STA 5207: Homework 11

Due: Friday, April 19 by 11:59 PM

Include your R code in an R chunks as part of your answer. In addition, your written answer to each exercise should be self-contained so that the grader can determine your solution without reading your code or deciphering its output.

```
library(ISLR2)
library(lmridge)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

Exercise 1 (Boston Housing) [50 points]

For this exercise, we will analyze a data set containing housing values in 506 suburbs of Boston. The data set was split into a training and testing data set. Note that this data set is a version of the `Boston` data set from the `ISLR2` package, so you can type `?ISLR2::Boston` in `R` to read about the data set and the meaning of the variables. The training data set contains 354 suburbs and 13 variables. In the following exercises, use `log(medv)` (the logarithm of the median value of owner-occupied homes in $1000s) as the response and the other variables as predictors. You should use the `boston_train.csv` data set unless otherwise specified.

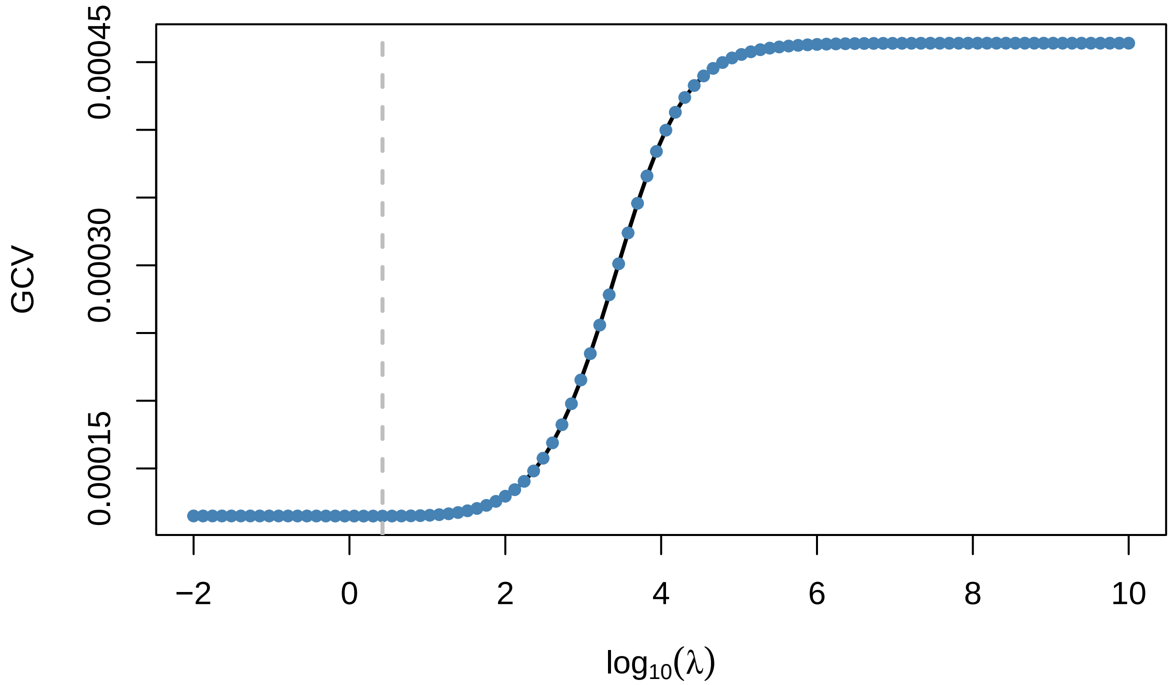
```
data = read.csv("boston_train.csv")
```

1. (10 points) Perform ridge regression with `log(medv)` as the response and the other variables as predictors using the data in `boston_train.csv`. Choose an appropriate value of λ using GCV. Justify the range of λ values you searched over and report your final choice of λ. Include any necessary plots in your response.

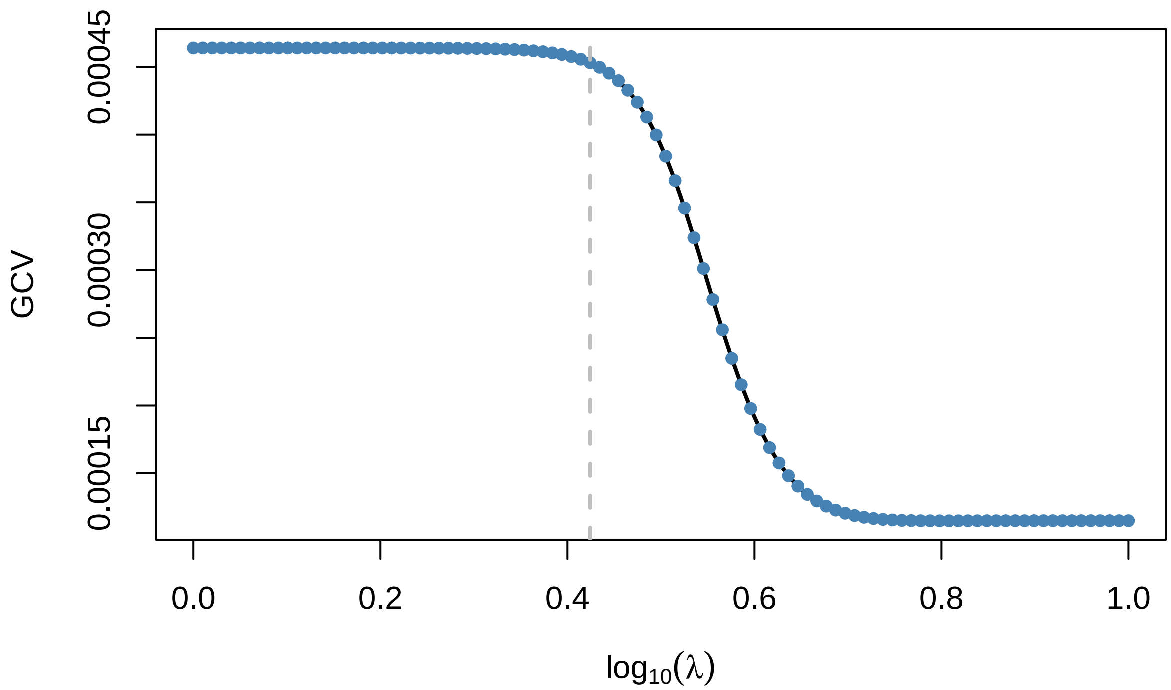
   ```
   grid = 10 ^ seq(10, -2 , length = 100)

   mod_ridge = lmridge(log(medv) ~ ., data = data,
                       scaling = 'scaled',
                       K = grid)

   k_est = kest(mod_ridge)

   print(k_est$kGCV)
   ```

   ```
   ## [1] 2.656088
   ```

   ```
   plot(log10(mod_ridge$K), k_est$GCV, type = 'l', lwd = 2,
        xlab = expression(log[10](lambda)), ylab = 'GCV')

   points(log10(mod_ridge$K), k_est$GCV,
          pch = 19, col = 'steelblue', cex = 0.75)

   abline(v=log10(k_est$kGCV), lty = 'dashed', col = 'grey',
          lwd = 2)
   ```




```r
grid = 10 ^ seq(0, 1 , length = 100)

mod_ridge = lmridge(log(medv) ~ ., data = data,
                    scaling = 'scaled',
                    K = grid)

plot(log10(mod_ridge$K), k_est$GCV, type = 'l', lwd = 2,
     xlab = expression(log[10](lambda)), ylab = 'GCV')

points(log10(mod_ridge$K), k_est$GCV,
       pch = 19, col = 'steelblue', cex = 0.75)

abline(v=log10(k_est$kGCV), lty = 'dashed', col = 'grey',
       lwd = 2)
```




```r
k_est = kest(mod_ridge)

print(k_est$kGCV)
```

```
## [1] 2.364489
```

$\lambda = 2.365$ occurs in the interior of the plot which verifies that this is an appropriate choice of λ. It correlates roughly to .4 on the x axis of the plot.

2. (6 points) Report the estimated regression equation and R^2 value for the ridge regression model you chose in Question 1.

```r
# best lambda value chosen by GCV
k_best = kest(mod_ridge)$kGCV

# re-fit the model using the best value of lambda according to GCV
mod_ridge_best = lmridge(log(medv) ~ ., data = data,
                    scaling = 'scaled',
                    K = k_best)
summary(mod_ridge_best)
```

```
##
## Call:
## lmridge.default(formula = log(medv) ~ ., data = data, K = k_best,
##     scaling = "scaled")
##
##
## Coefficients: for Ridge parameter K= 2.364489
##           Estimate Estimate (Sc) StdErr (Sc) t-value (Sc) Pr(>|t|)
## Intercept   4.4616       48.5762     11.9226       4.0743   0.0001 ***
## crim       -0.0113       -0.1044      0.0136      -7.6767   <2e-16 ***
## zn          0.0009        0.0209      0.0157       1.3273   0.1853
## indus       0.0006        0.0041      0.0193       0.2106   0.8333
## chas        0.0813        0.0192      0.0109       1.7723   0.0772 .
```

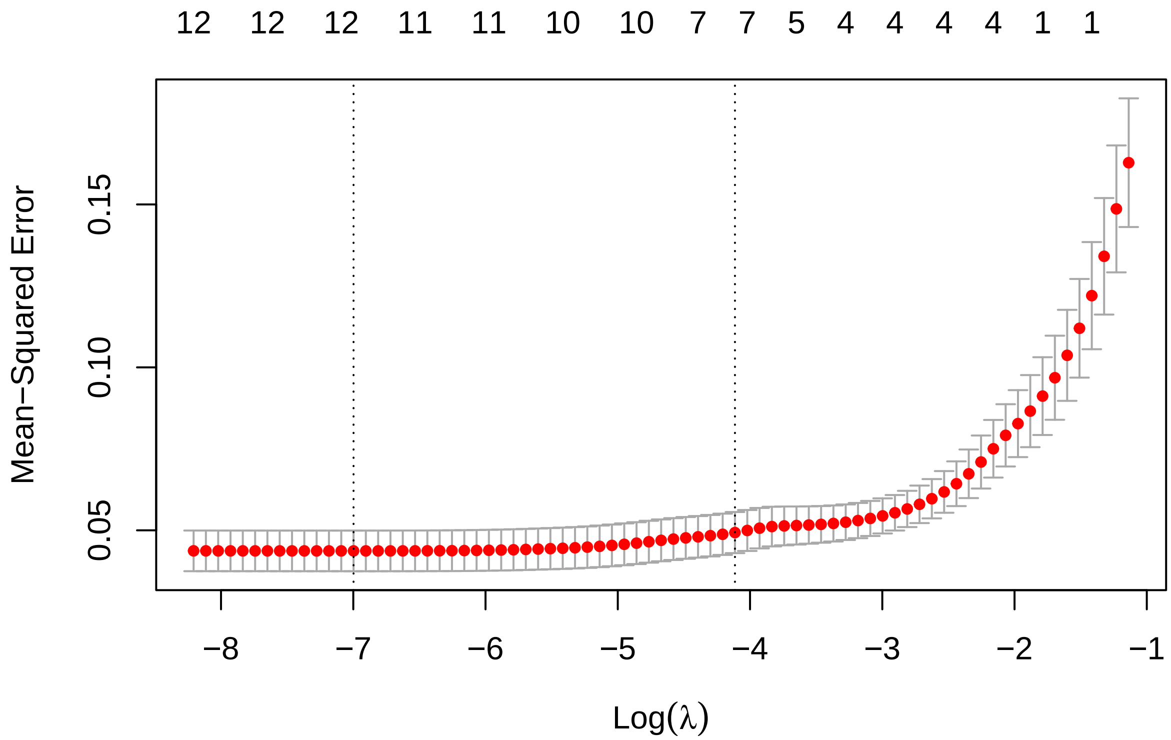
```
## nox        -0.8534       -0.0974      0.0210      -4.6459   <2e-16 ***
## rm          0.0818        0.0585      0.0140       4.1788   <2e-16 ***
## age        -0.0004       -0.0102      0.0178      -0.5768   0.5645
## dis        -0.0576       -0.1186      0.0208      -5.7024   <2e-16 ***
## rad         0.0154        0.1341      0.0265       5.0638   <2e-16 ***
## tax        -0.0006       -0.1021      0.0290      -3.5194   0.0005 ***
## ptratio    -0.0401       -0.0860      0.0138      -6.2502   <2e-16 ***
## lstat      -0.0285       -0.2032      0.0169     -12.0480   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Ridge Summary
##          R2      adj-R2    DF ridge           F         AIC         BIC
##     0.75920     0.75140    11.72634    94.68780 -1134.10936   988.99444
## Ridge minimum MSE= 0.004719227 at K= 2.364489
## P-value for F-test ( 11.72634 , 342.0138 ) = 9.039881e-100
## -------------------------------------------------------------------
```

$\log(\text{medv})_i = 4.5 - .011\text{crim}_i + .0009\text{zn}_i + .0006\text{indus}_i + .0813\text{chas}_i - .853\text{nox}_i + .0818\text{rm}_i - .0004\text{age}_i - .0576\text{dis}_i + .0154\text{rad}_i -$

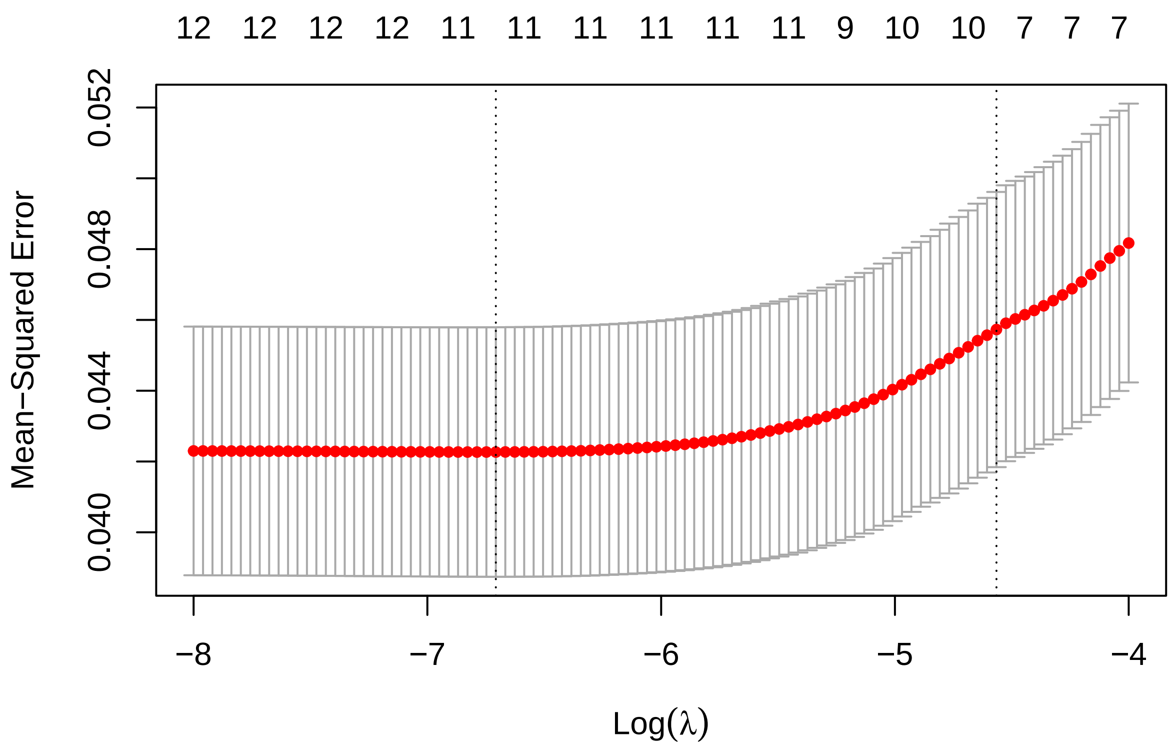
$R^2 = .75920$

3. (10 points) Perform lasso with `log(medv)` as the response and the other variables as predictors using the data in `boston_train.csv`. You should set a random seed of 42. Justify the range of λ values you searched over and report your chosen values of `lambda.min` and `lambda.1se`. Include any necessary plots in your response.

```r
set.seed(42)

x_train = model.matrix(log(medv) ~ ., data = data)[, -1]

y_train = log(data$medv)

mod_lasso = cv.glmnet(x_train, y_train)

plot(mod_lasso)
```




```r
grid = exp(seq(-8, -4, length=100))

mod_lasso = cv.glmnet(x_train, y_train, lambda = grid)

plot(mod_lasso)
```



```r
# lambda.min
mod_lasso$lambda.min
```

```
## [1] 0.001222239
```

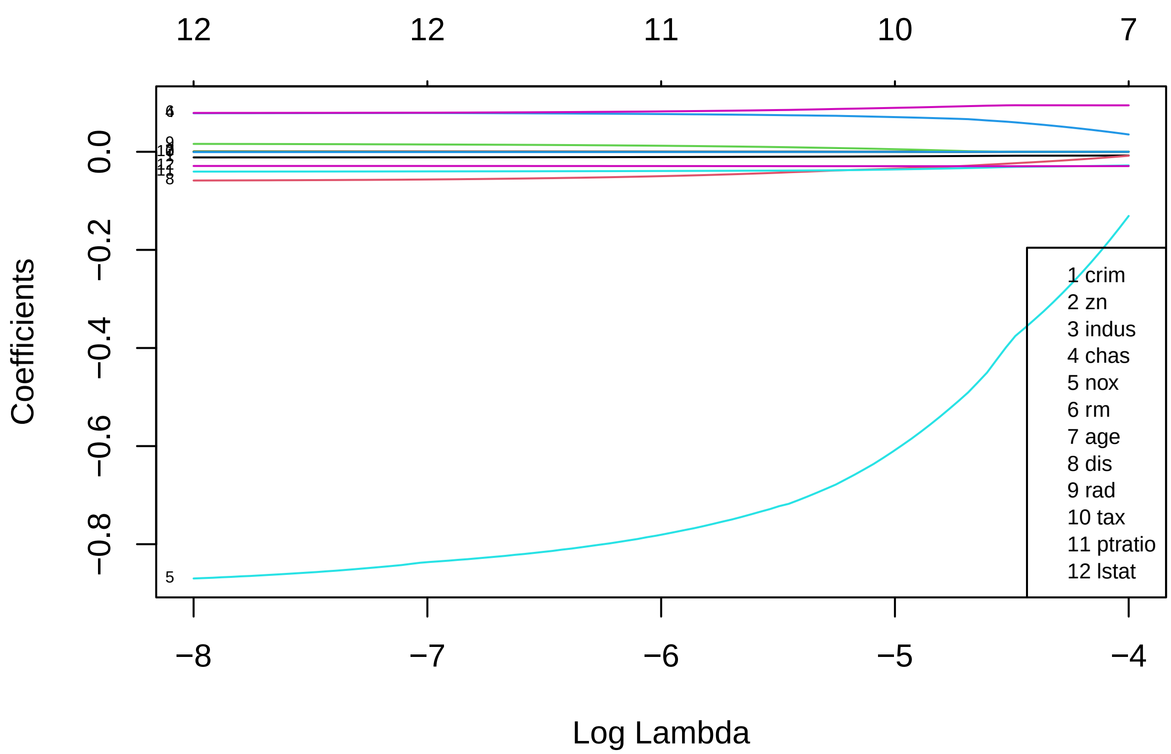
```r
# lambda.1se
mod_lasso$lambda.1se
```

```
## [1] 0.01040305
```

The range we chose is adequate as both lambda.min and lambda.1se occur in the interior of the plot. Specifically, lambda.min = .00122 and lambda.1se = .0104

4. (4 points) Report the plot of the solution path for the lasso coefficient estimates.

```r
# solution path plot
plot(mod_lasso$glmnet.fit, xvar = 'lambda', label = TRUE)

# include a legend of the variable names
pred_names = colnames(x_train)
legend('bottomright',
       legend = paste(1:length(pred_names), pred_names),
       cex= 0.7)
```



As λ increases, the number of zero coefficients increases

We also see that the coefficient for nox decreases drastically as lambda increases.

5. (6 points) Report the number of variables with non-zero coefficients and the estimated regression equation for the lasso model estimated using `lambda.min`.

```r
coef(mod_lasso, s = 'lambda.min')
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
```

```
## (Intercept)  4.4343784989
## crim        -0.0111125334
## zn           0.0007867701
## indus        .
## chas         0.0787073747
## nox         -0.8255069990
## rm           0.0802287493
## age         -0.0002427218
## dis         -0.0551138800
## rad          0.0144290732
## tax         -0.0005514806
## ptratio     -0.0397943568
## lstat       -0.0289399241
```

There are twelve non-zero variables.

$$\log(\text{medv})_i = 4.4 - .011\text{crim}_i + .0008\text{zn}_i + .0787\text{chas}_i - .826\text{nox}_i + .0802\text{rm}_i - .0002\text{age}_i - .0551\text{dis}_i + .0144\text{rad}_i - .0005\text{tax}_i - .03$$

6. (6 points) Report the number of variables with non-zero coefficients and the estimated regression equation for the lasso model estimated using `lambda.1se`.

```r
coef(mod_lasso, s = 'lambda.1se')
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  3.7560230080
## crim        -0.0082070849
## zn           .
## indus        .
## chas         0.0627192232
## nox         -0.4243620539
## rm           0.0943467740
## age          .
## dis         -0.0252676366
## rad          0.0004698932
## tax          .
## ptratio     -0.0316644673
## lstat       -0.0293826754
```

There are 9 non-zero variables.

$$\log(\text{medv})_i = 3.756 - .0082\text{crim}_i + .0627\text{chas}_i - .424\text{nox}_i + .0943\text{rm}_i - .0253\text{dis}_i + .0004\text{rad}_i - .0317\text{ptratio}_i - .0294\text{lstat}_i$$

7. (8 points) The file `boston_test.csv` on Canvas contains a new test data set of 152 houses not found in `boston_train.csv`. Calculate the RMSE values on this test data (`boston_test.csv`) for the following four models:

- **Model 1**: The ridge regression model you chose in Question 1,
- **Model 2**: The lasso model using `lambda.min` you reported in Question 5.
- **Model 3**: The lasso model using `lambda.1se` you reported in Question 6.
- **Model 4**: An OLS regression model estimated with `log(medv)` as the response and the other variables as predictors.

The RMSE should be calculated using the logarithm of the response, that is,

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\log(\texttt{medv}_i) - \hat{y}_i\right]^2}.$$

Based on these test RMSE values, which model do you prefer?

```
test_data = read.csv("boston_test.csv")
# quick function to calculate RMSE
rmse = function(y_true, y_pred) {
    sqrt(mean((y_true - y_pred)^2))
}

x_test = model.matrix(log(medv) ~ ., data = test_data)[, -1]
y_test = log(test_data$medv)

# Model 1: ridge

# predict on the new test data. This is the same as lm
y_pred = predict(mod_ridge_best, newdata = test_data)

# calculate the RMSE
rmse(y_test, y_pred)
```

```
## [1] 0.1841896
```

```
# Model 2: lasso with lambda.min

# predictions using lambda.min
y_pred = predict(mod_lasso, newx = x_test, s = 'lambda.min')

# calculate the RMSE
rmse(y_test, y_pred)
```

```
## [1] 0.1830979
```

```
# Model 3: lasso with lambda.1se

# predictions using lambda.1se
y_pred = predict(mod_lasso, newx = x_test, s = 'lambda.1se')

# calculate the RMSE
rmse(y_test, y_pred)
```

```
## [1] 0.1842544
```

```
# Model 4: OLS

# fit OLS model on the training data
mod_ols = lm(log(medv) ~ ., data = data)

# OLS predictions on the test data
y_pred = predict(mod_ols, test_data)

# calculate the RMSE
rmse(y_test, y_pred)
```

```
## [1] 0.1845913
```

I would choose Model 2: lasso with lambda.min as it has the lowest RMSE of the four models.

Exercise 2 (The `college` Data Set) [50 points]

This exercise will analyze statistics for a number of U.S. Colleges from the 1995 issue of the *US News and World Report*. The data set was split into a training and testing data set. The training data set can be found in `college_train.csv` on Canvas. Note that this data set is a version of the `College` data set from the `ISLR2` package. The training data set contains 388 universities and the following 18 variables:

- `Private`: A binary variable with 0 and 1 indicating a public or private university.
- `Apps`: Number of applications received.
- `Accept`: Number of applications accepted.
- `Enroll`: Number of new students enrolled.
- `Top10perc`: Percentage of new students who ranked in the top 10% of their high-school class.
- `Top25perc`: Percentage of new students who ranked in the top 25% of their high-school class.
- `F.Undergrad`: Number of fulltime undergraduates.
- `P.Undergrad`: Number of parttime undergraduates.
- `Room.Board`: Room and board costs.
- `Books`: Estimated book costs.
- `Personal`: Estimated personal spending.
- `PhD`: Pct. of faculty with Ph.D.'s.
- `Terminal`: Pct. of faculty with terminal degree.
- `S.F.Ratio`: Student/faculty ratio.
- `perc.alumni`: Pct. alumni who donate.
- `Expend`: Instructional expenditure per student.
- `Grad.Rate`: Graduation rate.

In the following exercise, we will use `Apps` as the response and the remaining variables as predictors. You should use the `college_train.csv` data set unless otherwise specified.

Important: The first column of `college_train.csv` and `college_test.csv` contains the row names, i.e., the college names. To properly load this data set into `R` using `read.csv`, you must set the argument `row.names = 1`, i.e., `college_train = read.csv('college_train.csv', row.names = 1)`.

```
data = college_train = read.csv('college_train.csv', row.names = 1)
```

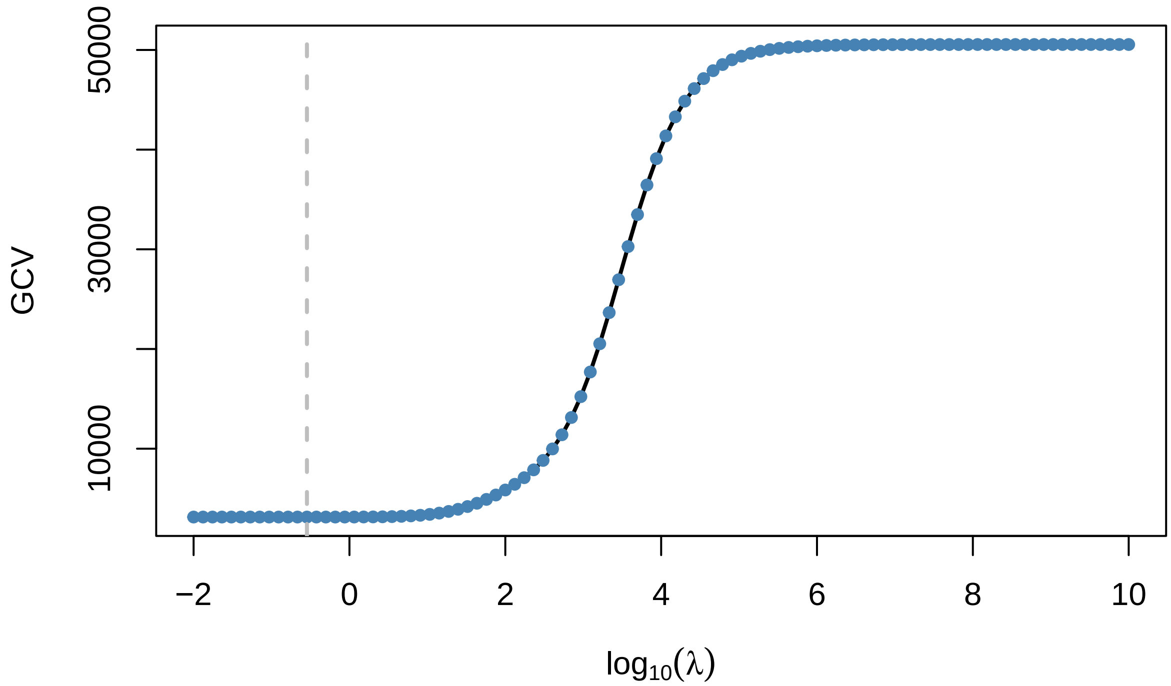
1. (10 points) Perform ridge regression with `Apps` as the response and the other variables as predictors using the data in `college_train.csv`. Choose an appropriate value of λ using GCV. Justify the range of λ values you searched over and report your final choice of λ. Include any necessary plots in your response.

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

mod_ridge = lmridge(Apps ~ ., data = data,
                    scaling = 'scaled',
                    K = grid)

k_est = kest(mod_ridge)

print(k_est$kGCV)
```

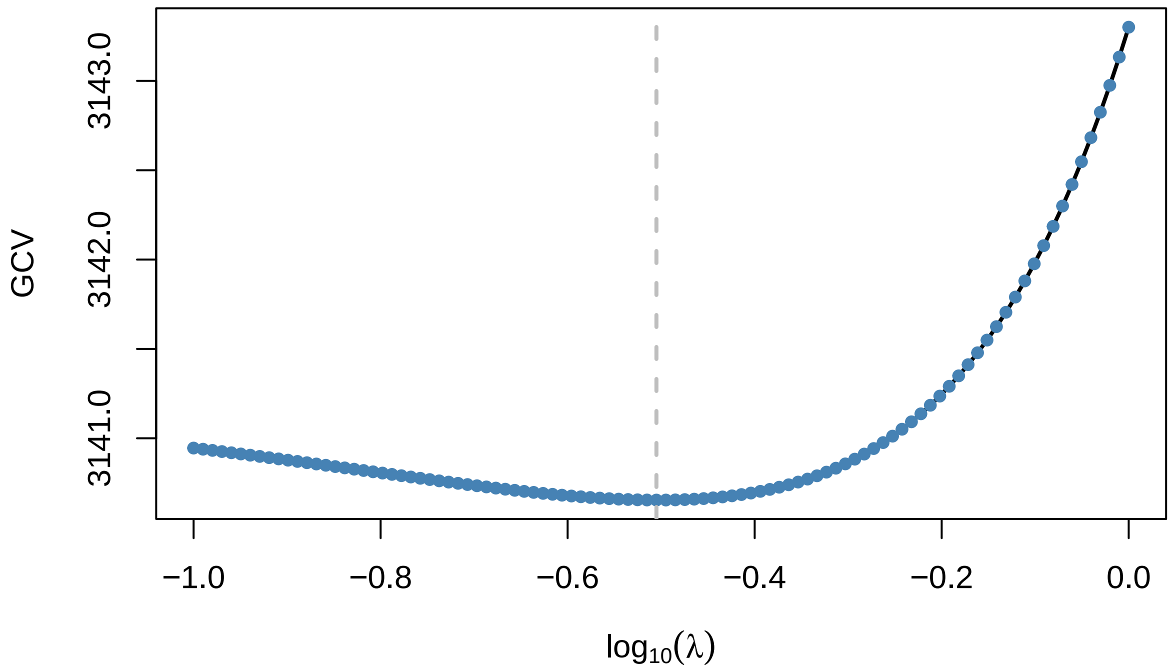
```
## [1] 0.2848036
```

```
plot(log10(mod_ridge$K), k_est$GCV, type = 'l', lwd = 2,
     xlab = expression(log[10](lambda)), ylab = 'GCV')
```

```
points(log10(mod_ridge$K), k_est$GCV,
       pch = 19, col = 'steelblue', cex = 0.75)

abline(v=log10(k_est$kGCV), lty = 'dashed', col = 'grey',
       lwd = 2)
```




```
# lambda values evenly spaced on the log-scale from 10^1 to 10^2.5.
grid = 10 ^ seq(-1, 0 , length = 100)

mod_ridge = lmridge(Apps ~ ., data = data,
                    scaling = 'scaled',
                    K = grid)

# extract the GCV errors and lambda that minimizes the GCV error
k_est = kest(mod_ridge)

# a plot of GCV vs. log10(lambda)
plot(log10(mod_ridge$K), k_est$GCV, type = 'l', lwd = 2,
     xlab = expression(log[10](lambda)), ylab = 'GCV')

points(log10(mod_ridge$K), k_est$GCV,
       pch = 19, col = 'steelblue', cex = 0.75)

# horizontal line at log10(kGCV), i.e.,
# the base 10 logarithm of the best lambda value
abline(v=log10(k_est$kGCV), lty = 'dashed', col = 'grey',
       lwd = 2)
```




```r
print(k_est$kGCV)
```

```
## [1] 0.3125716
```

$\lambda = 0.313$ occurs in the interior of the plot which verifies that this is an appropriate choice of λ. It correlates roughly to -.5 on the x axis of the plot.

2. (6 points) Report the R^2 value for the ridge regression model you chose in Question 1.

```r
# best lambda value chosen by GCV
k_best = kest(mod_ridge)$kGCV

# re-fit the model using the best value of lambda according to GCV
mod_ridge_best = lmridge(Apps ~ ., data = data,
                    scaling = 'scaled',
                    K = k_best)

summary(mod_ridge_best)
```

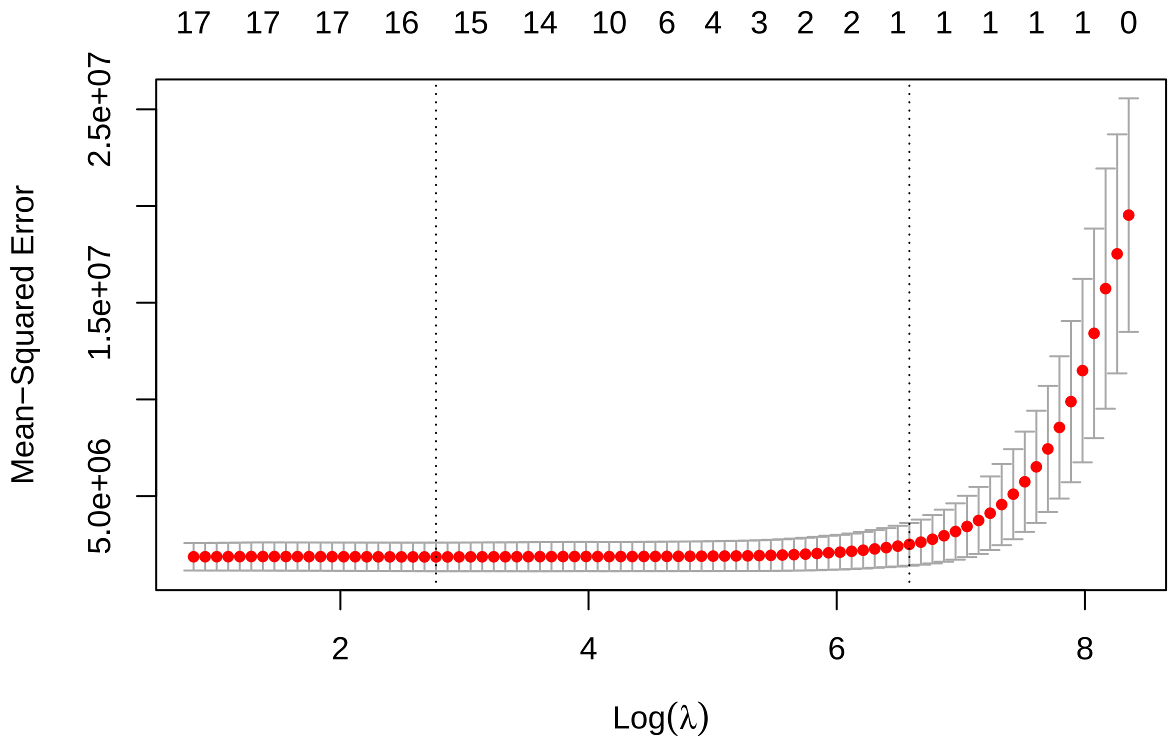
```
##
## Call:
## lmridge.default(formula = Apps ~ ., data = data, K = k_best,
##     scaling = "scaled")
##
##
## Coefficients: for Ridge parameter K= 0.3125716
##                Estimate Estimate (Sc) StdErr (Sc) t-value (Sc) Pr(>|t|)
## Intercept   -9.9656e+02   -1.3768e+07  1.8934e+06      -7.2718   <2e-16 ***
## Private     -4.3628e+02   -1.9579e+02  8.9562e+01      -2.1861   0.0294 *
## Accept       1.6291e+00    4.6477e+03  1.4139e+02      32.8713   <2e-16 ***
## Enroll      -1.4551e+00   -1.4980e+03  2.5715e+02      -5.8253   <2e-16 ***
## Top10perc    3.8153e+01    6.4657e+02  1.4056e+02       4.5999   <2e-16 ***
## Top25perc   -8.8390e+00   -1.7240e+02  1.3162e+02      -1.3098   0.1911
## F.Undergrad  1.4940e-01    8.0049e+02  2.4057e+02       3.3275   0.0010 ***
## P.Undergrad  5.2400e-02    9.5859e+01  7.0265e+01       1.3643   0.1733
```

```
## Outstate    -8.5400e-02   -3.4451e+02  1.1273e+02      -3.0560   0.0024 **
## Room.Board   1.5350e-01    1.6838e+02  7.5752e+01       2.2228   0.0268 *
## Books        1.4320e-01    1.9003e+01  5.8810e+01       0.3231   0.7468
## Personal     2.1600e-02    1.6214e+01  6.3047e+01       0.2572   0.7972
## PhD         -9.9280e+00   -1.5974e+02  1.1997e+02      -1.3315   0.1838
## Terminal    -3.4621e+00   -5.0010e+01  1.1791e+02      -0.4241   0.6717
## S.F.Ratio    4.6187e+01    1.7461e+02  8.0539e+01       2.1681   0.0308 *
## perc.alumni  3.6823e+00    4.5105e+01  7.1784e+01       0.6283   0.5302
## Expend       9.7400e-02    4.5297e+02  1.0187e+02       4.4467   <2e-16 ***
## Grad.Rate    6.4064e+00    1.0952e+02  7.2542e+01       1.5097   0.1320
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Ridge Summary
##         R2     adj-R2   DF ridge          F        AIC        BIC
##    0.94110    0.93850   16.92485  362.18692 5436.35785 7816.26734
## Ridge minimum MSE= 281754 at K= 0.3125716
## P-value for F-test ( 16.92485 , 371.0013 ) = 1.019901e-218
## -------------------------------------------------------------------
```

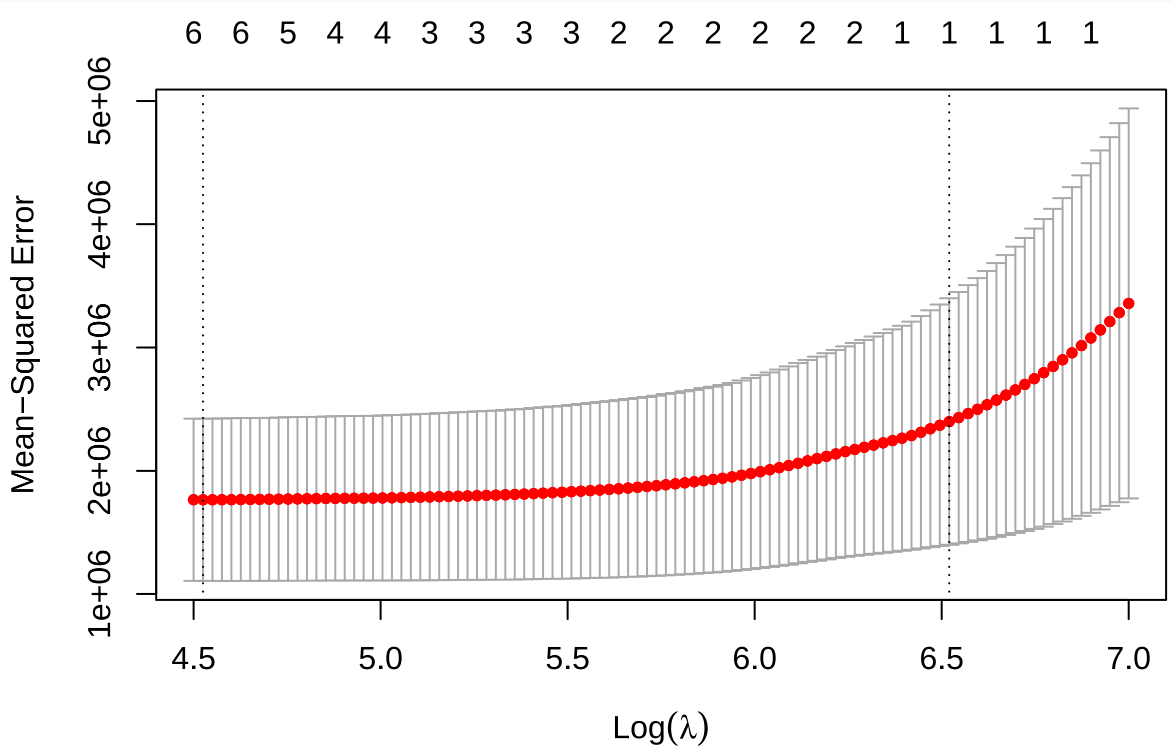
R^2 for the ridge regression model is .941

3. (10 points) Perform lasso with `Apps` as the response and the other variables as predictors using the data in `college_train.csv`. You should set a random seed of 42. Justify the range of λ values you searched over and report your chosen values of `lambda.min` and `lambda.1se`. Include any necessary plots in your response.

```r
set.seed(42)

x_train = model.matrix(Apps ~ ., data = data)[, -1]

y_train = data$Apps

mod_lasso = cv.glmnet(x_train, y_train)

plot(mod_lasso)
```




```r
grid = exp(seq(4.5, 7, length=100))

mod_lasso = cv.glmnet(x_train, y_train, lambda = grid)

plot(mod_lasso)
```



```r
# lambda.min
mod_lasso$lambda.min
```

```
## [1] 92.31924
```

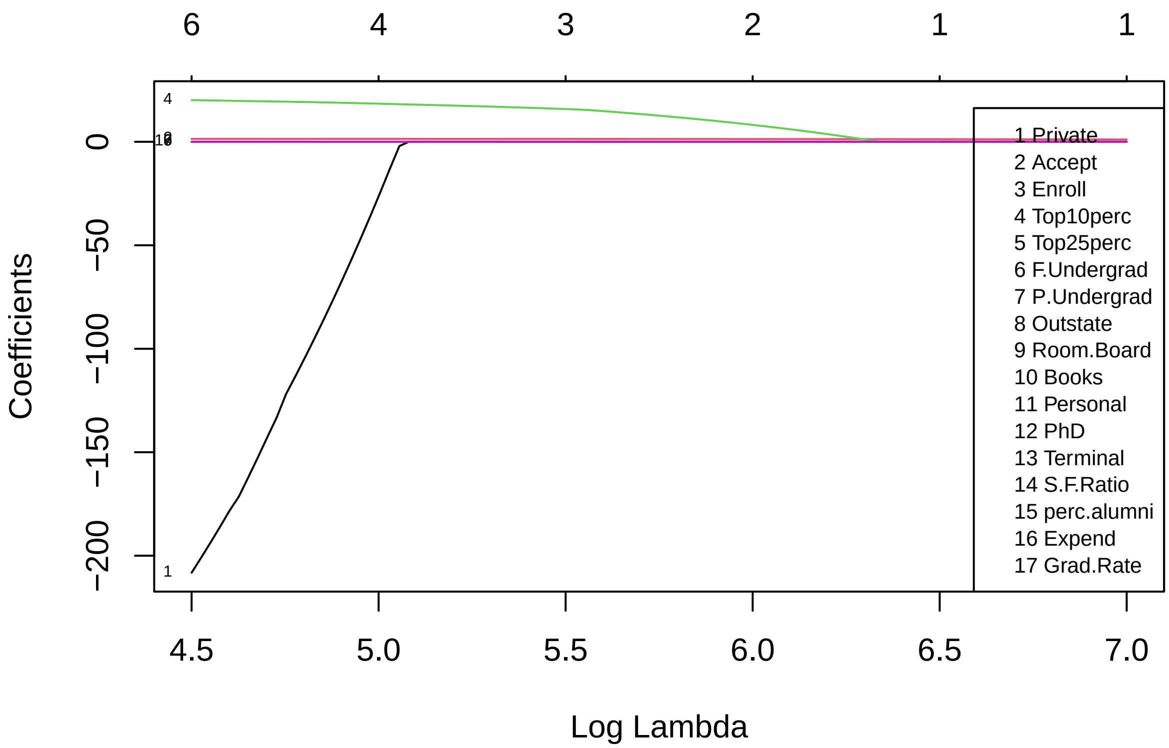
```r
# lambda.1se
mod_lasso$lambda.1se
```

```
## [1] 678.7155
```

The range we chose is adequate as both lambda.min and lambda.1se occur in the interior of the plot. Specifically, lambda.min = 92.32 and lambda.1se = 678.7

4. (4 points) Report the plot of the solution path for the lasso coefficient estimates.

```r
# solution path plot
plot(mod_lasso$glmnet.fit, xvar = 'lambda', label = TRUE)

# include a legend of the variable names
pred_names = colnames(x_train)
legend('bottomright',
       legend = paste(1:length(pred_names), pred_names),
       cex= 0.7)
```



As λ increases, the number of zero coefficients increases

We also see that the coefficient for Private decreases drastically as lambda increases.

5. (6 points) Report the number of variables with non-zero coefficients for the lasso model estimated using `lambda.min`.

```r
coef(mod_lasso, s = 'lambda.min')
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
```

```
## (Intercept) -6.291008e+02
## Private     -2.010337e+02
## Accept       1.412840e+00
## Enroll       .
## Top10perc    2.006315e+01
## Top25perc    .
## F.Undergrad  .
## P.Undergrad  3.625541e-03
## Outstate     .
## Room.Board   2.224068e-02
## Books        .
## Personal     .
## PhD          .
## Terminal     .
## S.F.Ratio    .
## perc.alumni  .
## Expend       2.550498e-02
## Grad.Rate    .
```

There are six non-zero coefficients of the lasso model with lambda.min.

6. (6 points) Report the number of variables with non-zero coefficients **and** the estimated regression equation for the lasso model estimated using `lambda.1se`.

```r
coef(mod_lasso, s = 'lambda.1se')
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept) 492.551898
## Private       .
## Accept        1.250894
## Enroll        .
## Top10perc     .
## Top25perc     .
## F.Undergrad   .
## P.Undergrad   .
## Outstate      .
## Room.Board    .
## Books         .
## Personal      .
## PhD           .
## Terminal      .
## S.F.Ratio     .
## perc.alumni   .
## Expend        .
## Grad.Rate     .
```

There is only one non-zero coefficient in the lasso model with lambda.1se.

$$\text{Apps}_i = 492.6 + 1.25\text{Accept}_i$$

1. (8 points) The file `college_test.csv` on Canvas contains a new test data set of 389 colleges not found in `college_train.csv`. Calculate the RMSE values on this test data (`college_test.csv`) for the following models:

- **Model 1**: The ridge regression model you chose in Question 1,
- **Model 2**: The lasso model using `lambda.min` you reported in Question 5.

- **Model 3**: The lasso model using `lambda.1se` you reported in Question 6.
- **Model 4**: An OLS regression model estimated with `Apps` as the response and the other variables as predictors.

Based on these test RMSE values, which model do you prefer?

```
test_data = read.csv('college_test.csv', row.names = 1)

x_test = model.matrix(Apps ~ ., data = test_data)[, -1]
y_test = test_data$Apps

# Model 1: ridge

# predict on the new test data. This is the same as lm
y_pred = predict(mod_ridge_best, newdata = test_data)

# calculate the RMSE
rmse(y_test, y_pred)
```

```
## [1] 1042.985
```

```
# Model 2: lasso with lambda.min

# predictions using lambda.min
y_pred = predict(mod_lasso, newx = x_test, s = 'lambda.min')

# calculate the RMSE
rmse(y_test, y_pred)
```

```
## [1] 1069.545
```

```
# Model 3: lasso with lambda.1se

# predictions using lambda.1se
y_pred = predict(mod_lasso, newx = x_test, s = 'lambda.1se')

# calculate the RMSE
rmse(y_test, y_pred)
```

```
## [1] 1378.69
```

```
# Model 4: OLS

# fit OLS model on the training data
mod_ols = lm(Apps ~ ., data = data)

# OLS predictions on the test data
y_pred = predict(mod_ols, test_data)

# calculate the RMSE
rmse(y_test, y_pred)
```

```
## [1] 1045.757
```

I would choose Model 1: ridge regression model as it has the lowest RMSE of the four models.