Assignment 1

David Fleischer – 260396047

Last Update: 17 January, 2018

Question 1

We wish to show that $\hat{\beta} = \left(\hat{\beta}_1,\, \hat{\beta}_{-1}^T\right)^T$ given by

$$
\begin{aligned}
\hat{\beta}_{-1} &= \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min}\, \|\tilde{Y} - \tilde{X}\beta\|_2^2 \\
\hat{\beta}_1 &= \bar{Y} - \bar{x}^T \hat{\beta}_{-1}
\end{aligned}
$$

is a global minimizer of the least squares problem

$$
\hat{\beta} = \underset{\beta \in \mathbb{R}^{p}}{\arg\min}\, \|Y - X\beta\|_2^2.
$$

Solution 1

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

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

Then

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

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

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

as desired.

Question 2

Consider the (centered) ridge regression problem of estimating β_* with the ℓ_2 penalized least squares regression coefficients $\hat{\beta}^{(\lambda)} = \left(\hat{\beta}_1^{(\lambda)},\, \hat{\beta}_{-1}^{(\lambda)\,T}\right)^T$ defined by

$$\begin{aligned}
\hat{\beta}_{-1}^{(\lambda)} &= \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min}\ \|\tilde{Y} - \tilde{X}\beta\|_2^2 + \lambda\|\beta\|_2^2 \\
\hat{\beta}_1^{(\lambda)} &= \bar{Y} - \bar{x}^T \hat{\beta}_{-1}^{(\lambda)}
\end{aligned}$$

(a)

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

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

Therefore, taking the gradient of our function $\nabla f(\beta)$ we find

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

as desired.

(b)

The second order gradient $\nabla^2 f(\beta)$ yields

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

where $\mathbb{I}_{p-1}$ is the $(p-1)\times(p-1)$ identity matrix. Note that $2\tilde{X}^T\tilde{X} \in \mathbb{S}_+^{p-1}$ is positive semi-definite and, with $\lambda > 0$, $2\lambda\mathbb{I}_{p-1} \in \mathbb{S}_+^{p-1}$, i.e. $2\lambda\mathbb{I}_{p-1}$ is also positive semi-definite. Therefore, since a sum of positive semi-definite matrices is also positive semi-definite, we find

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

and so f must be strictly convex in β.

(c)

Strict convexity implies that the global minimizer must be unique, and so 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 solving for the ridge coefficients we first note that setting $\nabla f(\beta) = 0$ yields

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

where $\left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)$ is guaranteed to be nonsingular (for $\lambda \neq 0$) because it will have have full rank via the identity matrix. For the purpose of computational efficiency we make use of the singular value decomposition on $\tilde{X}$

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

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. Then

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

Note that $D^TD + \lambda\mathbb{I}_{p-1}$ is a diagonal $(p-1)\times(p-1)$ matrix with entries $\left\{d_j^2 + \lambda\right\}_{j=1}^{p-1}$ along the main diagonal, and so the inverse $\left(D^TD + \lambda\mathbb{I}_{p-1}\right)^{-1}$ will also be diagonal with entries $\left\{\frac{1}{d_j^2+\lambda}\right\}_{j=1}^{p-1}$. We exploit this to avoid performing a matrix inversion in our code. For brevity we let

$$D^* = \left(D^TD + \lambda I_{p-1}\right)^{-1}D^T$$

so that

$$\hat{\beta}^{(\lambda)} = VD^*U^T\tilde{Y}$$

We present a function written in `R` performing such calculations below.

```r
ridge_coef <- function(X, y, lam) {
  Xm1 <- X[,-1] # remove leading column of 1's marking the intercept

  ytilde <- y - mean(y) # center response
  xbar <- colMeans(Xm1) # find predictor means
  Xtilde <- sweep(Xm1, 2, xbar) # center each predictor according to its mean
```

```r
  # 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))

  b <- V %*% (Dstar %*% crossprod(U, ytilde))
  b1 <- mean(y) - crossprod(xbar, b)
  return (list(b1 = b1, b = b))
}
```

Note the choice to use `V %*% (Dstar %*% crossprod(U, ytilde))` to compute the matrix product $VD^*U^T\tilde{Y}$ as opposed to the (perhaps more intuitive) `V %*% Dstar %*% t(U) %*% ytilde`. Such a choice can be justified via the following matrix multiplication benchmarks (for the cases of $n >> p$ and $p >> n$)

```r
library(microbenchmark)

#===== Large n case =====#
set.seed(124)

# set parameters
n <- 1e3
p <- 1e2
lam <- 1

# generate data
X <- matrix(rnorm(n * p), nrow = n, ncol = p)
beta <- rnorm(p)
eps <- rnorm(n)
y <- X %*% beta + eps

ytilde <- y - mean(y)
xbar <- colMeans(X)
Xtilde <- sweep(X, 2, xbar)

# compute decomposition
Xtilde_svd <- svd(Xtilde)
U <- Xtilde_svd$u
d <- Xtilde_svd$d
V <- Xtilde_svd$v
Dstar <- diag(d/(d^2 + lam))

# define multiplication functions
f1 <- function() V %*% Dstar %*% t(U) %*% ytilde
f2 <- function() V %*% Dstar %*% (t(U) %*% ytilde)
f3 <- function() V %*% (Dstar %*% (t(U) %*% ytilde))
f4 <- function() V %*% (Dstar %*% crossprod(U, ytilde))
f5 <- function() V %*% crossprod(Dstar, crossprod(U, ytilde))

# test speed
microbenchmark(f1(), f2(), f3(), f4(), f5(), times = 100, unit = "us")
```

```
## Unit: microseconds
##   expr      min        lq       mean     median        uq       max neval
##   f1() 8623.592 9578.7790 12320.5260 10201.6875 11010.958 82243.573   100
##   f2() 1107.524 1271.6625  2596.1848  1372.3665  2014.739 32190.036   100
##   f3()  375.482  456.0795   829.7821   529.0970  1012.103  4898.432   100
##   f4()  130.653  140.4425   166.1650   153.7565   170.759   493.951   100
##   f5()  127.160  138.7330   178.9998   151.8880   168.904  1180.798   100
```

```r
#===== Large p case =====#
set.seed(124)

# set parameters
n <- 1e2
p <- 1e3
lam <- 1

# generate data
X <- matrix(rnorm(n * p), nrow = n, ncol = p)
beta <- rnorm(p)
eps <- rnorm(n)
y <- X %*% beta + eps

# define multiplication functions
f1 <- function() V %*% Dstar %*% t(U) %*% ytilde
f2 <- function() V %*% Dstar %*% (t(U) %*% ytilde)
f3 <- function() V %*% (Dstar %*% (t(U) %*% ytilde))
f4 <- function() V %*% (Dstar %*% crossprod(U, ytilde))
f5 <- function() V %*% crossprod(Dstar, crossprod(U, ytilde))

# test speed
microbenchmark(f1(), f2(), f3(), f4(), f5(), times = 100, unit = "us")
```

```
## Unit: microseconds
##   expr      min        lq       mean     median        uq       max neval
##   f1() 8562.141 9452.2725 12159.7681 10149.8330 10989.846 43346.360   100
##   f2() 1097.999 1249.2695  2002.2149  1376.5490  1971.101 33102.035   100
##   f3()  368.061  438.8635   662.2515   474.8035   568.714  3872.856   100
##   f4()  130.172  148.3530   180.0820   155.7905   170.920  1149.647   100
##   f5()  127.735  142.5110   162.9198   150.5255   165.117   529.439   100
```

(e)

We take the expectation of $\hat{\beta}^{(\lambda)}$

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

and variance

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

Question 3

Question 4

For this problem we first define some additional functions and set some global parameters which remain constant across (a)-(d)

```
set.seed(124)
# global parameters
nsims <- 50
lams <- 10^seq(-8, 8, 0.5)
sigma_star <- sqrt(1/2)
```

(a)

```
# set parameters
n <- 100
p <- 50
theta <- 0.5

# generate data
beta_star <- rnorm(p, 0, sigma_star)
Z <- matrix(rnorm(n * (p - 1)), nrow = n, ncol = p - 1) # indep. normal deviates
SIGMA <- outer(1:(p - 1), 1:(p - 1), FUN = function(a, b) theta^abs(a - b))
C <- chol(SIGMA)
X <- cbind(rep(1, n), Z %*% C) # correlated normal deviates

# simulate noise and response
sim <- replicate(nsims, {
  eps <- rnorm(n, 0, sigma_star)
  y <- X %*% beta_star + eps

})
```

(b)

(c)

(d)

Question 5

(a)

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

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

and the gradient of g with respect to σ^2 yields

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

(b)

(c)

(d)

(e)

(f)

Question 6

(a)

Consider our objective function

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

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 $\left(b_j - \beta_{j-1} \right)^2$. 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 second order gradient

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

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)
  Xm1 <- X[,-1] # remove leading column of 1's marking the intercept

  ytilde <- y - mean(y) # center response
  xbar <- colMeans(Xm1) # find predictor means
  Xtilde <- sweep(Xm1, 2, xbar) # center each predictor according to its mean

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

```r
  Xm1 <- X[,-1] # remove leading column of 1's marking the intercept
  betam1 <- beta[-1] # remove intercept term

  xbar <- colMeans(Xm1) # find predictor means
  Xtilde <- sweep(Xm1, 2, xbar) # center each predictor according to its mean

  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)
  }

  return (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 solution nsim times
# outputs a matrix with rows corresponding to coefficients
# and columns correspond to simulation number
pt <- proc.time()
b_hat <- replicate(nsims, {
  eps <- rnorm(n, 0, sigma_star) # generate noise
  y <- X %*% beta_star + eps # generate response

  b_hat <- unlist(fused_ridge_coef(X, y, lam1, lam2))
```

```r
  return (b_hat)
})
proc.time() - pt
```

```
##    user  system elapsed
##   1.770   0.024   1.967
```

```r
vcv_hat <- var(t(b_hat[-1,])) # estimated variance of b1, ..., b_p estimates

# print estimated fused ridge coefficients vs. expected values
b <- rbind(rowMeans(b_hat)[-1], b_true)
rownames(b) <- c("b_hat", "b_true")
round(b, 4)
```

```
##             b1      b2     b3     b4
## 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)
```

```
##       b1    b2    b3    b4
## b1 2e-04 1e-04 1e-04 1e-04
## b2 1e-04 1e-04 1e-04 2e-04
## b3 1e-04 1e-04 0e+00 1e-04
## b4 1e-04 2e-04 1e-04 3e-04
```

As a case study, 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.