Question 1 [20 Points] Linear Model Selection

We will use the Boston Housing data again. This time, we do not scale the covariate. We will still remove `medv`, `town` and `tract` from the data and use `cmedv` as the outcome. If you do not use R, you can download a '.csv' file from the course website.

```r
library(mlbench)
data(BostonHousing2)
BH = BostonHousing2[, !(colnames(BostonHousing2) %in% c("medv", "town", "tract"))]
full.model <- lm(cmedv ~ ., data=as.data.frame(BH))
```

Answer the following questions:

a. [5 Points] Report the most significant variable from this full model with all features.

Ans: Both `rm` and `lstat` have the smallest p-values among all variables (2e-16). After scaling and centering the parameters, `lstat` has the largest coefficient so it's the most significant:

```r
bhCoef <- coef(lm(cmedv~., data=as.data.frame(cbind(scale(BH[,1:6]), BH[,7], scale(BH[,8:16])))))
bhCoef[order(bhCoef)]
```

```
##        lstat          dis  (Intercept)          tax      ptratio          nox
## -0.417972819 -0.321039342 -0.300183890 -0.236526155 -0.206804965 -0.199703772
##         crim          lon          age        indus          lat            b
## -0.097935969 -0.032316441  0.007564852  0.011390378  0.030245087  0.091235409
##           zn           V7           rm          rad
##  0.118273098  0.280763490  0.287232811  0.290850755
```

b. [5 Points] Starting from this full model, use stepwise regression with both forward and backward and BIC criterion to select the best model. Which variables are removed from the full model?

Ans: age, indus, lon and lat were removed

c. [5 Points] Starting from this full model, use the best subset selection and list the best model of each model size.

```r
library(leaps)
p <- ncol(BH)
b = regsubsets(cmedv ~ ., data=as.data.frame(BH), nvmax = p)
rs = summary(b)
rs$which
```

```
##    (Intercept)   lon   lat  crim    zn indus chas1   nox    rm   age   dis   rad   tax
## 1         TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 2         TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## 3         TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## 4         TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE
## 5         TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE
## 6         TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE
## 7         TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE
## 8         TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE
## 9         TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## 10        TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## 11        TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## 12        TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## 13        TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## 14        TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## 15        TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##    ptratio     b lstat
```

```
## 1    FALSE FALSE  TRUE
## 2    FALSE FALSE  TRUE
## 3     TRUE FALSE  TRUE
## 4     TRUE FALSE  TRUE
## 5     TRUE FALSE  TRUE
## 6     TRUE FALSE  TRUE
## 7     TRUE  TRUE  TRUE
## 8     TRUE  TRUE  TRUE
## 9     TRUE  TRUE  TRUE
## 10    TRUE  TRUE  TRUE
## 11    TRUE  TRUE  TRUE
## 12    TRUE  TRUE  TRUE
## 13    TRUE  TRUE  TRUE
## 14    TRUE  TRUE  TRUE
## 15    TRUE  TRUE  TRUE
```

d. [5 Points] Use the Cp criterion to select the best model from part c). Which variables are removed from the full model? What is the most significant variable?

```
rs$which[which.min(rs$cp),]
```

```
## (Intercept)         lon         lat        crim          zn       indus       chas1
##        TRUE       FALSE       FALSE        TRUE        TRUE       FALSE        TRUE
##         nox          rm         age         dis         rad         tax     ptratio
##        TRUE        TRUE       FALSE        TRUE        TRUE        TRUE        TRUE
##           b       lstat
##        TRUE        TRUE
```

```
summary(lm(cmedv~.-lon-lat-indus-age, data=BH))
```

```
##
## Call:
## lm(formula = cmedv ~ . - lon - lat - indus - age, data = BH)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.566  -2.686  -0.552   1.790  26.167
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.244827   5.022209   7.217 2.02e-12 ***
## crim         -0.106657   0.032487  -3.283 0.001099 **
## zn            0.047099   0.013402   3.514 0.000481 ***
## chas1         2.727209   0.846606   3.221 0.001360 **
## nox         -17.316823   3.503652  -4.943 1.06e-06 ***
## rm            3.778662   0.402685   9.384  < 2e-16 ***
## dis          -1.520270   0.184071  -8.259 1.35e-15 ***
## rad           0.296555   0.062836   4.720 3.08e-06 ***
## tax          -0.012077   0.003342  -3.613 0.000333 ***
## ptratio      -0.917035   0.127912  -7.169 2.77e-12 ***
## b             0.009202   0.002650   3.473 0.000561 ***
## lstat        -0.528441   0.047001 -11.243  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.694 on 494 degrees of freedom
```

```
## Multiple R-squared:  0.7444, Adjusted R-squared:  0.7387
## F-statistic: 130.8 on 11 and 494 DF,  p-value: < 2.2e-16
```

Ans: `lon`, `lat`, `indus`, and `age` were removed from the model. Most significant variable is still `lstat`

Question 2 (50 Points) Code Your Own Lasso

For this question, we will write our own Lasso code. You are not allowed to use any built-in package that already implements Lasso. First, we will generate simulated data. Here, only X_1, X_2 and X_3 are important, and we will not consider the intercept term.

```
library(MASS)
set.seed(1)
n = 200
p = 200

# generate data
V = matrix(0.2, p, p)
diag(V) = 1
X = as.matrix(mvrnorm(n, mu = rep(0, p), Sigma = V))
y = X[, 1] + 0.5*X[, 2] + 0.25*X[, 3] + rnorm(n)

# we will use a scaled version
X = scale(X)
y = scale(y)
```

As we already know, coordinate descent is an efficient approach for solving Lasso. The algorithm works by updating one parameter at a time, and loop around all parameters until convergence.

a. [10 Points] Hence, we need first to write a function that updates just one parameter, which is also known as the soft-thresholding function. Construct the function in the form of `soft_th <- function(b, lambda)`, where `b` is a number that represents the one-dimensional linear regression solution, and `lambda` is the penalty level. The function should output a scaler, which is the minimizer of

$$(x - b)^2 + \lambda|b|$$

Ans:

```
soft_th <- function(b, lambda) {
  ifelse(b > lambda/2, b-lambda/2, ifelse(b < -lambda/2, b+lambda/2, 0))
}
```

b. [10 Points] Now lets pretend that at an iteration, the current parameter $\boldsymbol{\beta}$ value is given below (as `beta_old`, i.e., $\boldsymbol{\beta}^{\text{old}}$). Apply the above soft-thresholding function to update all p parameters sequencially one by one to complete one "loop" of the updating scheme. Please note that we use the Gauss-Seidel style coordinate descent, in which the update of the next parameter is based on the new values of previous entries. Hence, each time a parameter is updated, you should re-calculate the residual

$$\mathbf{r} = \mathbf{y} - \mathbf{X}^{\mathrm{T}}\boldsymbol{\beta}$$

so that the next parameter update reflects this change. After completing this one enrire loop, print out the first 3 observations of $\mathbf{r}$ and the nonzero entries in the updated $\boldsymbol{\beta}^{\text{new}}$ vector. For this question, use `lambda` $= 0.7$ and

Ans: First 3 entries of residual: 0.55916349 0.22405764 -0.04292186 Non-zero entries of beta: 0.3210908 0.0939837

```r
beta_old = rep(0, p)

coord_descent <- function(X, y, lambda, j) {
  r <<- (y - X[,-j] %*% beta_old[-j])
  beta_old[j] <<- soft_th( X[,j] %*% r / sum( X[,j]^2 ), lambda)
}

for (j in (1:p)) {
  coord_descent(X, y, 0.7, j)
}
print(r[1:3])
```

```
## [1]  0.55916349  0.22405764 -0.04292186
```

```r
print(beta_old[beta_old > 0])
```

```
## [1] 0.3210908 0.0939837
```

c. [25 Points] Now, let us finish the entire Lasso algorithm. We will write a function `myLasso(X, y, lambda, tol, maxitr)`. Set the tolerance level `tol` = 1e-5, and `maxitr` = 100 as the default value. Use the "one loop" code that you just wrote in the previous question, and integrate that into a grand for-loop that will continue updating the parameters up to `maxitr` runs. Check your parameter updates once in this grand loop and stop the algorithm once the ℓ_1 distance between $\boldsymbol{\beta}^{\text{new}}$ and $\boldsymbol{\beta}^{\text{old}}$ is smaller than `tol`. Use `beta_old = rep(0, p)` as the initial value, and `lambda` = 0.3. After the algorithm converges, report the following: i) the number of iterations took; ii) the nonzero entries in the final beta parameter estimate, and iii) the first three observations of the residual. Please write your algorithm as efficient as possible.

Ans: i) 8 iterations ii) 6 non-zero entries in beta iii) 0.47099095 0.25510553 0.02272195

```r
myLasso <- function(X, y, lambda, tol, maxitr) {
  for (itr in 1:maxitr) {
    before <- beta_old
    for (j in 1:p) {
      coord_descent(X, y, lambda, j)
    }
    if (sum(abs(beta_old - before)) < tol) {
      break
    }
  }
  sprintf("Iterations: %d", itr)
}

beta_old = rep(0, p)
myLasso(X, y, 0.3, 1e-5, 100)
```

```
## [1] "Iterations: 8"
```

```r
print(beta_old[1:3])
```

```
## [1] 0.47099095 0.25510553 0.02272195
```

d. [5 Points] Now we have our own Lasso function, let's check the result and compare it with the `glmnet` package. Note that for the glmnet package, their `lambda` should be set as half of ours. Comment on the accuracy of the algorithm that we wrote. Please note that the distance of the two solutions should not be larger than 0.005.

Ans: glmnet result coefficients are different from myLasso by less than 0.0003

```r
library("glmnet")
lasso <- glmnet(X, y, alpha=1, lambda=0.15)
coef(lasso)[1:p+1] - beta_old
```

```
##   [1] -0.0002422493 -0.0001698450 -0.0001697053 -0.0002256843  0.0000000000  0.0000000000
##   [7] -0.0001637311  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##  [13]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##  [19]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##  [25]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##  [31]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##  [37]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##  [43]  0.0000000000  0.0000000000 -0.0001783890  0.0000000000  0.0000000000  0.0000000000
##  [49]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##  [55]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##  [61]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##  [67]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##  [73]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##  [79]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##  [85]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##  [91]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##  [97]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [103]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [109]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [115]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [121]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [127]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [133]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [139]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [145]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [151]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [157]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [163]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [169]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [175]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [181]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [187]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [193]  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
## [199]  0.0000000000  0.0000000000
```

Question 3 (30 Points) Cross-Validation for Model Selection

We will use the Walmart Sales data provided on Kaggle. For this question, we will use only the Train.csv file. The file is also available at here.

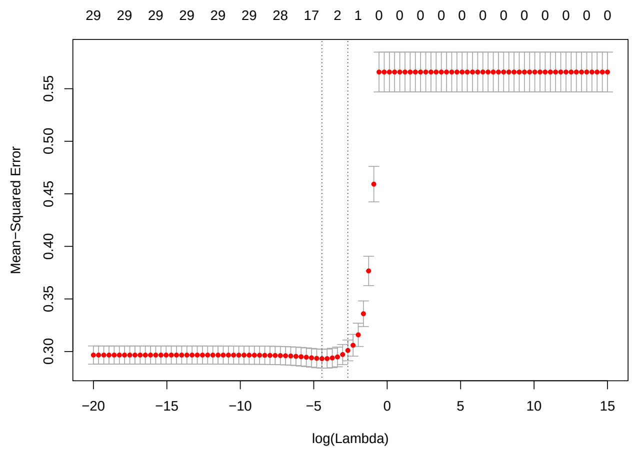
a. [10 Points] Do the following to process the data:
    - Read data into R
    - Convert character variables into factors
    - Remove `Item_Identifier`
    - Further convert all factors into dummy variables
b. [20 Points] Use all variables to model the outcome `Item_Outlet_Sales` in its *log* scale. First, we randomly split the data into two parts with equal size. Make sure that you set a random seed so that the result can be replicated. Treat one as the training data, and the other one as the testing data. For the training data, perform the following:

```
n <- nrow(WalMartData)
p <- ncol(WalMartData)-1
ntest = round(n / 2)
ntrain = n - ntest
set.seed(0)
test.id = sample(1:n, ntest)
Xtrain = WalMartData[test.id, 1:p]
Ytrain = log(WalMartData[test.id, p+1])
Xtest = WalMartData[-test.id, 1:p]
Ytest = log(WalMartData[-test.id, p+1])
```

```
+ Use cross-validation to select the best Lasso model. Consider both `lambda.min` and `lambda.min`. Pro
```

Ans: For Lasso, lambda.1se and lambda.min are 0.098 and 0.117 respectively. The model seems to fit very well with just 1 to 2 variables.

```
lam.seq = exp(seq(-20, 15, length=100))
lasso.cv.out = cv.glmnet(Xtrain, Ytrain, alpha=1, lambda=lam.seq)
plot(lasso.cv.out)
```

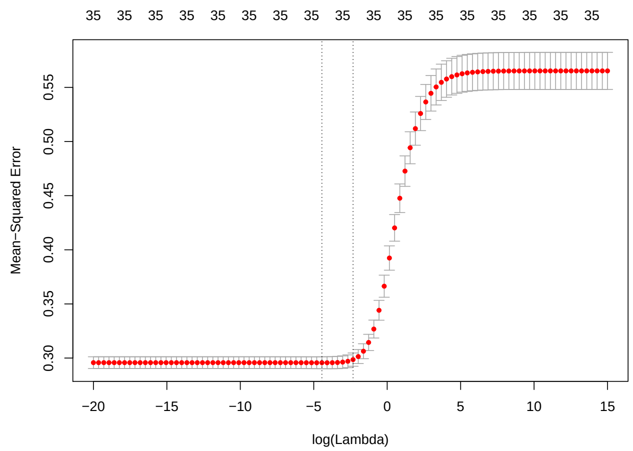



```
lasso.1se <- glmnet(Xtrain, Ytrain, lambda=lasso.cv.out$lambda.1se)
lasso.min <- glmnet(Xtrain, Ytrain, lambda=lasso.cv.out$lambda.min)
```

```
+ Use cross-validation to select the best Ridge model. Consider both `lambda.min` and `lambda.min`. Pro

Ans: After CV, ridge model encourages a slightly higher lambda regularisation than lasso, under almost
identical MSE. The lambda.1se and lambda.min values are 0.199 and 0.00825 respectively.
```

```
ridge.cv.out = cv.glmnet(Xtrain, Ytrain, alpha=0, lambda=lam.seq)
plot(ridge.cv.out)
```




```
ridge.1se <- glmnet(Xtrain, Ytrain, lambda=ridge.cv.out$lambda.1se)
ridge.min <- glmnet(Xtrain, Ytrain, lambda=ridge.cv.out$labmda.min)
```

```
+ Test these four models on the testing data and report and compare the prediction accuracy

Ans: For lowest test errors, ridge regression plus lambda.1se reduced overfitting best
```

```
testModelErr <- function(model) {
  ypredict <<- predict(model, newx=Xtest)
  mse <- mean((ypredict - Ytrain)^2)
}

mseList <- sapply(list(lasso.1se, lasso.min, ridge.1se, ridge.min), testModelErr)
barplot(mseList, main="MSE by model", names=c("lasso.1se", "lasso.min", "ridge.1se", "ridge.min"), ylab=
```