Assignment 1

Exercise 1

ISLR 6.9, parts (a)-(d) and (g), excluding PCR and PLS methods in (e) and (f). Dataset College is included in ISLR package.1 In this exercise, we will predict the number of applications received using the other variables in the College data set.

(a) Split the data set into a training set and a test set.

Besides splitting the data into training and test set, we need to check if there's any missing values and if so, the observations with at least one missing value is removed. It's also good to examine if there's any extreme outliers. No missing values are detected but one extreme outlier (*Rutgers at New Brunswick*) is removed. The response value for Rutgers is 48094 when the 3rd quantile is only 3624 and the second highest value is 21804 for *Purdue University at West Lafayette*. It is also good to keep in mind that cross-validation and splitting the data into sets both use random sampling which means that the outcome varies with different seeds.

```
library(ISLR)

# Checking and removing all NA if there is any

paste("Number of missing values", sum(is.na(College)))

# Summary of the response variable

Apps = sort(College$Apps)
paste("Summary of the response variable")
summary(Apps)

paste("Names of the colleges which receive the most and the second most applications")
rownames(College[which(College$Apps == max(Apps)), ])
rownames(College[which(College$Apps == Apps[length(Apps) - 1]), ])

# Removing the extreme outlier

College = College[-which(College$Apps == max(College$Apps)), ]

# Generating sample vector which is used to split up the data

set.seed(12345)

Sp.set = sample(1:nrow(College), floor(nrow(College)/2), replace = F)

# Splitting the data

TrainCollege = College[Sp.set, ]
TestCollege = College[-Sp.set, ]
yTrain = College$Apps[Sp.set]
yTest = College$Apps[-Sp.set]
```

```
[1] "Number of missing values 0"
[1] "Summary of the response variable"
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     81     776    1558    3002    3624   48094
[1] "Names of the colleges which receive the most and the second most applications"
[1] "Rutgers at New Brunswick"
[1] "Purdue University at West Lafayette"
```

(b) Fit a linear model using least squares on the training set, and report the test error obtained.

For simplicity a model without interactions is fitted. To be noted that the data has quite many outliers at the higher end of response variable which may affect to prediction accuracy.

```
# Fitting linear model wit OLS

OLSCollege = lm(as.formula(paste("Apps ~", paste(colnames(TrainCollege)[c(1,
    3:18)], collapse = "+"))), data = TrainCollege)

# summary(OLSCollege)

OLSColPred = predict.lm(OLSCollege, TestCollege)

# Test RMSE for predictions

paste("RMSE for OLS is", round(sqrt(mean((OLSColPred - yTest)^2)), 3))
```

```
[1] "RMSE for OLS is 1086.963"
```

(c) Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

Because glmnet method can only handle matrices, and characters have to be encoded to numeric in any case, we have to modify the original data into such form that it's accepted by glmnet.

```
library(glmnet)

# Modifying the data to proper form for further analysis

TrainM = model.matrix(Apps ~ ., data = TrainCollege)
TestM = model.matrix(Apps ~ ., data = TestCollege)
```

Before we can appropriately apply cross-validation to decide the parameter λ we need a range $[\lambda_{min}, \lambda_{max}]$ which results in a saturated and a null model, respectively. Usually $\lambda_{min} = 0.001\lambda_{max}$ and the number of calculations for the range is 100. The range could be confirmed by using Lasso due to it's property to set coefficients exactly to zero.

The model for which the CV error is the smallest is chosen and the lambda which gave us the model is printed out. The predictions are calculated by using this model and test set, after which the test error is calculated and printed out.

```
# Applying cross validation to determine the best lambda

# Grid for different lambda values used in cv

grid = 10^seq(4, -2, length = 100)

set.seed(12345)
```

```
ModRid = cv.glmnet(TrainM, yTrain, alpha = 0, lambda = grid, thresh = 1e-12)

BestLambdaRd = ModRid$lambda.min

paste("Minimum lambda for Ridge is", round(BestLambdaRd, 3))

# Calculating perdicted values by using the trained model with minimum
# lambda and test set

PredRid = predict(ModRid, s = BestLambdaRd, newx = TestM)

# Test error

paste("RMSE for Ridge", round(sqrt(mean((yTest - PredRid)^2)), 3))
```

```
[1] "Minimum lambda for Ridge is 18.738"
[1] "RMSE for Ridge 1077.797"
```

(d) Fit a lasso model on the training set, with λ chosen by crossvalidation. Report the test error obtained, along with the number of non-zero coefficient estimates.

The steps are the same as in (c) but the non zero coefficients are printed out after the test error.

```
[1] "Minimum lambda for Lasso 18.738"
[1] "RMSE for Lasso is 1101.812"
 (Intercept)       Accept    Top10perc       Expend
-65.02247645   1.29640549   7.18047087   0.02185438
```

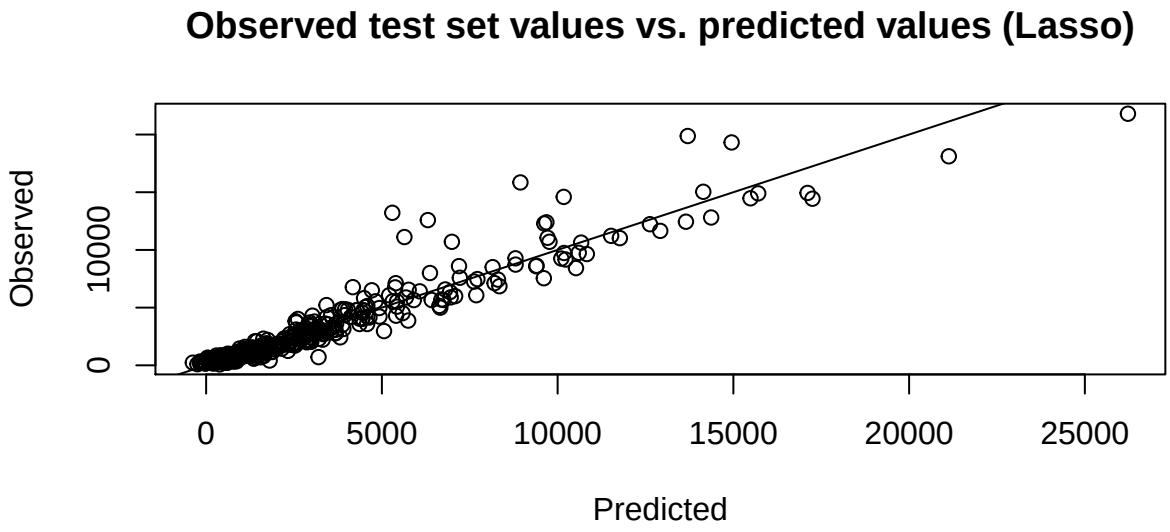
(g) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these three approaches?

With this split into trainind set and test set, the test RMSE (root mean squared error) for OLS, Ridge and Lasso are 1086.963, 1077.797 and 1101.812, respectively. We can also calculate test R^2 to the models.

```
# Calculating R^2 for the models using test set

AvgyTest = mean(yTest)

paste("R^2 for OLS =", round(1 - mean(mean((OLSColPred - yTest)^2)/mean((yTest -
    AvgyTest)^2)), 4))
paste("R^2 for Ridge =", round(1 - mean(mean((PredRid - yTest)^2)/mean((yTest -
    AvgyTest)^2)), 4))
paste("R^2 for Lasso =", round(1 - mean(mean((PredLas - yTest)^2)/mean((yTest -
    AvgyTest)^2)), 4))

# Plotting predicted values against the observed values and fitting a 1:1
# line

plot(PredLas, yTest, main = "Observed test set values vs. predicted values (Lasso)",
    xlab = "Predicted", ylab = "Observed")
abline(coef = c(0, 1))
```

```
[1] "R^2 for OLS = 0.9112"
```

```
[1] "R^2 for Ridge = 0.9127"
[1] "R^2 for Lasso = 0.9088"
```



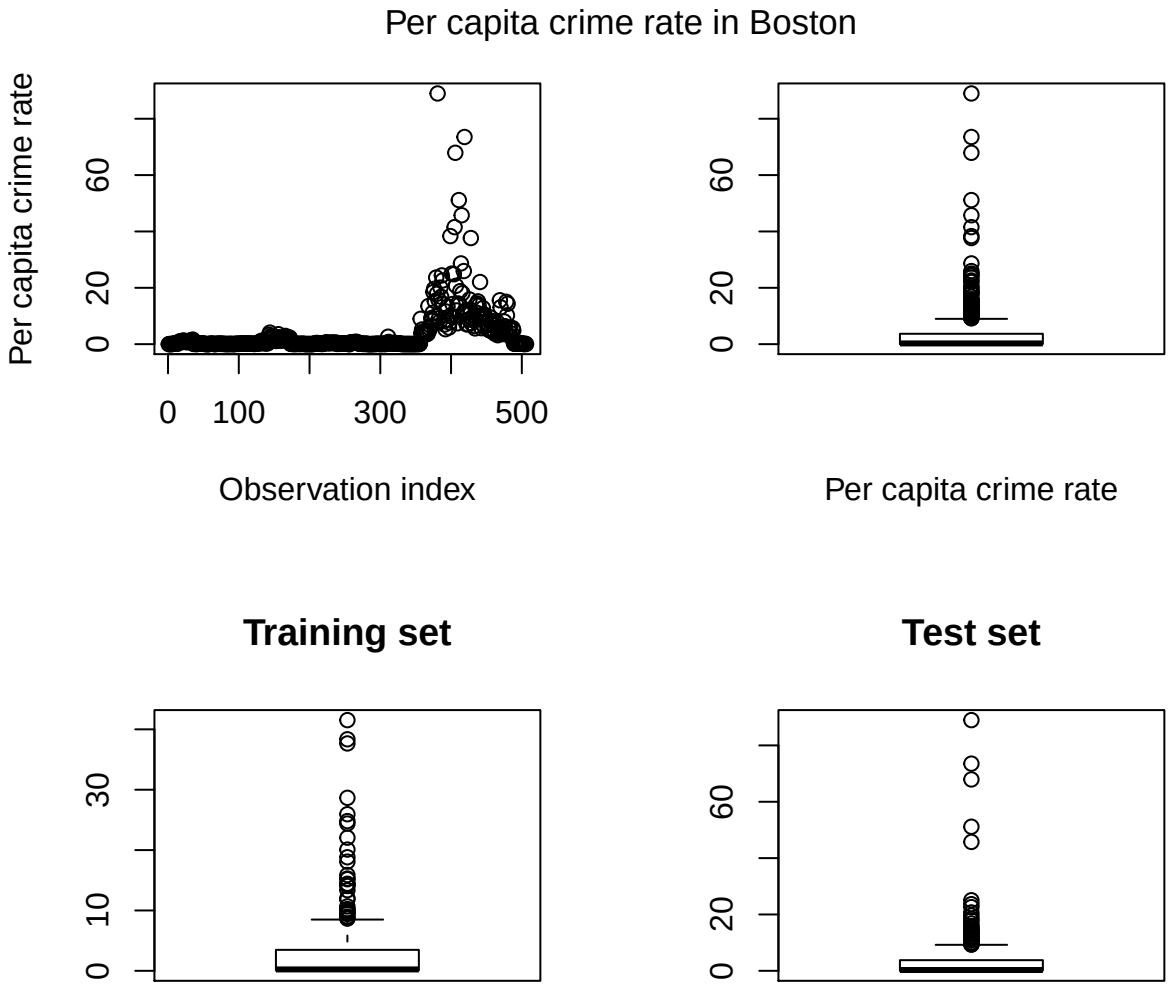
All the models have good $R^2 > 0.9$. The test RMSE is vary only by 14.849 so all the models have approximately the same prediction accuracy. All models tend to underpredict the response variable after 7000 applications, this may due to outliers in the data. To be remembered that both OLS and Ridge have all 18 parameters in their models where as Lasso has only 4.

Exercise 2

ISLR 6.11. Exclude PCR method which will be considered later on. In other words, consider best subset selection, ridge and lasso as covered in this round. Dataset Boston is included in ISLR package. Try to predict per capita crime rate in the Boston dataset.

First we have to split the data into training set and test set, look for missing values and extreme outliers. No missing values are detected but few extreme outliers are detected. They're kept in the data because there's no evidence for example miswriting like in the previus exercise. Again results below may vary due to usage of random sampling.

```
[1] "Number of missing values 0"
```

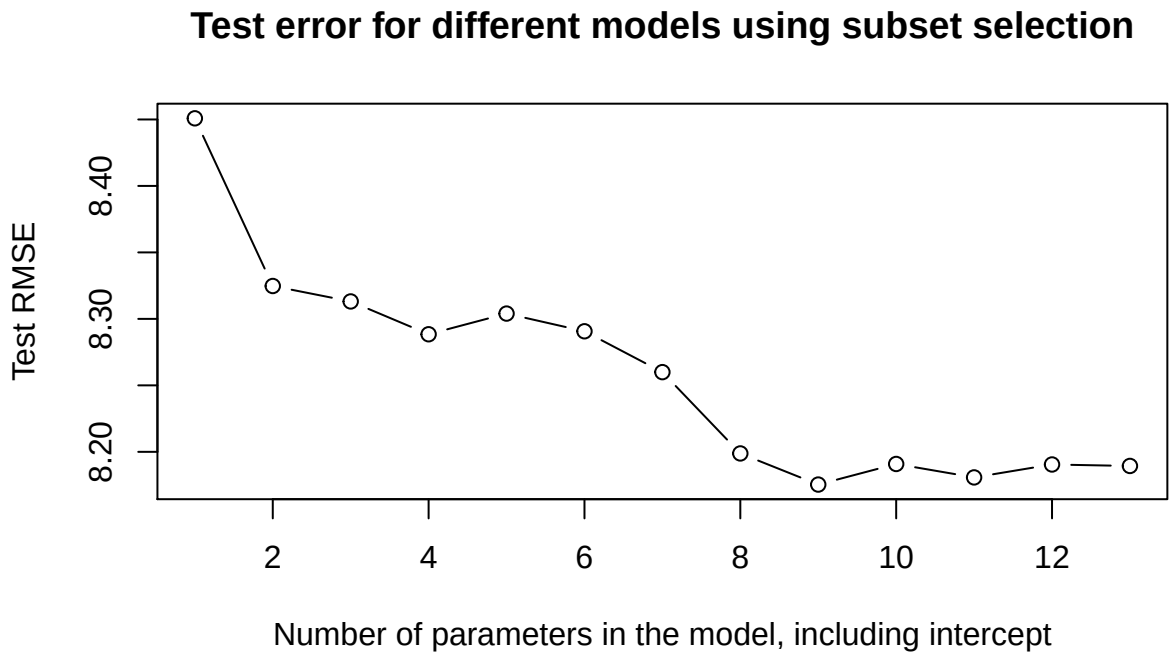


(a) Try out some of the regression methods explored in this chapter, such as best subset selection, the lasso, ridge regression. Present and discuss results for the approaches that you consider.

A grid is used in cross-validation for Ridge and Lasso. The grid has been simulated with Lasso, basically the values in the grid vary between minimum and maximum λ. In subset selection training set and test (validation) set are used to form the predictions. The results are shown below.

```
[1] "Final model using subset selection"
 (Intercept)           zn         chas          nox          dis
 8.639645020  0.029201563 -1.350884316 -7.309733323 -0.483100170
         rad      ptratio        black        lstat         medv
 0.473144736 -0.228202470 -0.003117134  0.211708691 -0.069186149
```

```
[1] "Test RMSE for subset selection 8.175"
```




```
[1] "Minimum lambda for Lasso = 0.032"
[1] "Test RMSE for Lasso 8.213"
(Intercept)         rad       lstat
 0.38518022  0.26657071  0.02126624
```

```
[1] "Minimum lambda for Ridge = 0.521"
[1] "Test RMSE for Ridge 8.264"
14 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept)  0.986161917
zn          -0.002508028
indus        0.031598267
chas        -0.229574783
nox          2.158724491
rm          -0.217731237
age          0.006063165
dis         -0.100316352
rad          0.061508005
tax          0.002740562
ptratio      0.075962703
black       -0.002933578
lstat        0.048005262
medv        -0.027371451
```

(b) Propose a model (or set of models) that seem to perform well on this data set, and justify your answer. Make sure that you are evaluating model performance using validation set error, crossvalidation, or some other reasonable alternative, as opposed to using training error.

The test RMSE for subset selection, Ridge and Lasso are 8.175, 8.264 and 8.213, respectively. Subset selection ends up in a model with 10 parameters, Ridge has all i.e. 14 parameters and Lasso has only 3 parameters. Note that due to random sampling the models may differ. Because the deviance between test RMSE for all the models is small and usually a parsimonious model is preferred, Lasso seems to be a good choice.

(c) Does your chosen model involve all of the features in the dataset? Why or why not?

Lasso model doesn't involve all the features and has only 3 parameters: the intercept and the coefficients for variables rad and lstat. Due to Lasso's L_1 norm some coefficients may shrink exactly to zero which has happend here.

Exercise 3

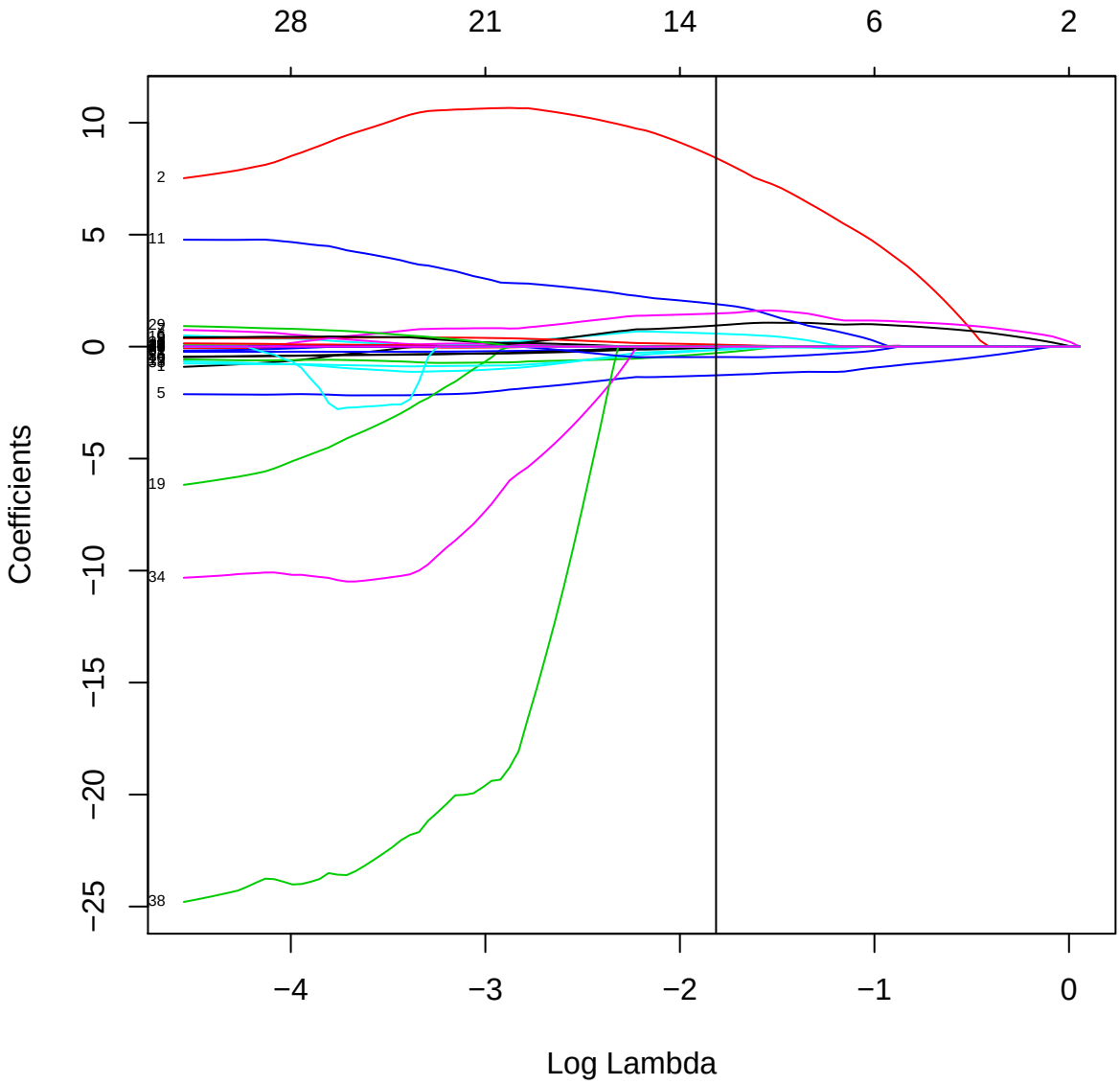
Varian (2014, JEP): Replicate the Lasso analysis in connection to Table 4 in terms of finding the Lasso estimates. Dataset is FLS-data.csv and can be found (with R code) in the supplementary material of the article published in the JEP (see the link in Moodle). Report and comment on the estimation results (and different stages of lasso modelling) more generally than just reporting the rank of the predictors as in Table 4 in Varian (2014). In other words, which variables lasso excludes from the model, how to specify the tuning parameter and present the estimation result of the final model.

The steps of the modelling can be seen in the code. Cross-validation is used to figure out λ and the model for which the CV error is the smallest is chosen. The result is below the code but the name of the variable and the exact coefficient is printed out.

```
library(readr)

# Importing data

FLSdata <- read_csv("FLS-data.csv")

# Checking for missing values: sum(is.na(FLSdata)) Checking for outliers:
# boxplot(FLSdata$y)

# Splitting the data

set.seed(12345)
Sp.FLS = sample(1:nrow(FLSdata), size = floor(nrow(FLSdata)/2), replace = F)

FLSTrain = as.matrix(FLSdata[Sp.FLS, ])
FLSTrainY = FLSTrain[, 1]
FLSTrain = FLSTrain[, -1]

FLSTest = as.matrix(FLSdata[-Sp.FLS, ])
FLSTestY = FLSTest[, 1]
FLSTest = FLSTest[, -1]

# Figuring out the maximum lambda to creat the grid for CV sim.lbd(FLSTrain,
# FLSTrainY)
```

```r
gridFLS = 10^seq(3, -2, length = 100)

# Lasso model

set.seed(1234)
FLSLasB = cv.glmnet(FLSTrain, FLSTrainY, alpha = 1, lambda = gridFLS)

FLSBestL = FLSLasB$lambda.min

paste("Minimum lambda for Lasso =", round(FLSBestL, 3))

# Calculating perdicted values by using the trained model with minimum
# lambda and test set

FLSPredLasB = predict(FLSLasB, s = FLSBestL, newx = FLSTest)

# Test Error

paste("Test RMSE for Lasso", round(sqrt(mean((FLSPredLasB - FLSTestY)^2)), 3))

# Printing out non-zero coefficients

coef(FLSLasB)[which(coef(FLSLasB)[, 1] != 0), ]

# Plotting the coefficients appearance in the model

FLSLasBfit = glmnet(FLSTrain, FLSTrainY, alpha = 1)

plot(FLSLasBfit, xvar = "lambda", label = T)

# Adding the line which is used in final model

abline(v = log(FLSBestL))
```

```
[1] "Minimum lambda for Lasso = 0.163"
[1] "Test RMSE for Lasso 1.507"
(Intercept) Confuncious   SubSahara      Muslim Rule of Law    Yrs Open
-0.07034172  7.92019311 -1.25393902  0.53305873  0.06813942  1.00478959
    Eco Org Protestants  NEquip Inv  LatAmerica PrSc Enroll  Pol Rights
 0.06702271 -0.19970925  1.78442415 -0.11007851  1.50524039 -0.01893380
   Work/Pop  Rev & Coup
-0.06230949 -0.46366047
```



The model differs from the model introduced by Varian. Firstly there's 13 variables in the final model and the order is different which can be somewhat seen on the picture above.

Exercise 4

It is well-known that ridge regression tends to give similar coefficient values to correlated variables, whereas the lasso may give quite different coefficient values to correlated variables. We will now explore this property in a very simple setting. Suppose that $n = 2$, $p = 2$, $x_{11} = x_{12}$, $x_{21} = x_{22}$. Furthermore, suppose that $y_1 + y_2 = 0$ and $x_{11} + x_{21} = 0$ and $x_{12} + x_{22} = 0$, so that the estimate for the intercept in a least squares, ridge regression, or lasso model is zero: $\hat{\beta}_0 = 0$.

(a) Write out the ridge regression optimization problem in this setting.

$$\hat{\boldsymbol{\beta}}_\lambda = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \sum_{j=1}^{p} \hat{\beta}_j x_j)^2 + \lambda \sum_{i=1}^{p} \hat{\beta}_i^2 \right\}$$

In this case, $\hat{\beta}_0 = 0$ and $n = p = 2$. So, the optimization looks like:

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}} \Big\{ (y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2 + \lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2) \Big\}$$

(b) Argue that in this setting, the ridge coefficient estimates satisfy $\hat{\beta}_1 = \hat{\beta}_2$.

Given that, $x_{11} = x_{12} = x_1$ and $x_{21} = x_{22} = x_2$. We take derivatives of above expression with respect to both $\hat{\beta}_1$ and $\hat{\beta}_2$ and setting them equal to zero find that,

$$\hat{\beta^*}_1 = \frac{x_1 y_1 + x_2 y_2 - \hat{\beta^*}_2 (x_1^2 + x_2^2)}{\lambda + x_1^2 + x_2^2}$$

and

$$\hat{\beta^*}_2 = \frac{x_1 y_1 + x_2 y_2 - \hat{\beta^*}_1 (x_1^2 + x_2^2)}{\lambda + x_1^2 + x_2^2}.$$

Symmetry in these expressions suggests that $\hat{\beta^*}_1 = \hat{\beta^*}_2$ ### (c) Write out the lasso optimization problem in this setting.

$$\hat{\boldsymbol{\beta}}_\lambda = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \Big\{ (y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2 + \lambda(|\hat{\beta}_1| + |\hat{\beta}_2|) \Big\}$$

(d) Argue that in this setting, the lasso coefficients $\hat{\beta}_1$ and $\hat{\beta}_2$ are not unique-in other words, there are many possible solutions to the optimization problem in (c). Describe these solutions.

The Lasso constraint takes the form $|\hat{\beta}_1| + |\hat{\beta}_2| < t$, which when plotted takes the familiar shape of a diamond centered at origin $(0, 0)$. The above argument can be simplified to

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}} \Big\{ (y_1 - (\hat{\beta}_1 + \hat{\beta}_2) x_{11})^2 + (y_2 - (\hat{\beta}_1 + \hat{\beta}_2) x_{21})^2 + \lambda(|\hat{\beta}_1| + |\hat{\beta}_2|) \Big\}$$

and using the facts given in the beginning and replacing variables with correct form, the first part of the equation simplifies to quadratic function $2(y_1 - (\hat{\beta}_1 + \hat{\beta}_2) x_{11})^2$.

Using calculus, the vertex point, being a maximum or minimum of the quadratic function, can be obtained by finding the roots of the derivative which results to: $\hat{\beta}_1 + \hat{\beta}_2 = \frac{y_1}{x_{11}}$. This is a line parallel to the edge of Lasso-diamond $\hat{\beta}_1 + \hat{\beta}_2 = t$, which means that the entire edge $\hat{\beta}_1 + \hat{\beta}_2 = t$ is a potential solution to the Lasso optimization problem.

Similar argument can be made for the opposite Lasso-diamond edge: $\hat{\beta}_1 + \hat{\beta}_2 = -t$.

Thus, the Lasso problem does not have a unique solution. The general form of solution is given by two line segments:

$\hat{\beta}_1 + \hat{\beta}_2 = t; \hat{\beta}_1 \geq 0; \hat{\beta}_2 \geq 0$ and $\hat{\beta}_1 + \hat{\beta}_2 = -t; \hat{\beta}_1 \leq 0; \hat{\beta}_2 \leq 0$.

Rest of the R code not shown above

```
install.packages("assertthat")
install.packages("ISLR")
install.packages("glmnet")
install.packages("rlang")
install.packages("MASS")
install.packages("leaps")
```

```
# CV to to determine the best lambda
set.seed(12345)
ModLas = cv.glmnet(TrainM, yTrain, alpha = 1, lambda = grid)

BestLambdaLa = ModLas$lambda.min

paste("Minimum lambda for Lasso", round(BestLambdaLa, 3))

# Calculating perdicted values by using the trained model with minimum
# lambda and test set

PredLas = predict(ModLas, s = BestLambdaLa, newx = TestM)

# Test Error

paste("RMSE for Lasso is", round(sqrt(mean((PredLas - yTest)^2)), 3))

# Printing out non-zero coefficients

coef(ModLas)[which(coef(ModLas)[, 1] != 0), ]
```

```
library(MASS)

# Checkinf if there exist missing values

paste("Number of missing values", sum(is.na(Boston)))

# Plotting the response variable

par(mfrow = c(1, 2))
plot(Boston$crim, xlab = "Observation index", ylab = "Per capita crime rate")
boxplot(Boston$crim, xlab = "Per capita crime rate")
mtext("Per capita crime rate in Boston", side = 3, line = -3, outer = TRUE,
    cex = 1.1)

# Creating training, test and validation sets

set.seed(12345)

Sp.Train = sample(1:nrow(Boston), size = floor(nrow(Boston)/2), replace = F)
```

```r
TrainB = Boston[Sp.Train, ]
TestB = Boston[-Sp.Train, ]

TrainY = TrainB[, 1]
TestY = TestB[, 1]

# Visual diagnostics of the distribution of response variable in each set

par(mfrow = c(1, 2))
boxplot(TrainY, main = "Training set")
boxplot(TestY, main = "Test set")

# Changing data to correct form

TrainMB = model.matrix(crim ~ ., data = TrainB)
# The first column must be removed to run regsubsets() otherwise there exist
# linear dependencies which results to error message
TrainMB = TrainMB[, -1]
TestMB = model.matrix(crim ~ ., data = TestB)

# Creating a method sim.lbd() to find the maximum lambda which gives a null
# model

sim.lbd = function(md.matrix, response, ...) {
    CoefMat = data.frame(matrix(ncol = 10, nrow = 10))
    colnames(CoefMat) = c("int^1", "int^2", "int^3", "int^4", "int^5", "int^6",
        "int^7", "int^8", "int^9", "int^10")
    for (i in 1:10) {
        for (j in 1:10) {
            LamCvFit = glmnet(md.matrix, response, alpha = 1, lambda = i^j)
            if (!(any(LamCvFit$beta[1:13] != 0))) {
                CoefMat[i, j] = "Null"
            } else {
                CoefMat[i, j] = "Coef"
            }
        }
    }
    return(CoefMat)
}

# Grid used in CV sim.lbd(TrainMB,TrainY)

gridB = 10^seq(3, -2, length = 100)
```

```r
library(leaps)

# Forming the models, nvmax = 14 because intercept is included in each model

SubsB = regsubsets(x = TrainMB, y = TrainY, nvmax = 13)

# Storing test errors for each model from which the final model is chosen

test.errors =rep(NA, 13)
```

```r
for(i in 1:13){
  # Storing the coefficients for each model including i coefficients
  coefi=coef(SubsB ,id=i)
  # Forming predictions by multiplying each observed validation set
  # value with the corresponding coefficient
  pred=TestMB[,names(coefi)]%*%coefi
  # Calculating test error
  test.errors[i]=mean((TestY-pred)^2)
}

# Selecting the model for which the test error is the smallest

paste("Final model using subset selection")

BestCoefB = coef(SubsB, id = which.min(test.errors))
BestCoefB

paste("Test RMSE for subset selection",round(sqrt(test.errors[which.min(test.errors)]),3))

par(mfrow=c(1,1))
plot(sqrt(test.errors), type = "b", ylab = "Test RMSE",
     main = "Test error for different models using subset selection",
     xlab = "Number of parameters in the model, including intercept")
```

```r
set.seed(12345)
LasB = cv.glmnet(TrainMB, TrainY, alpha = 1, lambda = gridB)

BestL = LasB$lambda.min

paste("Minimum lambda for Lasso =", round(BestL, 3))

# Calculating perdicted values by using the trained model with minimum
# lambda and test set

PredLasB = predict(LasB, s = BestL, newx = TestMB[, -1])

# Test Error

paste("Test RMSE for Lasso", round(sqrt(mean((PredLasB - TestY)^2)), 3))

# Printing out non-zero coefficients

coef(LasB)[which(coef(LasB)[, 1] != 0), ]
```

```r
set.seed(12345)
RidB = cv.glmnet(TrainMB, TrainY, alpha = 0, lambda = gridB)

BestRd = RidB$lambda.min

paste("Minimum lambda for Ridge =", round(BestRd, 3))

# Calculating perdicted values by using the trained model with minimum
# lambda and test set
```

```r
PredRidB = predict(RidB, s = BestRd, newx = TestMB[, -1])

# Test error

paste("Test RMSE for Ridge", round(sqrt(mean((TestY - PredRidB)^2)), 3))
coef(RidB)
```