Singular Value Decomposition for Lasso and Ridge Regression

Problem 1

a) To show that the condition number of the modified matrix is shifted, we start by observing that since $\mathbf{X}'\mathbf{X}$ is square and symmetric, we can write it using SVD, where the diagonal matrix $\mathbf{D}$ contains the eigenvalues of $\mathbf{X}'\mathbf{X}$. Then, solving for the eigenvalues of $\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}$ we get

$$\begin{aligned}\mathbf{X}'\mathbf{X} + \lambda\mathbf{I} &= \mathbf{V}\mathbf{D}\mathbf{V}' + \lambda\mathbf{I} \\ &= \mathbf{V}(\mathbf{D} + \lambda\mathbf{I})\mathbf{V}' \\ &= \mathbf{V}\begin{bmatrix}\lambda_1 + \lambda & & & \\ & \lambda_2 + \lambda & & \\ & & \ddots & \\ & & & \lambda_{p+1} + \lambda\end{bmatrix}\mathbf{V}'.\end{aligned}$$

We observe now that the eigenvalues of $\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}$ are simply the eigenvalues of $\mathbf{X}'\mathbf{X}$ shifted by λ. Therefore, the condition number $\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}$ is

$$\kappa(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}) = \frac{\lambda_1(\mathbf{X}'\mathbf{X}) + \lambda}{\lambda_{p+1}(\mathbf{X}'\mathbf{X}) + \lambda}.$$

b)

```
# Load necessary libraries
library(tidyverse)
library(Matrix)
library(pracma)
library(glmnet)
```

```
# Read the dataset
bikedata <- read_csv("reducedbikedata2011.csv")
bikedata <- na.omit(bikedata)

# Prepare the data matrix X for OLS regression
predictor_columns <- setdiff(names(bikedata), c('Unnamed: 0', 'X', '...1'))
X_ols <- bikedata %>%
  select(all_of(predictor_columns)) %>%
  add_column(intercept = 1, .before = 1)

X_ols <- data.frame(lapply(X_ols, function(x) as.numeric(as.character(x))))
X_ols_matrix <- as.matrix(X_ols)
```

b) *i)* When the condition number is near 0, $\mathbf{X}'\mathbf{X}+\lambda\mathbf{I}$ is very unstable, as the maximum eigenvalue is much larger than the minimum eigenvalue. As $\lambda \to \infty$, we notice that the condition number approaches one, indicating the minimum and maximum eigenvalues are near equal. When the minimum and maximum eigenvalues are near equal, the matrix is much easier to invert, and is there for more stable.

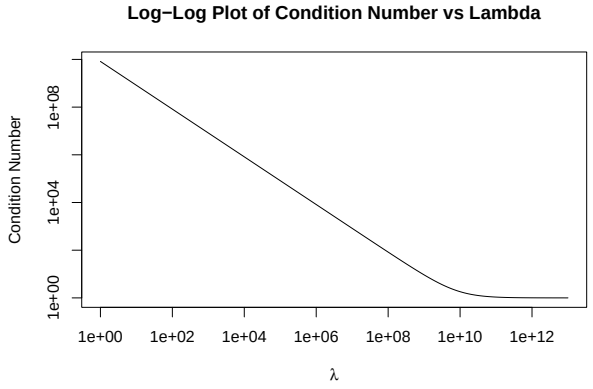
```
calc_condition_number <- function(lambda, eigenvalues_XtX) {
  lambda_1 <- max(eigenvalues_XtX)
  lambda_m <- min(eigenvalues_XtX)
  condition_number <- (lambda_1 + lambda) / (lambda_m + lambda)
```

```
  return(condition_number)
}

XtX <- t(X_ols_matrix) %*% X_ols_matrix
eigenvalues_XtX <- eigen(XtX, only.values = TRUE)$values
lambda_sequence <- 10^seq(0, 13, length.out = 1000)
condition_numbers <- numeric(length(lambda_sequence))

for(i in seq_along(lambda_sequence)) {
  condition_numbers[i] <- calc_condition_number(lambda_sequence[i], eigenvalues_XtX)
}

plot(lambda_sequence, condition_numbers, log = "xy", type = "l",
     xlab = expression(lambda), ylab = "Condition Number",
     main = "Log-Log Plot of Condition Number vs Lambda")
```



b) *ii)* From the formula for the fitted values in ridge regression

$$\hat{\mathbf{Y}}_R = \sum_{i=0}^{p} \frac{\sigma_i^2}{\sigma_i^2 + \lambda} (\mathbf{u}_i' \mathbf{Y}) \mathbf{u}_i,$$

we observe that as $\lambda \to \infty$, the fitted values for the regression approach 0, which is likely not reasonable. Therefore, it is not reasonable to use a very large λ for ridge regression. On the other hand, small lambda values near 0 give the same fitted values as those obtained from fitting a traditional least squares model. A balance is to be achieved between the two, as to not the defeat the purpose of using ridge over regular least squares.

Problem 2

a)

```
# Load in data
swiss <- datasets::swiss
summary(swiss)
```

```
##    Fertility      Agriculture     Examination      Education
```

```
##  Min.   :35.00   Min.   : 1.20   Min.   : 3.00   Min.   : 1.00
##  1st Qu.:64.70   1st Qu.:35.90   1st Qu.:12.00   1st Qu.: 6.00
##  Median :70.40   Median :54.10   Median :16.00   Median : 8.00
##  Mean   :70.14   Mean   :50.66   Mean   :16.49   Mean   :10.98
##  3rd Qu.:78.45   3rd Qu.:67.65   3rd Qu.:22.00   3rd Qu.:12.00
##  Max.   :92.50   Max.   :89.70   Max.   :37.00   Max.   :53.00
##     Catholic       Infant.Mortality
##  Min.   :  2.150   Min.   :10.80
##  1st Qu.:  5.195   1st Qu.:18.15
##  Median : 15.140   Median :20.00
##  Mean   : 41.144   Mean   :19.94
##  3rd Qu.: 93.125   3rd Qu.:21.70
##  Max.   :100.000   Max.   :26.60
```

```
x <- model.matrix(Fertility~., swiss)[,-1]
y <- swiss$Fertility
lambda <- 10^seq(10, -2, length = 100)

# Form train and test sets
set.seed(489)
train = sample(1:nrow(x), nrow(x)/2)
test = (-train)
ytest = y[test]
```

b)

```
train_indices <- sample(1:nrow(x), nrow(x)/2)
test_indices <- setdiff(1:nrow(x), train_indices)
x_train <- x[train_indices, ]
y_train <- y[train_indices]
x_test <- x[test_indices, ]
y_test <- y[test_indices]

# run OLS model on training data
ols_model <- lm(Fertility ~ ., data = swiss, subset=train_indices)
summary(ols_model)
```

```
##
## Call:
## lm(formula = Fertility ~ ., data = swiss, subset = train_indices)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.6123  -5.8985   0.3505   3.7291  15.6139
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      57.61010   17.56840   3.279  0.00442 **
## Agriculture      -0.15748    0.12745  -1.236  0.23341
## Examination      -0.23538    0.45798  -0.514  0.61391
## Education        -0.92411    0.26249  -3.521  0.00262 **
## Catholic          0.10465    0.06519   1.605  0.12686
## Infant.Mortality  1.55636    0.57091   2.726  0.01437 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.279 on 17 degrees of freedom
## Multiple R-squared:  0.745,  Adjusted R-squared:   0.67
## F-statistic: 9.933 on 5 and 17 DF,  p-value: 0.0001402
```

c)

```
ridge_model <- glmnet(x_train, y_train, alpha=0, lambda=lambda)
print(ridge_model)
```

```
##
## Call:  glmnet(x = x_train, y = y_train, alpha = 0, lambda = lambda)
##
##     Df  %Dev    Lambda
## 1    5  0.00 1.000e+10
## 2    5  0.00 7.565e+09
## 3    5  0.00 5.722e+09
## 4    5  0.00 4.329e+09
## 5    5  0.00 3.275e+09
## 6    5  0.00 2.477e+09
## 7    5  0.00 1.874e+09
## 8    5  0.00 1.417e+09
## 9    5  0.00 1.072e+09
## 10   5  0.00 8.111e+08
## 11   5  0.00 6.136e+08
## 12   5  0.00 4.642e+08
## 13   5  0.00 3.511e+08
## 14   5  0.00 2.656e+08
## 15   5  0.00 2.009e+08
## 16   5  0.00 1.520e+08
## 17   5  0.00 1.150e+08
## 18   5  0.00 8.697e+07
## 19   5  0.00 6.579e+07
## 20   5  0.00 4.977e+07
## 21   5  0.00 3.765e+07
## 22   5  0.00 2.848e+07
## 23   5  0.00 2.154e+07
## 24   5  0.00 1.630e+07
## 25   5  0.00 1.233e+07
## 26   5  0.00 9.326e+06
## 27   5  0.00 7.055e+06
## 28   5  0.00 5.337e+06
## 29   5  0.00 4.037e+06
## 30   5  0.00 3.054e+06
## 31   5  0.00 2.310e+06
## 32   5  0.00 1.748e+06
## 33   5  0.00 1.322e+06
## 34   5  0.00 1.000e+06
## 35   5  0.01 7.565e+05
## 36   5  0.01 5.722e+05
## 37   5  0.01 4.329e+05
```

```
## 38   5  0.01 3.275e+05
## 39   5  0.02 2.477e+05
## 40   5  0.02 1.874e+05
## 41   5  0.03 1.417e+05
## 42   5  0.04 1.072e+05
## 43   5  0.05 8.111e+04
## 44   5  0.07 6.136e+04
## 45   5  0.09 4.642e+04
## 46   5  0.12 3.511e+04
## 47   5  0.16 2.656e+04
## 48   5  0.21 2.009e+04
## 49   5  0.27 1.520e+04
## 50   5  0.36 1.150e+04
## 51   5  0.47 8.697e+03
## 52   5  0.63 6.579e+03
## 53   5  0.83 4.977e+03
## 54   5  1.09 3.765e+03
## 55   5  1.43 2.848e+03
## 56   5  1.88 2.154e+03
## 57   5  2.47 1.630e+03
## 58   5  3.23 1.233e+03
## 59   5  4.21 9.330e+02
## 60   5  5.47 7.060e+02
## 61   5  7.07 5.340e+02
## 62   5  9.08 4.040e+02
## 63   5 11.57 3.050e+02
## 64   5 14.59 2.310e+02
## 65   5 18.17 1.750e+02
## 66   5 22.30 1.320e+02
## 67   5 26.92 1.000e+02
## 68   5 31.90 7.600e+01
## 69   5 37.06 5.700e+01
## 70   5 42.19 4.300e+01
## 71   5 47.10 3.300e+01
## 72   5 51.62 2.500e+01
## 73   5 55.65 1.900e+01
## 74   5 59.14 1.400e+01
## 75   5 62.13 1.100e+01
## 76   5 64.65 8.000e+00
## 77   5 66.76 6.000e+00
## 78   5 68.53 5.000e+00
## 79   5 69.98 4.000e+00
## 80   5 71.17 3.000e+00
## 81   5 72.11 2.000e+00
## 82   5 72.83 2.000e+00
## 83   5 73.37 1.000e+00
## 84   5 73.76 1.000e+00
## 85   5 74.02 1.000e+00
## 86   5 74.20 0.000e+00
## 87   5 74.31 0.000e+00
## 88   5 74.39 0.000e+00
## 89   5 74.43 0.000e+00
## 90   5 74.46 0.000e+00
## 91   5 74.48 0.000e+00
```

```
## 92   5 74.49 0.000e+00
## 93   5 74.49 0.000e+00
## 94   5 74.49 0.000e+00
## 95   5 74.50 0.000e+00
## 96   5 74.50 0.000e+00
## 97   5 74.50 0.000e+00
## 98   5 74.50 0.000e+00
## 99   5 74.50 0.000e+00
## 100  5 74.50 0.000e+00
```

d) The best λ found via ridge regression is 3.10569, leading to an MSE near 100.

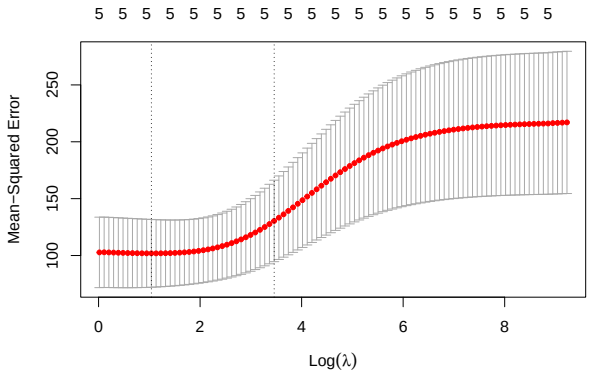
```
cv_out <- cv.glmnet(x_train, y_train, alpha = 0)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```
cv_out$lambda.min
```

```
## [1] 2.829789
```

```
plot(cv_out)
```



e) Using the best λ from cross-validation ridge model, we find that the MSE values are 45.04 and 44.19 for the OLS and ridge models, respectively.

```
# OLS prediction
pred_ols <- predict(ols_model, newdata = swiss[test_indices,])
mse_ols <- mean((y_test - pred_ols)^2)

# ridge prediction
optimal_lambda_ridge <- cv_out$lambda.min
pred_ridge <- predict(ridge_model, s = optimal_lambda_ridge, newx = as.matrix(x_test))
mse_ridge <- mean((y_test - pred_ridge)^2)

list(MSE_OLS = mse_ols, MSE_Ridge = mse_ridge)
```

```
## $MSE_OLS
## [1] 45.03527
##
## $MSE_Ridge
## [1] 43.87479
```

f) The best λ for the lasso model is 0.8248985, leading to a predicted MSE of 48.39 on the testing data.

```r
lasso_model <- glmnet(x_train, y_train, alpha=1, lambda=lambda)
print(lasso_model)
```

```
##
## Call:  glmnet(x = x_train, y = y_train, alpha = 1, lambda = lambda)
##
##     Df  %Dev    Lambda
## 1    0  0.00 1.000e+10
## 2    0  0.00 7.565e+09
## 3    0  0.00 5.722e+09
## 4    0  0.00 4.329e+09
## 5    0  0.00 3.275e+09
## 6    0  0.00 2.477e+09
## 7    0  0.00 1.874e+09
## 8    0  0.00 1.417e+09
## 9    0  0.00 1.072e+09
## 10   0  0.00 8.111e+08
## 11   0  0.00 6.136e+08
## 12   0  0.00 4.642e+08
## 13   0  0.00 3.511e+08
## 14   0  0.00 2.656e+08
## 15   0  0.00 2.009e+08
## 16   0  0.00 1.520e+08
## 17   0  0.00 1.150e+08
## 18   0  0.00 8.697e+07
## 19   0  0.00 6.579e+07
## 20   0  0.00 4.977e+07
## 21   0  0.00 3.765e+07
## 22   0  0.00 2.848e+07
## 23   0  0.00 2.154e+07
## 24   0  0.00 1.630e+07
## 25   0  0.00 1.233e+07
## 26   0  0.00 9.326e+06
## 27   0  0.00 7.055e+06
## 28   0  0.00 5.337e+06
## 29   0  0.00 4.037e+06
## 30   0  0.00 3.054e+06
## 31   0  0.00 2.310e+06
## 32   0  0.00 1.748e+06
## 33   0  0.00 1.322e+06
## 34   0  0.00 1.000e+06
## 35   0  0.00 7.565e+05
## 36   0  0.00 5.722e+05
## 37   0  0.00 4.329e+05
## 38   0  0.00 3.275e+05
```

```
## 39   0  0.00 2.477e+05
## 40   0  0.00 1.874e+05
## 41   0  0.00 1.417e+05
## 42   0  0.00 1.072e+05
## 43   0  0.00 8.111e+04
## 44   0  0.00 6.136e+04
## 45   0  0.00 4.642e+04
## 46   0  0.00 3.511e+04
## 47   0  0.00 2.656e+04
## 48   0  0.00 2.009e+04
## 49   0  0.00 1.520e+04
## 50   0  0.00 1.150e+04
## 51   0  0.00 8.697e+03
## 52   0  0.00 6.579e+03
## 53   0  0.00 4.977e+03
## 54   0  0.00 3.765e+03
## 55   0  0.00 2.848e+03
## 56   0  0.00 2.154e+03
## 57   0  0.00 1.630e+03
## 58   0  0.00 1.233e+03
## 59   0  0.00 9.330e+02
## 60   0  0.00 7.060e+02
## 61   0  0.00 5.340e+02
## 62   0  0.00 4.040e+02
## 63   0  0.00 3.050e+02
## 64   0  0.00 2.310e+02
## 65   0  0.00 1.750e+02
## 66   0  0.00 1.320e+02
## 67   0  0.00 1.000e+02
## 68   0  0.00 7.600e+01
## 69   0  0.00 5.700e+01
## 70   0  0.00 4.300e+01
## 71   0  0.00 3.300e+01
## 72   0  0.00 2.500e+01
## 73   0  0.00 1.900e+01
## 74   0  0.00 1.400e+01
## 75   0  0.00 1.100e+01
## 76   1 18.94 8.000e+00
## 77   1 33.11 6.000e+00
## 78   3 47.01 5.000e+00
## 79   3 56.50 4.000e+00
## 80   4 62.78 3.000e+00
## 81   4 66.81 2.000e+00
## 82   4 69.12 2.000e+00
## 83   4 70.44 1.000e+00
## 84   4 71.20 1.000e+00
## 85   4 71.63 1.000e+00
## 86   5 72.53 0.000e+00
## 87   5 73.37 0.000e+00
## 88   5 73.85 0.000e+00
## 89   5 74.13 0.000e+00
## 90   5 74.29 0.000e+00
## 91   5 74.38 0.000e+00
## 92   5 74.43 0.000e+00
```

```
## 93   5 74.46 0.000e+00
## 94   5 74.48 0.000e+00
## 95   5 74.49 0.000e+00
## 96   5 74.49 0.000e+00
## 97   5 74.49 0.000e+00
## 98   5 74.50 0.000e+00
## 99   5 74.50 0.000e+00
## 100  5 74.50 0.000e+00
```

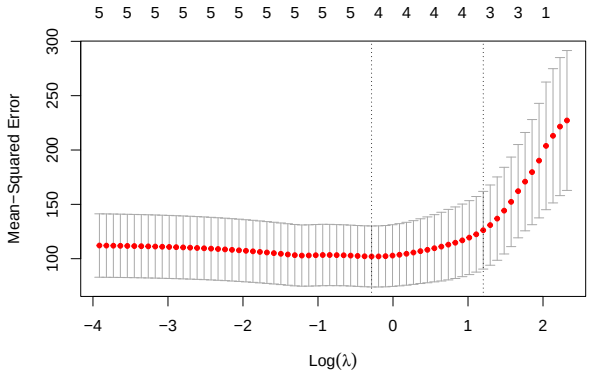
```
cv_out_lasso <- cv.glmnet(x_train, y_train, alpha = 1)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```
print(cv_out_lasso$lambda.min)
```

```
## [1] 0.7516168
```

```
plot(cv_out_lasso)
```




```
optimal_lambda_lasso <- cv_out_lasso$lambda.min
pred_lasso <- predict(lasso_model, s = optimal_lambda_lasso, newx = as.matrix(x_test))
mse_lasso <- mean((y_test - pred_lasso)^2)
list(MSE_OLS = mse_ols, MSE_Ridge = mse_ridge, MSE_Lasso = mse_lasso)
```

```
## $MSE_OLS
## [1] 45.03527
##
## $MSE_Ridge
## [1] 43.87479
##
## $MSE_Lasso
## [1] 48.38982
```

```
# OLS coefficients
summary(ols_model)
```

```
##
## Call:
## lm(formula = Fertility ~ ., data = swiss, subset = train_indices)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.6123  -5.8985   0.3505   3.7291  15.6139
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      57.61010   17.56840   3.279  0.00442 **
## Agriculture      -0.15748    0.12745  -1.236  0.23341
## Examination      -0.23538    0.45798  -0.514  0.61391
## Education        -0.92411    0.26249  -3.521  0.00262 **
## Catholic          0.10465    0.06519   1.605  0.12686
## Infant.Mortality  1.55636    0.57091   2.726  0.01437 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.279 on 17 degrees of freedom
## Multiple R-squared:  0.745,  Adjusted R-squared:   0.67
## F-statistic: 9.933 on 5 and 17 DF,  p-value: 0.0001402
```

```
# ridge coefficients
print(coef(cv_out, s = optimal_lambda_ridge))
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                             s1
## (Intercept)      52.92075512
## Agriculture      -0.02985240
## Examination      -0.32233465
## Education        -0.57957721
## Catholic          0.05698707
## Infant.Mortality  1.39813010
```

```
# lasso coefficients
print(coef(cv_out_lasso, s = optimal_lambda_lasso))
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                             s1
## (Intercept)      47.16969026
## Agriculture       .
## Examination      -0.03456093
## Education        -0.73010648
## Catholic          0.06915655
## Infant.Mortality  1.43087831
```

g) We find that the model with the best predicted MSE on the testing data is the ridge model MSE = 44.1905. The optimal constraint parameter ridge regression was found through cross-validation to be $\lambda_R = 2.829789$, lower than the optimal lambda found through cross validation for lasso, $\lambda_L = 0.8248985$. We find that the OLS coefficients are all within (-2, 2), except for the intercept of 57.61010. For ridge, the coefficients are also all within (-2, 2), except for the intercept of 52.83218. For lasso,

the coefficients are all within (-2, 2), where notably the coefficient for Agriculture has been shrunk to 0, meaning it that data is no longer used in the model. This is unique to lasso and not possible in ridge, the counter equivalent in OLS being an insignificant t-test, even though the causes are different. Since ridge uses the L2 norm, it can shrink coefficients close to 0, but never actually to be equal to 0. Nonetheless, we find that the coefficients between the three models are generally quite similar and lead to similar predicted MSE values on the test set.