A3Q3

(i)

```r
set.seed(250)
college <- read.csv("College.csv", header = T)
college <- college[-c(1)]
divide <- sample(nrow(college), 0.7*nrow(college))
train <- college[divide,]
test <- college[-divide,]
```

(ii)

```r
model1 <- lm(Apps~., data=train)
pred1 <- predict(model1, test)
pmse1 <- mean((pred1-test$Apps)^2)
pmse1
```

```
## [1] 876896.5
```

(iii)

```r
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.0.5
```

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

```
## Loaded glmnet 4.1-3
```

```r
cv1 <- cv.glmnet(data.matrix(train[-c(2)]),train$Apps, alpha=0)
best <- cv1$lambda.min
ridge <- glmnet(data.matrix(train[-c(2)]), train$Apps, lambda = best, alpha = 0)
pred2 <- predict(ridge, data.matrix(test[-c(2)]))
pmse2 <- mean((pred2-test$Apps)^2)
pmse2
```

```
## [1] 850188.9
```

(iv)

```
cv2 <- cv.glmnet(data.matrix(train[-c(2)]),train$Apps,alpha=1)
best1 <- cv2$lambda.min
lasso <- glmnet(data.matrix(train[-c(2)]), train$Apps, lambda = best1,
                alpha = 1)
pred3 <- predict(lasso, data.matrix(test[-c(2)]))
pmse3 <- mean((pred3-test$Apps)^2)
pmse3
```

```
## [1] 839186.6
```

(v)

```
w <- as.numeric(coef(cv1, s= cv1$lambda.min))[-1]
cv3 <- cv.glmnet(data.matrix(train[-c(2)]),train$Apps,alpha=1,
                 penalty.factor = 1/w)
best2 <- cv3$lambda.min
adp_lasso <- glmnet(data.matrix(train[-c(2)]),train$Apps, alpha=1,
                    penalty.factor = 1/w, lambda = best2)
pred4 <- predict(adp_lasso, data.matrix(test[-c(2)]))
pmse4 <- mean((pred4-test$Apps)^2)
pmse4
```

```
## [1] 871651
```

(vi)

```
library("caret")
```

```
## Warning: package 'caret' was built under R version 4.0.5

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.0.5

## Loading required package: lattice
```

```
cont <- trainControl(method = "repeatedcv",
                     number = 10,
                     search = "random",
                     verboseIter = TRUE)
elastic <- train(Apps~.,
                 data = train,
                 method = "glmnet",
                 tunelength = 20,
                 trControl = cont)
```

```
## + Fold01.Rep1: alpha=0.2411, lambda=0.01201
## - Fold01.Rep1: alpha=0.2411, lambda=0.01201
## + Fold01.Rep1: alpha=0.5743, lambda=5.20745
## - Fold01.Rep1: alpha=0.5743, lambda=5.20745
## + Fold01.Rep1: alpha=0.8795, lambda=0.06614
## - Fold01.Rep1: alpha=0.8795, lambda=0.06614
## + Fold02.Rep1: alpha=0.2411, lambda=0.01201
## - Fold02.Rep1: alpha=0.2411, lambda=0.01201
## + Fold02.Rep1: alpha=0.5743, lambda=5.20745
## - Fold02.Rep1: alpha=0.5743, lambda=5.20745
## + Fold02.Rep1: alpha=0.8795, lambda=0.06614
## - Fold02.Rep1: alpha=0.8795, lambda=0.06614
## + Fold03.Rep1: alpha=0.2411, lambda=0.01201
## - Fold03.Rep1: alpha=0.2411, lambda=0.01201
## + Fold03.Rep1: alpha=0.5743, lambda=5.20745
## - Fold03.Rep1: alpha=0.5743, lambda=5.20745
## + Fold03.Rep1: alpha=0.8795, lambda=0.06614
## - Fold03.Rep1: alpha=0.8795, lambda=0.06614
## + Fold04.Rep1: alpha=0.2411, lambda=0.01201
## - Fold04.Rep1: alpha=0.2411, lambda=0.01201
## + Fold04.Rep1: alpha=0.5743, lambda=5.20745
## - Fold04.Rep1: alpha=0.5743, lambda=5.20745
## + Fold04.Rep1: alpha=0.8795, lambda=0.06614
## - Fold04.Rep1: alpha=0.8795, lambda=0.06614
## + Fold05.Rep1: alpha=0.2411, lambda=0.01201
## - Fold05.Rep1: alpha=0.2411, lambda=0.01201
## + Fold05.Rep1: alpha=0.5743, lambda=5.20745
## - Fold05.Rep1: alpha=0.5743, lambda=5.20745
## + Fold05.Rep1: alpha=0.8795, lambda=0.06614
## - Fold05.Rep1: alpha=0.8795, lambda=0.06614
## + Fold06.Rep1: alpha=0.2411, lambda=0.01201
## - Fold06.Rep1: alpha=0.2411, lambda=0.01201
## + Fold06.Rep1: alpha=0.5743, lambda=5.20745
## - Fold06.Rep1: alpha=0.5743, lambda=5.20745
## + Fold06.Rep1: alpha=0.8795, lambda=0.06614
## - Fold06.Rep1: alpha=0.8795, lambda=0.06614
## + Fold07.Rep1: alpha=0.2411, lambda=0.01201
## - Fold07.Rep1: alpha=0.2411, lambda=0.01201
## + Fold07.Rep1: alpha=0.5743, lambda=5.20745
## - Fold07.Rep1: alpha=0.5743, lambda=5.20745
## + Fold07.Rep1: alpha=0.8795, lambda=0.06614
## - Fold07.Rep1: alpha=0.8795, lambda=0.06614
## + Fold08.Rep1: alpha=0.2411, lambda=0.01201
## - Fold08.Rep1: alpha=0.2411, lambda=0.01201
## + Fold08.Rep1: alpha=0.5743, lambda=5.20745
## - Fold08.Rep1: alpha=0.5743, lambda=5.20745
## + Fold08.Rep1: alpha=0.8795, lambda=0.06614
## - Fold08.Rep1: alpha=0.8795, lambda=0.06614
## + Fold09.Rep1: alpha=0.2411, lambda=0.01201
## - Fold09.Rep1: alpha=0.2411, lambda=0.01201
## + Fold09.Rep1: alpha=0.5743, lambda=5.20745
## - Fold09.Rep1: alpha=0.5743, lambda=5.20745
## + Fold09.Rep1: alpha=0.8795, lambda=0.06614
## - Fold09.Rep1: alpha=0.8795, lambda=0.06614
```

```
## + Fold10.Rep1: alpha=0.2411, lambda=0.01201
## - Fold10.Rep1: alpha=0.2411, lambda=0.01201
## + Fold10.Rep1: alpha=0.5743, lambda=5.20745
## - Fold10.Rep1: alpha=0.5743, lambda=5.20745
## + Fold10.Rep1: alpha=0.8795, lambda=0.06614
## - Fold10.Rep1: alpha=0.8795, lambda=0.06614
## Aggregating results
## Selecting tuning parameters
## Fitting alpha = 0.574, lambda = 5.21 on full training set
```

```r
pred5 <- predict(elastic, test)
pmse5 <- mean((pred5-test$Apps)^2)
pmse5
```

```
## [1] 862879.4
```

(vi)

```r
avg <- mean(test$Apps)
tss <- mean((avg-test$Apps)^2)
data.frame(Linear=1-pmse1/tss, Ridge=1-pmse2/tss, Lasso=1-pmse3/tss,
           Adaptive_Lasso=1-pmse4/tss, Elastic_net=1-pmse5/tss)
```

```
##      Linear     Ridge     Lasso Adaptive_Lasso Elastic_net
## 1 0.9300996 0.9322285 0.9331056      0.9305177   0.9312169
```

Comments: From the table of R squared, we see that all five models perform pretty well with the data, the Lasso regression performs the best with R squared 0.9331056, the OLS regression performs the worst with R squared 0.9300996. There is not much difference among the test errors resulting from these five approaches.