma^2\right) \geq 0$. However, such a condition is equivalent to

$$
n \geq \frac{2}{\sigma^2}\|\tilde{Y} - \tilde{X}\beta\|_2^2.
$$

As a counterexample consider the following data

```r
set.seed(124)
n <- 20
p <- 100
beta <- rep(0.1, p)
sigma <- sqrt(2)

Xtilde <- matrix(rnorm(n * p), nrow = n)
ytilde <- Xtilde %*% beta + rnorm(n, 0, sigma^2)

(rhs <- as.numeric(2/sigma^2 * crossprod(ytilde - Xtilde %*% beta)))
```

```
## [1] 55.03599
```

```r
n >= rhs
```

```
## [1] FALSE
```

and so it is not the case that $\nabla^2_{\sigma^2} g\left(\beta, \sigma^2\right)$ is (always) nonnegative, implying that our objective function $g\left(\beta, \sigma^2\right)$ is *not* convex in σ^2.

(c)

Let $\bar{\beta}$ be a solution to our maximum likelihood ridge estimation problem such that, for $\lambda > 0$, we have

$$\tilde{Y} - \tilde{X}\bar{\beta} = 0.$$

Since $\bar{\beta}$ is a solution it must satisfy our first order condition

$$\nabla_{\beta} g(\beta, \sigma^2) = \frac{1}{\sigma^2}\left(-\tilde{X}^T\tilde{Y} + \tilde{X}^T\tilde{X}\beta\right) + \lambda\beta = 0 \iff \frac{1}{\sigma^2}\left(\tilde{X}^T\left(-\tilde{Y} + \tilde{X}\beta\right)\right) + \lambda\beta = 0.$$

Thus, for such a solution $\bar{\beta}$ and $\lambda > 0$,

$$\begin{aligned}
0 &= \frac{1}{\sigma^2}\left(\tilde{X}^T\left(-\tilde{Y} + \tilde{X}\bar{\beta}\right)\right) + \lambda\bar{\beta} \\
&= \frac{1}{\sigma^2}\left(\tilde{X}^T\left(-\tilde{Y} + \tilde{Y}\right)\right) + \lambda\bar{\beta} \\
&= \lambda\bar{\beta} \\
\iff \bar{\beta} &= 0.
\end{aligned}$$

Similarly, using our second first order condition $\nabla_{\sigma^2} g(\beta, \sigma^2) = 0$, at $\beta = \bar{\beta}$,

$$\begin{aligned}
\nabla_{\sigma^2} g(\beta, \sigma^2) &= \frac{n}{2\sigma^2} - \frac{1}{2\sigma^4}\|\tilde{Y} - \tilde{X}\beta\|_2^2 \\
&= \frac{n}{2\sigma^2} - \frac{1}{2\sigma^4}\|\tilde{Y} - \tilde{X}\bar{\beta}\|_2^2 \\
&= \frac{n}{2\sigma^2} - \frac{1}{2\sigma^4}\|\tilde{Y} - \tilde{Y}\|_2^2 \\
&= \frac{n}{2\sigma^2} = 0
\end{aligned}$$

This conditions implies that either $n = 0$ or $\sigma^2 \to \infty$. Thus, no such global minimizer could exist.

(d)

Solving our first order conditions

$$\begin{aligned}
\frac{1}{\sigma^2}\left(\tilde{X}^T\left(-\tilde{Y} + \tilde{X}\bar{\beta}\right)\right) + \lambda\bar{\beta} = 0 \\
\frac{n}{2\sigma^2} - \frac{1}{2\sigma^4}\|\tilde{Y} - \tilde{X}\beta\|_2^2 = 0,
\end{aligned}$$

we find the maximum likelihood estimate $\hat{\beta}^{(\lambda,\, ML)}$ to be

$$\hat{\beta}^{(\lambda,\, ML)} = \left(\tilde{X}^T\tilde{X} + \sigma^2\lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\tilde{Y}.$$

and the maximum likelihood estimate $\hat{\sigma}^{2\,(\lambda,\, ML)}$ to be

$$\hat{\sigma}^{2\,(\lambda,\,ML)} = \frac{1}{n}\|\tilde{Y} - \tilde{X}\hat{\beta}^{(\lambda,\,ML)}\|_2^2$$

To compute such estimates we may use the following algorithm: Consider some fixed data set $\mathcal{D} = \{X, Y\}$ and a fixed tuning parameter λ.

(1) Center the data: Center each predictor by its mean $X \mapsto \tilde{X}$, center the response vector by its mean $Y \mapsto \tilde{Y}$.

(2) Have some initial proposal for the estimate $\hat{\sigma}_0^{2\,(\lambda,\,ML)} \in \mathbb{R}^+$.

(3) Compute an initial proposal for $\hat{\beta}_0^{(\lambda,\,ML)}$ based on $\hat{\sigma}_0^{2\,(\lambda,\,ML)}$.

(4) Update our variance estimate $\hat{\sigma}_i^{2\,(\lambda,\,ML)}$ using the previous estimate of $\hat{\beta}_{i-1}^{(\lambda,\,ML)}$.

(5) Update our coefficient estimate $\hat{\beta}_i^{(\lambda,\,ML)}$ using the new estimate of $\hat{\sigma}_i^{2\,(\lambda,\,ML)}$.

(6) Repeat steps (5)-(6) until some convergence criteria is met, say $\|\hat{\sigma}_i^{2\,(\lambda,\,ML)} - \hat{\sigma}_{i-1}^{2\,(\lambda,\,ML)}\|$, is small.

(e)

Our function is as follows

```r
ridge_coef_mle <- function(X, y, lam, tol = 1e-16) {

  # remove leading column, find predictor means, center response
  Xm1 <- X[,-1]; xbar <- colMeans(Xm1); ytilde <- y - mean(y)
  # center each predictor according to its mean
  Xtilde <- sweep(Xm1, 2, xbar)

  # compute the SVD on the centered design matrix
  Xtilde_svd <- svd(Xtilde)
  U <- Xtilde_svd$u; d <- Xtilde_svd$d; V <- Xtilde_svd$v

  ## generate some initial guess for sigma and beta
  sig0 <- rexp(1)
  Dstar <- diag(d/(d^2 + sig0^2 * lam))
  b0 <- V %*% (Dstar %*% crossprod(U, ytilde))

  i <- 1
  repeat {
    # update sigma and beta
    sig_new <- sqrt(1/n * crossprod(ytilde - Xtilde %*% b0))
    Dstar <- diag(d/(d^2 + sig_new^2 * lam))
    b_new <- V %*% (Dstar %*% crossprod(U, ytilde))

    if (abs(sig_new^2 - sig0^2) < tol)
      break

    sig0 <- sig_new; b0 <- b_new; i <- i + 1
  }
  list(niter = i, sigma = as.numeric(sig_new), b = b_new)
}
grad_mle <- function(X, y, lam, b, s) {
  n <- nrow(X)
```

```r
  # remove leading column, find predictor means, center response
  Xm1 <- X[,-1]; xbar <- colMeans(Xm1); ytilde <- y - mean(y)
  # center each predictor according to its mean
  Xtilde <- sweep(Xm1, 2, xbar)

  gb <- 1/s^2 * crossprod(Xtilde, Xtilde %*% b - ytilde) + lam * b
  gs <- n/(2 * s^2)  - 1/(2 * s^4) * crossprod(ytilde - Xtilde %*% b)
  c(grad_b = gb, grad_s = gs)
}
```

(f)

```r
set.seed(124)
n <- 100
p <- 5
lam <- 1
beta_star <- (-1)^(1:p) * rep(5, p)
sigma_star <- sqrt(1/2)

X <- cbind(1, matrix(rnorm(n * (p - 1)), nrow = n))
y <- X %*% beta_star + rnorm(n, 0, sigma_star)

(rcm <- ridge_coef_mle(X, y, lam))
```

```
## $niter
## [1] 9
##
## $sigma
## [1] 0.6559084
##
## $b
##           [,1]
## [1,]  4.976904
## [2,] -5.000078
## [3,]  4.888082
## [4,] -5.017066
```

```r
grad_mle(X, y, lam, rcm$b, rcm$sigma)
```

```
##       grad_b1       grad_b2       grad_b3       grad_b4        grad_s
##  5.178080e-13 -1.419309e-12  4.849454e-13 -9.281464e-13  1.421085e-14
```

as desired.

Question 6

(a)

Consider our objective function

$$f(\beta) = \frac{1}{2}\|\tilde{Y} - \tilde{X}\beta\|_2^2 + \frac{\lambda_1}{2}\|\beta\|_2^2 + \frac{\lambda_2}{2}\sum_{j=2}^{p}(\beta_j - \beta_{j-1})^2 .$$

To show convexity we wish to show $\nabla^2 f(\beta) \in \mathbb{S}_+^{p-1}$. However, it's not immediately obvious how to take such a gradient with our fused sum terms $(b_j - \beta_{j-1})^2$. One way to get around this is to define vector $B \in \mathbb{R}^{p-1}$ given by

$$B = \begin{bmatrix} \beta_2 - \beta_1 \\ \vdots \\ \beta_p - \beta_{p-1} \end{bmatrix}.$$

Then

$$\sum_{j=2}^{p}(\beta_j - \beta_{j-1})^2 = B^T B.$$

In order to achieve our task of expressing the fused sum in terms of the vector β we must next decompose B into a product of β and some matrix. To this end we define matrix $A \in \mathbb{R}^{(p-2)\times(p-1)}$ with entries -1 along the main diagonal and 1 along the upper diagonal, i.e.,

$$A = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix}.$$

Then

$$\begin{aligned} \sum_{j=2}^{p}(\beta_j - \beta_{j-1})^2 &= B^T B \\ &= \beta^T A^T A \beta \\ &\equiv \|A\beta\|_2^2. \end{aligned}$$

Therefore, our objective function can be expressed as

$$\begin{aligned} f(\beta) &= \frac{1}{2}\|\tilde{Y} - \tilde{X}\beta\|_2^2 + \frac{\lambda_1}{2}\|\beta\|_2^2 + \frac{\lambda_2}{2}\|A\beta\|_2^2 \\ &\equiv \frac{1}{2}\tilde{Y}^T\tilde{Y} - \beta^T\tilde{X}^T\tilde{Y} + \frac{1}{2}\beta^T\tilde{X}^T\tilde{X}\beta + \frac{\lambda_1}{2}\beta^T\beta + \frac{\lambda_2}{2}\beta^T A^T A\beta. \end{aligned}$$

Hence

$$\nabla f(\beta) = -\tilde{X}^T\tilde{Y} + \tilde{X}^T\tilde{X}\beta + \lambda_1\beta + \lambda_2 A^T A\beta,$$

admitting the Hessian

$$\nabla^2 f(\beta) = \tilde{X}^T\tilde{X} + \lambda_1\mathbb{I}_{p-1} + \lambda_2 A^T A.$$

Recalling that a matrix multiplied with its transpose must always be positive semi-definite, we find $\tilde{X}^T X$ and $A^T A$ must be positive semi-definite. Thus, since $\lambda_1 > 0$, we find that our sum $\tilde{X}^T\tilde{X} + \lambda_1\mathbb{I}_{p-1} + \lambda_2 A^T A = \nabla^2 f(\beta)$ is positive semi-definite, and so $f(\beta)$ must be strictly convex, as desired.

(b)

We first solve for $\hat{\beta}_{-1}^{(\lambda_1,\,\lambda_2)}$ in (a) by setting $\nabla f(\beta) = 0$

$$
\begin{aligned}
0 &= -\tilde{X}^T\tilde{Y} + \tilde{X}^T\tilde{X}\beta + \lambda_1\beta + \lambda_2 A^T A\beta \\
\tilde{X}^T\tilde{Y} &= \left(\tilde{X}^T\tilde{X} + \lambda_1\mathbb{I}_{p-1} + \lambda_2 A^T A\right)\beta \\
\implies \hat{\beta}_{-1}^{(\lambda_1,\,\lambda_2)} &= M\tilde{X}^T\tilde{Y},
\end{aligned}
$$

where we have set $M = \left(\tilde{X}^T\tilde{X} + \lambda_1\mathbb{I}_{p-1} + \lambda_2 A^T A\right)^{-1}$ for brevity. Therefore

$$
\begin{aligned}
\mathbb{E}\left[\hat{\beta}_{-1}^{(\lambda_1,\,\lambda_2)}\right] &= \mathbb{E}\left[M\tilde{X}^T\tilde{Y}\right] \\
&= M\tilde{X}^T\mathbb{E}\left[\tilde{Y}\right] \\
&= M\tilde{X}^T\beta_{*,\,-1},
\end{aligned}
$$

and

$$
\begin{aligned}
\mathrm{Var}\left(\hat{\beta}_{-1}^{(\lambda_1,\,\lambda_2)}\right) &= \mathrm{Var}\left(M\tilde{X}^T Y\right) \\
&= M\tilde{X}^T\mathrm{Var}\left(\tilde{Y}\right)\tilde{X}M^T \\
&= \sigma_*^2 M\tilde{X}^T\tilde{X}M^T,
\end{aligned}
$$

as desired. We now perform our fused ridge simulation study to test the theoretical values with some empirical estimates. We first define our fused ridge coefficient estimation function (as well as functions permitting us to easily compute the theoretical means and variances of the fused ridge problem)

```
fused_ridge_coef <- function(X, y, lam1, lam2) {
  n <- nrow(X); p <- ncol(X)

  # remove leading column, find predictor means, center response
  Xm1 <- X[,-1]; xbar <- colMeans(Xm1); ytilde <- y - mean(y)
  # center each predictor according to its mean
  Xtilde <- sweep(Xm1, 2, xbar)

  I <- diag(p - 1)
  UD <- cbind(rep(0, p - 2), diag(p - 2)) # upper diagonal matrix
  J <- -1 * cbind(diag(p - 2), rep(0, p - 2)) # diag (p - 2)*(p - 1) matrix
  A <- J + UD

  M <- solve(crossprod(Xtilde) + lam1 * I + lam2 * crossprod(A))
  b <- M %*% crossprod(Xtilde, y)
  b0 <- mean(y) - crossprod(xbar, b)
  return(list(b0 = b0, b = b))
}
```

```r
fused_ridge_coef_params <- function(X, lam1, lam2, beta, sigma) {
  # Returns theoretical means and variances for the fused ridge problem
  # Note: Omits intercept term b0
  n <- nrow(X); p <- ncol(X)

  # remove leading column, remove intercept
  Xm1 <- X[,-1]; betam1 <- beta[-1]
  # find predictor means, center response
  xbar <- colMeans(Xm1); ytilde <- y - mean(y)
  # center each predictor according to its mean
  Xtilde <- sweep(Xm1, 2, xbar)

  I <- diag(p - 1)
  UD <- cbind(rep(0, p - 2), diag(p - 2)) # upper diagonal matrix
  J <- -1 * cbind(diag(p - 2), rep(0, p - 2)) # diag (p - 2)*(p - 1) matrix
  A <- J + UD

  M <- solve(crossprod(Xtilde) + lam1 * I + lam2 * crossprod(A))
  b <- M %*% crossprod(Xtilde, (Xtilde %*% betam1))

  vcv <- matrix(0, nrow = p - 1, ncol = p - 1)
  if (n > p) { # when n > p this matrix multiplication routine is quicker
    vcv <- sigma^2 * M %*% tcrossprod(crossprod(Xtilde), M)
  } else { # when p > n this matrix multiplication routine is quicker
   vcv <- sigma^2 * tcrossprod(M, Xtilde) %*% tcrossprod(Xtilde, M)
  }

  list(b = b, vcv = vcv)
}
```

We now simulate some data to test our estimates

```r
set.seed(124)

# set parameters
nsims <- 1e4
n <- 1e2
p <- 5
lam1 <- 1
lam2 <- 1
sigma_star <- 1
beta_star <- rnorm(p)

# generate (fixed) design matrix
X <- cbind(rep(1, n), matrix(rnorm(n * (p - 1)), nrow = n, ncol = p - 1))

# compute expected parameter values
par_true <- fused_ridge_coef_params(X, lam1, lam2, beta_star, sigma_star)
b_true <- as.vector(par_true$b)
vcv_true <- par_true$vcv

# simulate our fused ridge coefficients nsims times
# outputs a matrix with rows corresponding to coefficients
# and columns correspond to simulation number
```

```r
pt <- proc.time()
b_hat <- replicate(nsims, {
  y <- X %*% beta_star + rnorm(n, 0, sigma_star) # generate response
  as.vector(fused_ridge_coef(X, y, lam1, lam2)$b)
})
proc.time() - pt
```

```
##    user  system elapsed
##   1.697   0.011   1.719
```

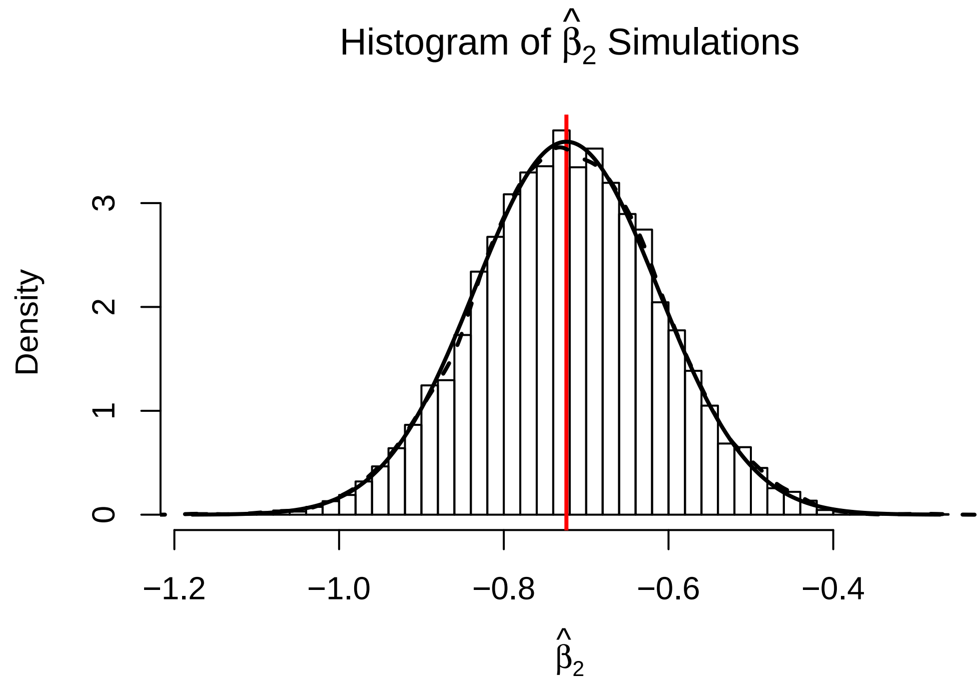
```r
# estimate variance of b2, ..., b_p estimates
vcv_hat <- var(t(b_hat))

# print estimated fused ridge coefficients vs. expected values
b <- rbind(rowMeans(b_hat), b_true)
rownames(b) <- c("b_hat", "b_true")
round(b, 4)
```

```
##          [,1]    [,2]   [,3]   [,4]
## b_hat  0.0316 -0.7226 0.2226 1.3899
## b_true 0.0313 -0.7240 0.2235 1.3920
```

```r
# print absolute error between estimated and true fused ridge variances
round(abs(vcv_true - vcv_hat), 4)
```

```
##       [,1]  [,2]  [,3]  [,4]
## [1,] 2e-04 1e-04 1e-04 1e-04
## [2,] 1e-04 1e-04 1e-04 2e-04
## [3,] 1e-04 1e-04 0e+00 1e-04
## [4,] 1e-04 2e-04 1e-04 3e-04
```

As a final point, we may look at the simulations of $\hat{\beta}_2^{(\lambda_1,\lambda_2)}$ and compare it with it's theoretical distribution. Note that the estimates $\hat{\beta}^{(\lambda_1,\lambda_2)} = M\tilde{X}^T\tilde{Y}$ are normally distributed because they are a linear combination of $\tilde{Y} \sim \mathcal{N}(\tilde{X}\beta, \sigma^2)$ (when our noise terms $\epsilon \sim \mathcal{N}(0, \sigma^2)$). We visualize the histogram of the $\hat{\beta}_2^{(\lambda_1,\lambda_2)}$ simulations with its empirical and theoretical densities overlaid (dashed, solid), along with its expected value (vertical line) below.


Histogram of $\hat{\beta}_2$ Simulations
Density
0
1
2
3
−1.2
−1.0
−0.8
−0.6
−0.4
$\hat{\beta}_2$