HW8

Minh Luc, Devin Pham, Kyle Moore

Friday of Week 8, 05/20/2022

Contents

We all contributed equally for this homework.

Question 0

Member 1:

- Name: Minh Luc
- Student ID: A17209607

Member 2:

- Name: Kyle Moore
- Student ID: A14271413

Member 3:

- Name: Devin Pham
- Student ID: A17198936

Question 1

- **(a)**

```
library(ISLR2)
data <- College

set.seed(2)

n <- nrow(data)

train_index <- sample(1:n, size = n / 2)

train <- data[train_index,]
test <- data[-train_index,]
```

- **(b)**

```
train_lm <- lm(Apps ~ ., data = train) # fit model to training set

train_error <- mean((train$Apps - predict(train_lm))^2); train_error # train error
```

```
## [1] 1114356
```

```
test_error <- mean((test$Apps - predict(train_lm, test))^2); test_error # test error
```

```
## [1] 1093608
```

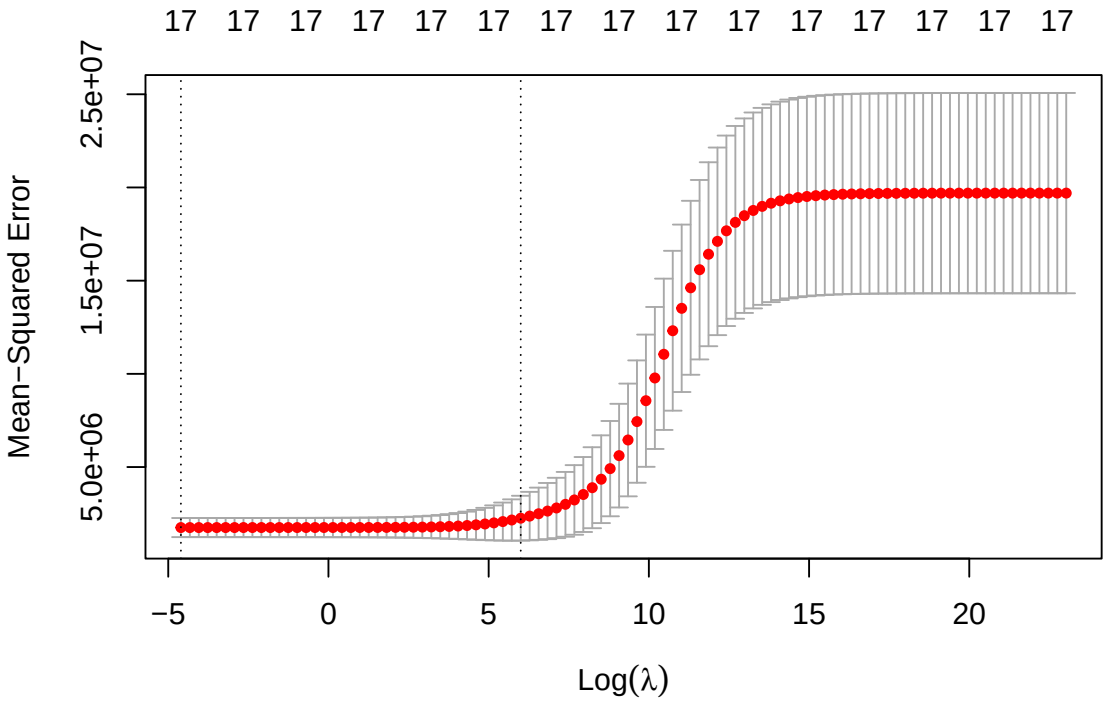
- **(c)**

```
library(glmnet)
```

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

```
## Loaded glmnet 4.1-4
```

```
x_train <- model.matrix(Apps ~ ., data = train)
y_train <- train$Apps
x_test <- model.matrix(Apps ~ ., data = test)
y_test <- test$Apps

grid <- 10^seq(10,-2,length=100)
ridge.mod <- glmnet(x_train, y_train, alpha=0, lambda = grid)

cv.out <- cv.glmnet(x_train, y_train, alpha=0, lambda = grid)
plot(cv.out)
```




```
best_lam <- cv.out$lambda.min; best_lam # best lambda from cross validation
```

```
## [1] 0.01
```

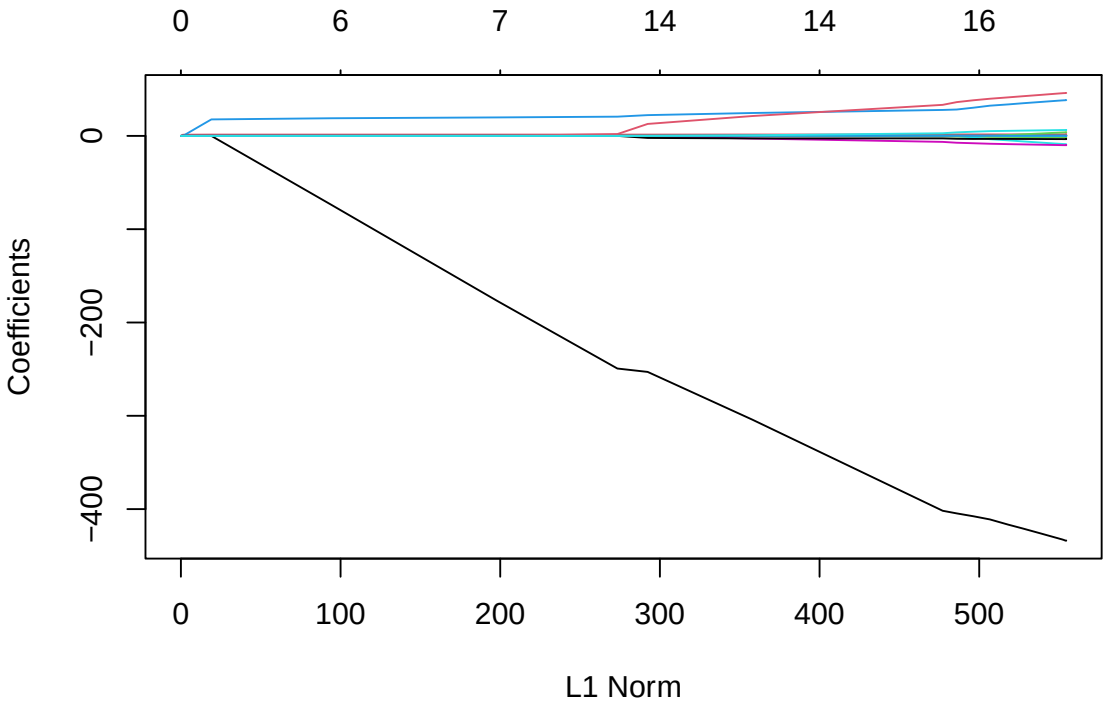
```
ridge.pred <- predict(ridge.mod, s = best_lam, newx = x_test)
mean((ridge.pred - y_test)^2) # test error for ridge regression
```

```
## [1] 1092971
```

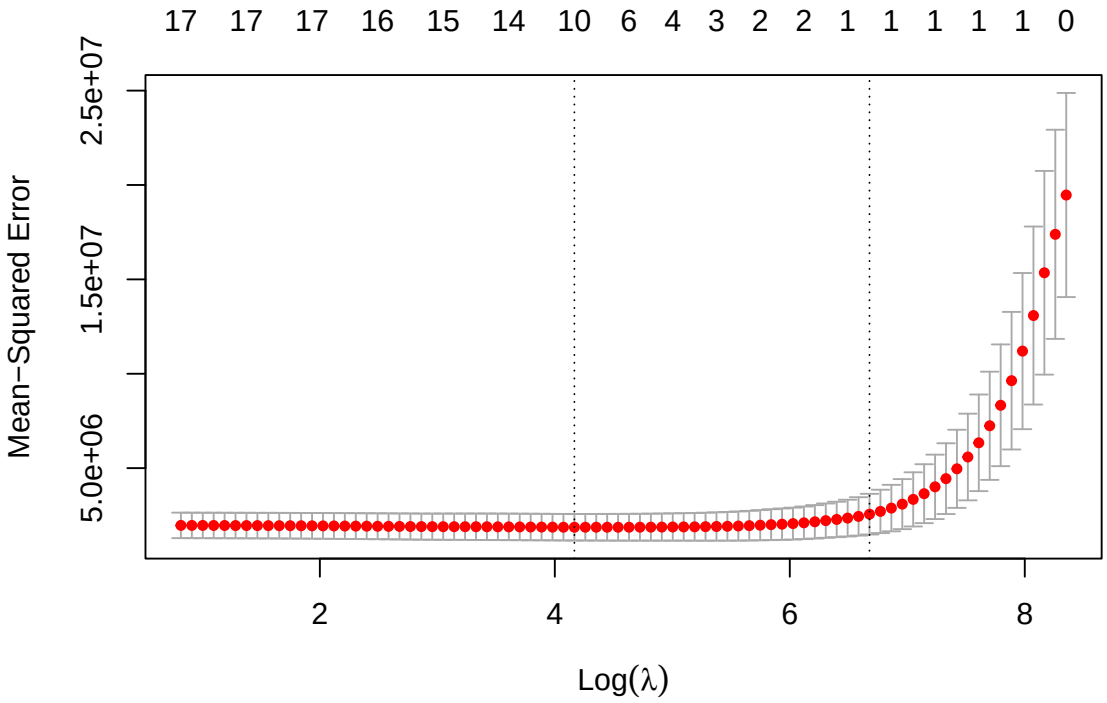
- (d)

```
lasso_mod <- glmnet(x_train,y_train,alpha=1,lambda=grid)
plot(lasso_mod)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```




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




```
best_lam <- cv.out$lambda.min # best lambda from cross validation

lasso_pred <- predict(lasso_mod,s=best_lam,newx = x_test)
mean((lasso_pred - y_test)^2) # test error for lasso
```

```
## [1] 1100343
```

```
lasso.coef = predict(lasso_mod,type="coefficients",s = best_lam)[1:19,]
lasso.coef
```

```
##   (Intercept)   (Intercept)    PrivateYes        Accept        Enroll
## -8.197697e+02  0.000000e+00 -2.515325e+02  1.410955e+00  0.000000e+00
##     Top10perc     Top25perc   F.Undergrad   P.Undergrad      Outstate
##  2.160644e+01  0.000000e+00  0.000000e+00  1.065627e-02 -1.116217e-02
##    Room.Board         Books      Personal           PhD      Terminal
##  6.376726e-02  0.000000e+00  2.935991e-03 -3.146104e-01 -1.336449e+00
##     S.F.Ratio   perc.alumni        Expend     Grad.Rate
##  8.540110e+00  0.000000e+00  3.826007e-02  0.000000e+00
```

```
lasso.coef[lasso.coef!=0]
```

```
##   (Intercept)    PrivateYes        Accept     Top10perc   P.Undergrad
## -8.197697e+02 -2.515325e+02  1.410955e+00  2.160644e+01  1.065627e-02
##      Outstate    Room.Board      Personal           PhD      Terminal
## -1.116217e-02  6.376726e-02  2.935991e-03 -3.146104e-01 -1.336449e+00
##     S.F.Ratio        Expend
##  8.540110e+00  3.826007e-02
```

- **(e)**

```r
library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```r
pcr.fit <- pcr(Apps ~ ., data = train, scale = TRUE, validation = "CV")

summary(pcr.fit)
```

```
## Data:    X dimension: 388 17
##  Y dimension: 388 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            4440     4473     2382     2389     2055     1836     1835
## adjCV         4440     4473     2377     2385     2042     1824     1825
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1848     1835     1758      1764      1770      1776      1790
## adjCV     1836     1825     1751      1758      1764      1769      1784
##        14 comps  15 comps  16 comps  17 comps
## CV         1795      1738      1413      1310
## adjCV      1791      1698      1396      1295
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X     31.4816    57.40    64.84    70.54    75.80    80.23    84.04    87.64
## Apps   0.1398    72.45    72.50    80.45    84.74    84.77    85.06    85.16
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       90.87     93.29     95.33     96.98     98.05     98.75     99.37
## Apps    85.93     86.16     86.21     86.32     86.40     86.61     92.49
##       16 comps  17 comps
## X        99.85    100.00
## Apps     93.65     94.32
```

We choose $M = 17$ because that had the lowest estimated test error.

```r
pcr.pred <- predict(pcr.fit, test, ncomp = 17)
mean((test$Apps - pcr.pred)^2) # test error for PCR
```

```
## [1] 1093608
```

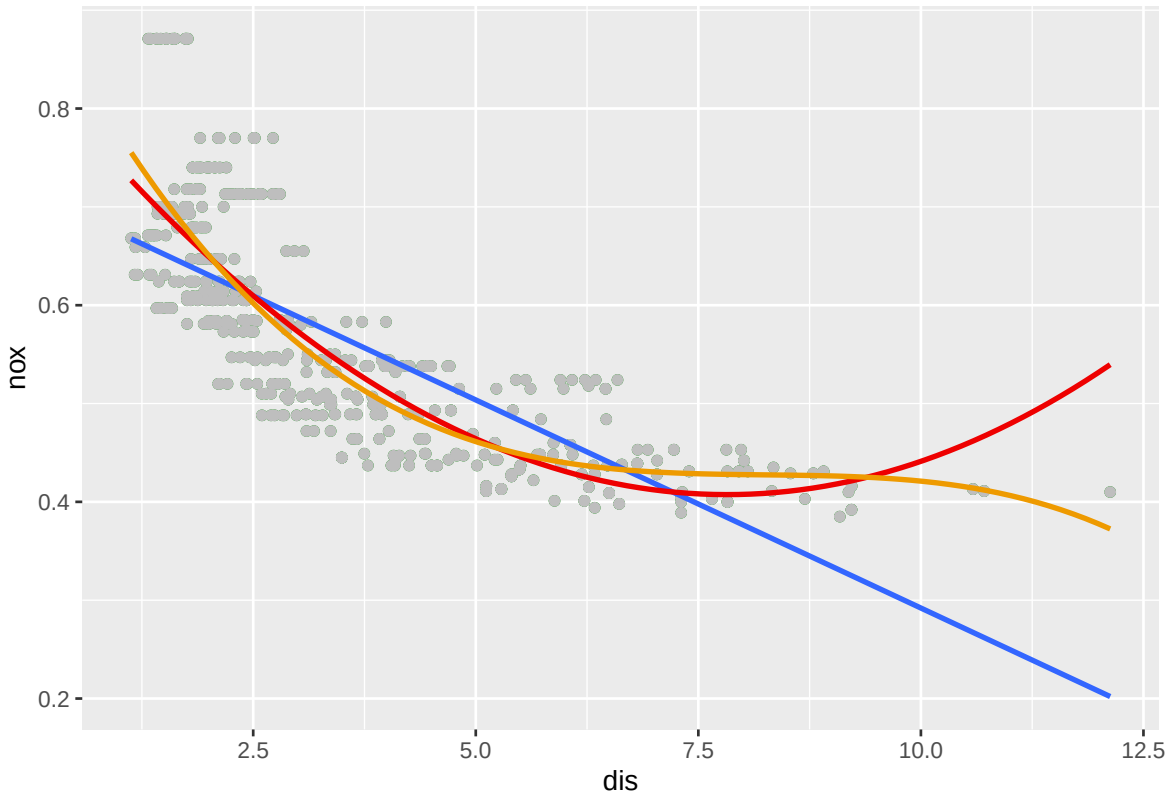
Question 2

- **(a)**

```
library(ISLR2)
library(ggplot2)

data2 <- Boston

lm2 <- lm(nox ~ poly(dis, 3), data = data2) # fit a polynomial with p = 3

summary(lm2)
```

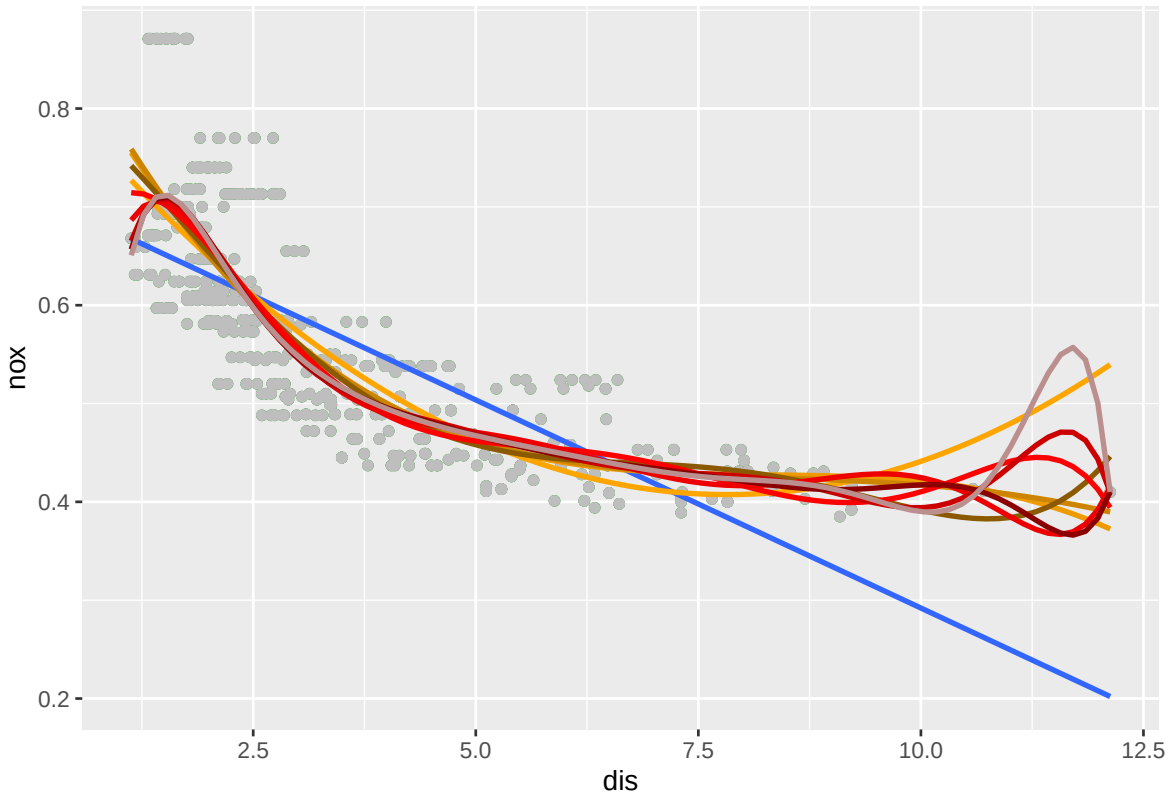
```
##
## Call:
## lm(formula = nox ~ poly(dis, 3), data = data2)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.121130 -0.040619 -0.009738  0.023385  0.194904
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
## poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
## poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
## poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06207 on 502 degrees of freedom
## Multiple R-squared:  0.7148, Adjusted R-squared:  0.7131
## F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
ggplot(data = data2, aes(x = dis, y = nox)) +
  geom_point(color = 'green4') +
  geom_point(data = data2, mapping = aes(x = dis, y = nox), color = "grey") +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), se = FALSE, color = "red2") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), se = FALSE, color = "orange2")
```



- **(b)**

```
ggplot(data = data2, aes(x = dis, y = nox)) +
  geom_point(color = 'green4') +
  geom_point(data = data2, mapping = aes(x = dis, y = nox), color = "grey") +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), se = FALSE, color = "orange1") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), se = FALSE, color = "orange2") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 4), se = FALSE, color = "orange3") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 5), se = FALSE, color = "orange4") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 6), se = FALSE, color = "red1") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 7), se = FALSE, color = "red2") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 8), se = FALSE, color = "red3") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 9), se = FALSE, color = "red4") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 10), se = FALSE, color = "rosybrown")
```




```
rss <- list()

for (i in 1:10) {
  lm_all <- lm(nox ~ poly(dis, i), data = data2)
  lm_pred <- predict(lm_all)
  rss[i] <- sum((data2$nox - lm_pred)^2) # RSS
}

print(rss)
```

```
## [[1]]
## [1] 2.768563
##
## [[2]]
## [1] 2.035262
##
## [[3]]
## [1] 1.934107
##
## [[4]]
## [1] 1.932981
##
## [[5]]
## [1] 1.91529
##
## [[6]]
## [1] 1.878257
##
```

```
## [[7]]
## [1] 1.849484
##
## [[8]]
## [1] 1.83563
##
## [[9]]
## [1] 1.833331
##
## [[10]]
## [1] 1.832171
```

- **(c)**

```
library(boot)

cv.error = rep(0, 10)
for (i in 1:10) {
  glm.fit = glm(nox ~ poly(dis, i), data = data2)
  cv.error[i] = cv.glm(data2, glm.fit)$delta[1]
}
cv.error # list of errors
```

```
##  [1] 0.005523868 0.004079449 0.003874762 0.003887521 0.004164865 0.005384278
##  [7] 0.011068782 0.008121397 0.017616356 0.004430276
```

```
which.min(cv.error) # minimum error in list
```

```
## [1] 3
```

The index of the minimum error is 3, with error of 0.003874762.
