Homework3 - Stats 2620

Tyler Poelking, Cornell Blake

10/15/2017

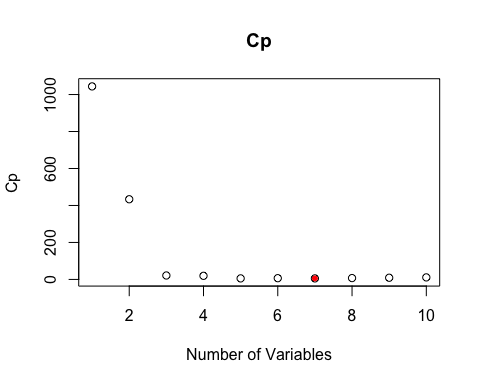
# Question 1

1. The lasso, relative to least squares: iii is true. Lasso is less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.
2. Ridge regression relative to least squares: Again, iii is true. Ridge regression is less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.
3. Non-linear methods relative to least squares: ii is true. Non- linear methods are more flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

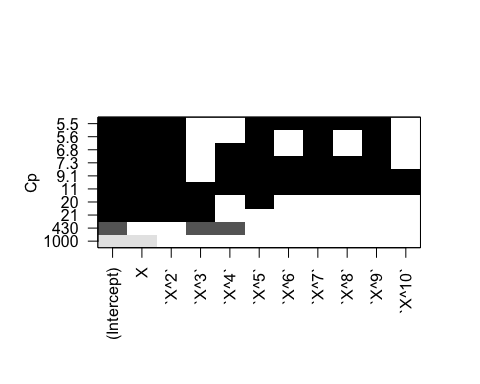
# a-c)

#Part a  
set.seed(42)  
X = rnorm(n=100)  
e = rnorm(n=100)  
  
# B0=1 B1=3 B2=2 B3=1  
B = c(1,3,2,1)  
  
#Part B  
Y = rep(B[1], 100)  
Y = Y + (B[2]\*X)+ (B[3]\*X^2)+ (B[4]\*X^3)+ e  
  
#Part C  
x\_data = data.frame(X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^10, Y)  
colnames(x\_data) =c('X', 'X^2', 'X^3', 'X^4', 'X^5', 'X^6', 'X^7', 'X^8', 'X^9', 'X^10', 'Y')  
  
  
  
a<-regsubsets(Y~., data = x\_data, nvmax = 10)  
  
mysummary(a)

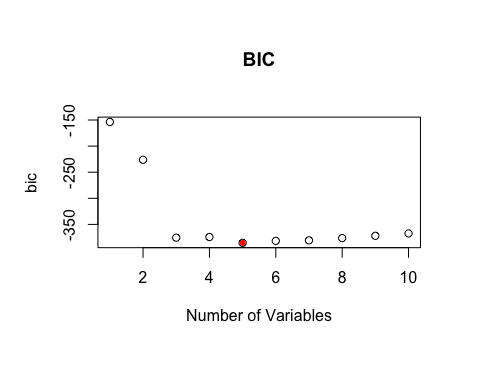
## Subset selection object  
## Call: regsubsets.formula(Y ~ ., data = x\_data, nvmax = 10)  
## 10 Variables (and intercept)  
## Forced in Forced out  
## X FALSE FALSE  
## `X^2` FALSE FALSE  
## `X^3` FALSE FALSE  
## `X^4` FALSE FALSE  
## `X^5` FALSE FALSE  
## `X^6` FALSE FALSE  
## `X^7` FALSE FALSE  
## `X^8` FALSE FALSE  
## `X^9` FALSE FALSE  
## `X^10` FALSE FALSE  
## 1 subsets of each size up to 10  
## Selection Algorithm: exhaustive  
## X `X^2` `X^3` `X^4` `X^5` `X^6` `X^7` `X^8` `X^9` `X^10`  
## 1 ( 1 ) "\*" " " " " " " " " " " " " " " " " " "   
## 2 ( 1 ) " " " " "\*" "\*" " " " " " " " " " " " "   
## 3 ( 1 ) "\*" "\*" "\*" " " " " " " " " " " " " " "   
## 4 ( 1 ) "\*" "\*" "\*" " " "\*" " " " " " " " " " "   
## 5 ( 1 ) "\*" "\*" " " " " "\*" " " "\*" " " "\*" " "   
## 6 ( 1 ) "\*" "\*" " " "\*" "\*" " " "\*" " " "\*" " "   
## 7 ( 1 ) "\*" "\*" " " " " "\*" "\*" "\*" "\*" "\*" " "   
## 8 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" " "   
## 9 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 10 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## [1] "Cps: "  
## [1] 1043.992234 433.630846 21.318293 19.561749 5.568909  
## [6] 6.763693 5.528537 7.337124 9.080219 11.000000



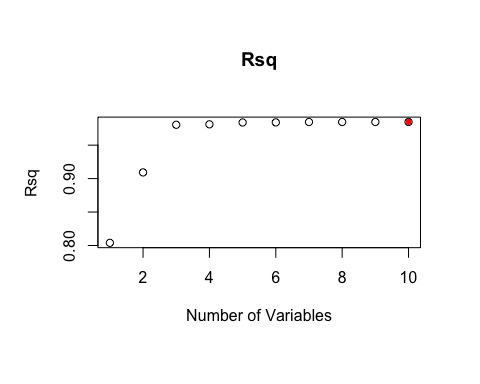
## [1] "Optimal model variable count:"  
## [1] 7



## [1] "BICs: "  
## [1] -153.7239 -226.1567 -375.3712 -374.1372 -385.3111 -381.5702 -380.5148  
## [8] -376.1237 -371.8065 -367.2914



## [1] "Optimal model variable count:"  
## [1] 5  
## [1] "R-squared: "  
## [1] 0.8039415 0.9092568 0.9805113 0.9811573 0.9839078 0.9840463 0.9846027  
## [8] 0.9846356 0.9846798 0.9846936



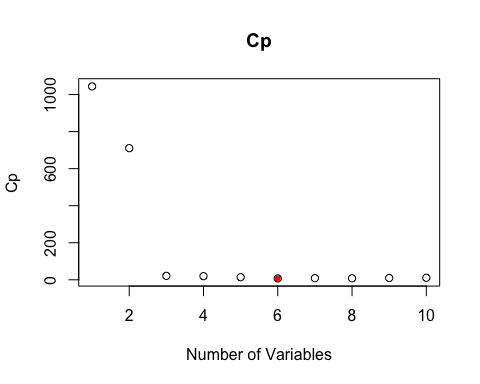
## [1] "Optimal model variable count:"  
## [1] 10

The 7th model is best according to Cp. The 5th model is best according to BIC. The 8th model is best according to R-Squared.

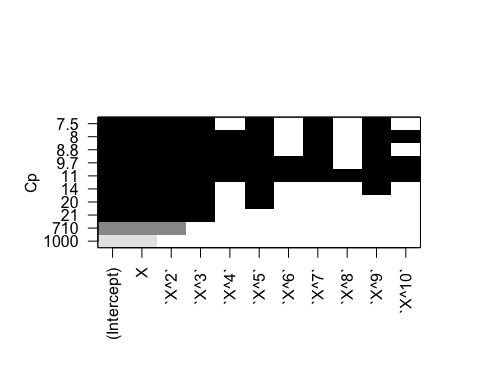
# d)

a2<-regsubsets(Y~., data = x\_data, method = "forward", nvmax = 10)  
mysummary(a2)

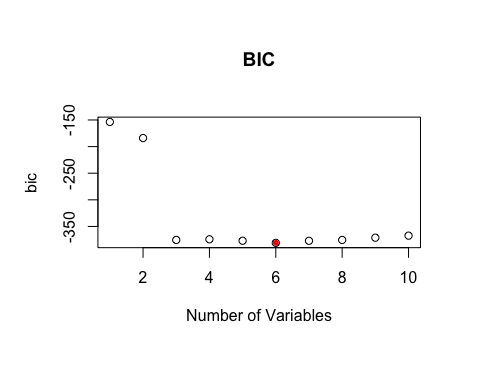
## Subset selection object  
## Call: regsubsets.formula(Y ~ ., data = x\_data, method = "forward",   
## nvmax = 10)  
## 10 Variables (and intercept)  
## Forced in Forced out  
## X FALSE FALSE  
## `X^2` FALSE FALSE  
## `X^3` FALSE FALSE  
## `X^4` FALSE FALSE  
## `X^5` FALSE FALSE  
## `X^6` FALSE FALSE  
## `X^7` FALSE FALSE  
## `X^8` FALSE FALSE  
## `X^9` FALSE FALSE  
## `X^10` FALSE FALSE  
## 1 subsets of each size up to 10  
## Selection Algorithm: forward  
## X `X^2` `X^3` `X^4` `X^5` `X^6` `X^7` `X^8` `X^9` `X^10`  
## 1 ( 1 ) "\*" " " " " " " " " " " " " " " " " " "   
## 2 ( 1 ) "\*" "\*" " " " " " " " " " " " " " " " "   
## 3 ( 1 ) "\*" "\*" "\*" " " " " " " " " " " " " " "   
## 4 ( 1 ) "\*" "\*" "\*" " " "\*" " " " " " " " " " "   
## 5 ( 1 ) "\*" "\*" "\*" " " "\*" " " " " " " "\*" " "   
## 6 ( 1 ) "\*" "\*" "\*" " " "\*" " " "\*" " " "\*" " "   
## 7 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " "\*" " " "\*" " "   
## 8 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " "\*" " " "\*" "\*"   
## 9 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" " " "\*" "\*"   
## 10 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## [1] "Cps: "  
## [1] 1043.992234 710.611218 21.318293 19.561749 13.822436  
## [6] 7.469641 8.763127 7.984529 9.697202 11.000000



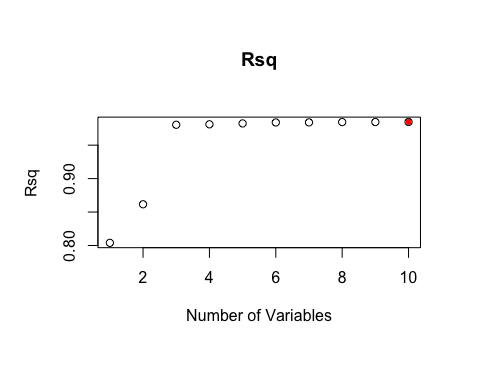
## [1] "Optimal model variable count:"  
## [1] 6



## [1] "BICs: "  
## [1] -153.7239 -183.9605 -375.3712 -374.1372 -376.8578 -380.8120 -376.9656  
## [8] -375.4016 -371.1162 -367.2914



## [1] "Optimal model variable count:"  
## [1] 6  
## [1] "R-squared: "  
## [1] 0.8039415 0.8616211 0.9805113 0.9811573 0.9824883 0.9839249 0.9840464  
## [8] 0.9845243 0.9845737 0.9846936

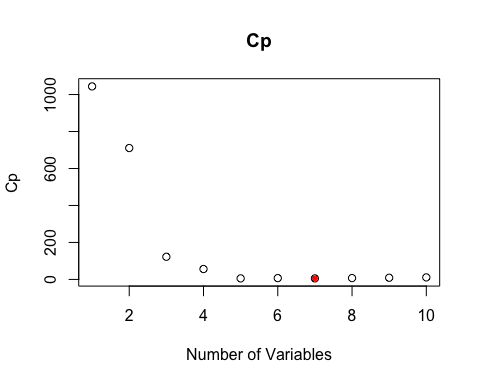


## [1] "Optimal model variable count:"  
## [1] 10

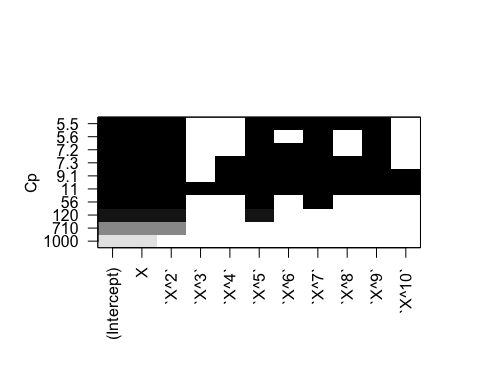
The 6th model is best according to Cp. The 6th model is best according to BIC. The 8th model is best according to R-Squared.

a3<-regsubsets(Y~., data = x\_data, method = "backward", nvmax = 10)  
mysummary(a3)

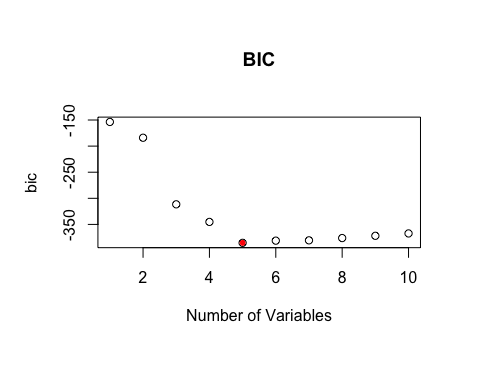
## Subset selection object  
## Call: regsubsets.formula(Y ~ ., data = x\_data, method = "backward",   
## nvmax = 10)  
## 10 Variables (and intercept)  
## Forced in Forced out  
## X FALSE FALSE  
## `X^2` FALSE FALSE  
## `X^3` FALSE FALSE  
## `X^4` FALSE FALSE  
## `X^5` FALSE FALSE  
## `X^6` FALSE FALSE  
## `X^7` FALSE FALSE  
## `X^8` FALSE FALSE  
## `X^9` FALSE FALSE  
## `X^10` FALSE FALSE  
## 1 subsets of each size up to 10  
## Selection Algorithm: backward  
## X `X^2` `X^3` `X^4` `X^5` `X^6` `X^7` `X^8` `X^9` `X^10`  
## 1 ( 1 ) "\*" " " " " " " " " " " " " " " " " " "   
## 2 ( 1 ) "\*" "\*" " " " " " " " " " " " " " " " "   
## 3 ( 1 ) "\*" "\*" " " " " "\*" " " " " " " " " " "   
## 4 ( 1 ) "\*" "\*" " " " " "\*" " " "\*" " " " " " "   
## 5 ( 1 ) "\*" "\*" " " " " "\*" " " "\*" " " "\*" " "   
## 6 ( 1 ) "\*" "\*" " " " " "\*" "\*" "\*" " " "\*" " "   
## 7 ( 1 ) "\*" "\*" " " " " "\*" "\*" "\*" "\*" "\*" " "   
## 8 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" " "   
## 9 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 10 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## [1] "Cps: "  
## [1] 1043.992234 710.611218 122.597547 56.309625 5.568909  
## [6] 7.181454 5.528537 7.337124 9.080219 11.000000



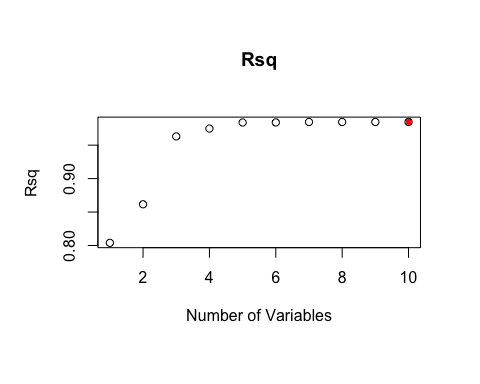
## [1] "Optimal model variable count:"  
## [1] 7



## [1] "BICs: "  
## [1] -153.7239 -183.9605 -311.5148 -345.2135 -385.3111 -381.1208 -380.5148  
## [8] -376.1237 -371.8065 -367.2914



## [1] "Optimal model variable count:"  
## [1] 5  
## [1] "R-squared: "  
## [1] 0.8039415 0.8616211 0.9630930 0.9748373 0.9839078 0.9839744 0.9846027  
## [8] 0.9846356 0.9846798 0.9846936



## [1] "Optimal model variable count:"  
## [1] 10

The 7th model is best according to Cp. The 5th model is best according to BIC. The 8th model is best according to R-Squared.

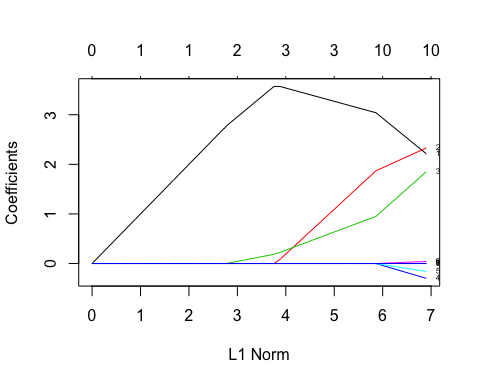
The model with the most amount of variables always has the larges R-squared value becasue, unlike Cp and BIC, there is no punishment for adding cofactors in the calculation for R-squared. That being said, it is not suprising that the best model according to the R-Squared value is consistently the model with 8 variables. That model is the same between the exaustive search and backward search, which is the model that includes every X except X^3 and X^10. For forward stepwise, that model is the model that includes every X except X^6 and X^8.

According to Cp, the best model found in the exaustive search was the model with 7 variables, which included every X except X^3, X^4, and X^10. For the forward stepwise it was the model includeing 6 variables (X to the 1,2,3,5,7, and 9th power). For the backward stepwise it was the same 7 variable model found from the exaustive search.

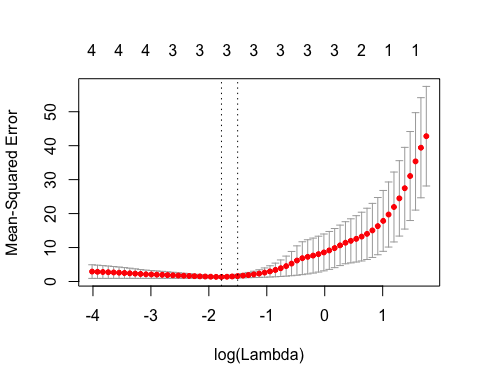
According to BIC, the best model found in the exaustive search was the model with 5 variables (X to the 1,2,5,7, and 9th power). The same 5 variable model was found in when backward stepwise was applied. For the forward stepwise, however, the 6 variable model with X to the 1,2,3,5,7, and 9th powers was the best.

# e)

#fit lasso reg. model  
x\_dataM = as.matrix(x\_data)  
grid = seq(0,10, length =100)  
lasso.mod=glmnet(x\_dataM[,-11],x\_dataM[ ,11],alpha=1, lambda = grid)  
plot(lasso.mod, label=TRUE)



#Cross-validation  
set.seed(42)  
#x\_dataf = as.data.frame(x\_data)   
cv.out=cv.glmnet(x\_dataM[,-11],x\_dataM[ ,11],alpha=1)  
plot(cv.out)



#find lambda corresponding to smallest mse  
bestlam =cv.out$lambda.min  
bestlam

## [1] 0.168265

#extract coeficients for model with this lambda  
coef(cv.out, s = "lambda.min")

## 11 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 1.1188429  
## X 3.0702398  
## X^2 1.7706815  
## X^3 0.9109737  
## X^4 .   
## X^5 .   
## X^6 .   
## X^7 .   
## X^8 .   
## X^9 .   
## X^10 .

The lambda that gives the smallest cross-validated mean-squared error is 0.168265. This lambda corresponds with the point on the cross-validation graph whose log(Lambda) = log(0.168265) = -0.77401, which is the point between the two calculated vertical dotted lines. The coefficient values can be found in the 11 x 1 spase matrix above. Notice that X, X^2, and X^3 are the only covariates with non-zero coefficients, and their coefficients are 3.07, 1.77, and 0.91, correspondingly. Remember our data was created using the real coefficient valeus of 3, 2, and 1. Thus, this model is fairly accurate in its assumptions. It overestimates the X coefficient, and underestimate both the X^2 and the X^3 coefficient. The fact that these particular covariates have non-zero coeficients for the optimal model is not suprising, since our data was generated using exclusively these three variables.

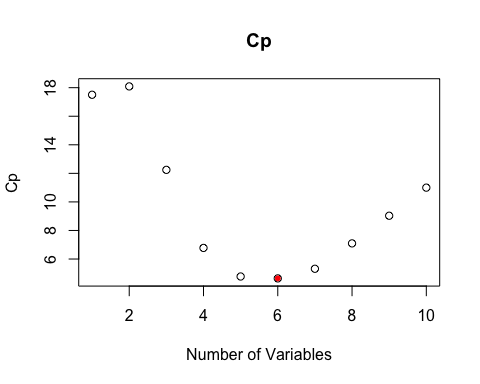
# Data Generation)

# B0=1 B7=2.5  
B\_2 = c(1,2.5)  
  
Y\_2 = rep(B\_2[1], 100)  
Y\_2 = Y\_2 + (B\_2[2]\*X^7) + e  
  
  
x\_data\_2 = data.frame(X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^10, Y\_2)  
colnames(x\_data) =c('X', 'X^2', 'X^3', 'X^4', 'X^5', 'X^6', 'X^7', 'X^8', 'X^9', 'X^10', 'Y\_2')

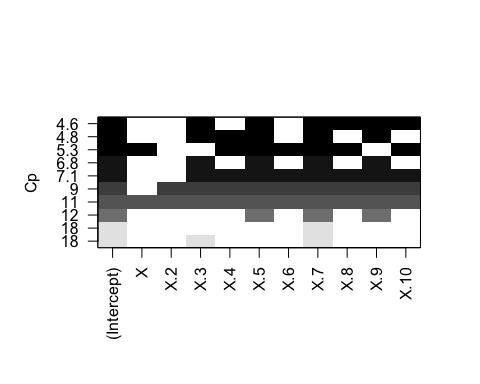
# Subset Selection)

#Exhaustive  
b<-regsubsets(Y\_2~., data = x\_data\_2, nvmax = 10)  
mysummary(b)

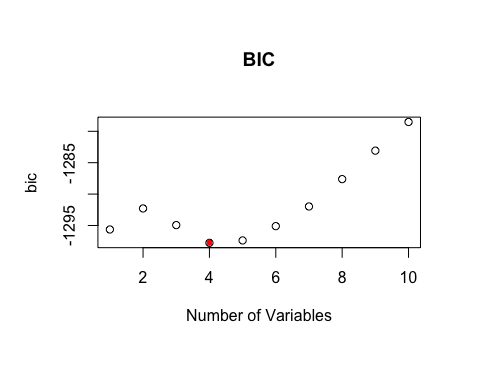
## Subset selection object  
## Call: regsubsets.formula(Y\_2 ~ ., data = x\_data\_2, nvmax = 10)  
## 10 Variables (and intercept)  
## Forced in Forced out  
## X FALSE FALSE  
## X.2 FALSE FALSE  
## X.3 FALSE FALSE  
## X.4 FALSE FALSE  
## X.5 FALSE FALSE  
## X.6 FALSE FALSE  
## X.7 FALSE FALSE  
## X.8 FALSE FALSE  
## X.9 FALSE FALSE  
## X.10 FALSE FALSE  
## 1 subsets of each size up to 10  
## Selection Algorithm: exhaustive  
## X X.2 X.3 X.4 X.5 X.6 X.7 X.8 X.9 X.10  
## 1 ( 1 ) " " " " " " " " " " " " "\*" " " " " " "   
## 2 ( 1 ) " " " " "\*" " " " " " " "\*" " " " " " "   
## 3 ( 1 ) " " " " " " " " "\*" " " "\*" " " "\*" " "   
## 4 ( 1 ) " " " " "\*" " " "\*" " " "\*" " " "\*" " "   
## 5 ( 1 ) " " " " "\*" "\*" "\*" " " "\*" " " "\*" " "   
## 6 ( 1 ) " " " " "\*" " " "\*" " " "\*" "\*" "\*" "\*"   
## 7 ( 1 ) "\*" " " " " "\*" "\*" "\*" "\*" "\*" " " "\*"   
## 8 ( 1 ) " " " " "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 9 ( 1 ) " " "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 10 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## [1] "Cps: "  
## [1] 17.503967 18.089401 12.245486 6.777278 4.776474 4.643057 5.317350  
## [8] 7.095679 9.031722 11.000000



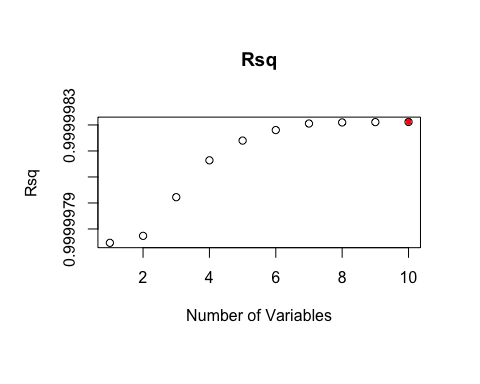
## [1] "Optimal model variable count:"  
## [1] 6



## [1] "BICs: "  
## [1] -1295.638 -1292.287 -1294.937 -1297.765 -1297.382 -1295.103 -1291.971  
## [8] -1287.615 -1283.081 -1278.512



## [1] "Optimal model variable count:"  
## [1] 4  
## [1] "R-squared: "  
## [1] 0.9999978 0.9999979 0.9999980 0.9999982 0.9999982 0.9999983 0.9999983  
## [8] 0.9999983 0.9999983 0.9999983



## [1] "Optimal model variable count:"  
## [1] 10

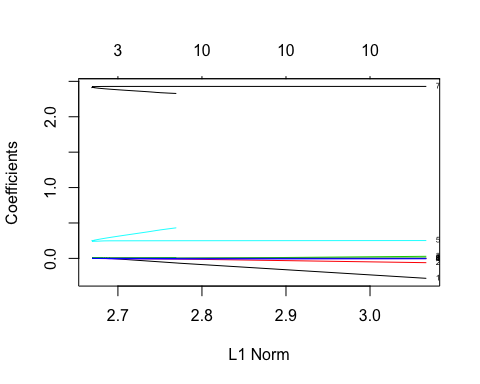
Performing exhaustive subset selection via regsubsets, we see that the optimal model according to Cps is the 6 cofactor model with X to the power of 3, 5, 7, 8, 9, and 10. For tihs model, the Cp is at a minimum of 4.643057.

According to the BIC measure the optimal model is the 4 cofactor model with X to the power of 3, 5, 7, and 9. For this model, the BIC is at a minimum of -1297.765.

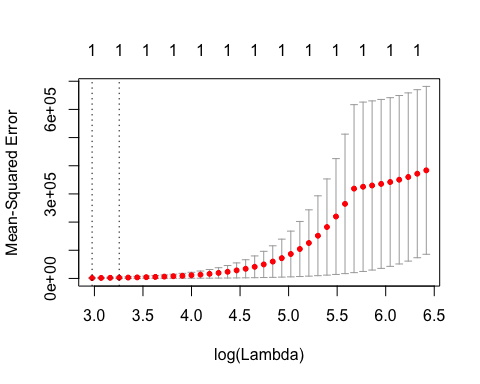
Similar to the previous problem, the optimal model according to the R-squared is unsuprisingly the model with the most amount of cofactors. Here, all models considered have a very large r-squared, since the one variable model sufficiently explains the response variable Y (since it was made with only one variable X^7).

# f)

#fit lasso reg. model  
x\_dataM\_2 = as.matrix(x\_data\_2)  
  
lasso.mod\_2=glmnet(x\_dataM\_2[,-11],x\_dataM\_2[ ,11],alpha=1, lambda = grid)  
plot(lasso.mod\_2, label=TRUE)



#Cross-validation  
set.seed(42)  
#x\_dataf\_2 = as.data.frame(x\_data\_2)   
cv.out\_2=cv.glmnet(x\_dataM\_2[,-11],x\_dataM\_2[ ,11],alpha=1)  
plot(cv.out\_2)



#find lambda corresponding to smallest mse  
bestlam\_2 =cv.out\_2$lambda.min  
bestlam\_2

## [1] 19.60987

#extract coeficients for model with this lambda  
coef(cv.out\_2, s = "lambda.min")

## 11 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -1.958884  
## X .   
## X.2 .   
## X.3 .   
## X.4 .   
## X.5 .   
## X.6 .   
## X.7 2.419947  
## X.8 .   
## X.9 .   
## X.10 .

The lambda that gives the smallest cross-validated mean-squared error is 19.60987. This lambda corresponds with the point on the cross-validation graph whose log(Lambda) = log(19.60987) = 1.2924, which is the point between the two calculated vertical dotted lines. The coefficient values can be found in the 11 x 1 spase matrix above. Notice that X^7 is the only covariate with non-zero coefficients, and it coefficient is 2.5. Remember our data was created using the real coefficient valeu of 2.5. Thus, this model is accurate in its assumptions, though it underestimates the X^2 coefficient. Again, the fact that this particular covariate has a non-zero coeficient in the optimal model is not suprising, since our data was generated using exclusively this X^7 variable multiplied by 2.5.

# a)

indexes = sample(1:nrow(uni), size=0.5\*nrow(uni))  
test = uni[indexes,]  
train = uni[-indexes,]

# b)

uniTrainingLM <- lm(Apps ~ ., train)  
summary(uniTrainingLM)

##   
## Call:  
## lm(formula = Apps ~ ., data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3249.2 -497.2 -82.7 368.1 6922.8   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -2.712e+02 6.109e+02 -0.444 0.657341   
## PrivateYes -7.848e+02 2.131e+02 -3.682 0.000265 \*\*\*  
## Accept 1.263e+00 7.158e-02 17.646 < 2e-16 \*\*\*  
## Enroll 7.895e-02 2.844e-01 0.278 0.781475   
## Top10perc 5.412e+01 7.620e+00 7.102 6.32e-12 \*\*\*  
## Top25perc -1.387e+01 6.431e+00 -2.158 0.031602 \*   
## F.Undergrad 2.522e-03 4.889e-02 0.052 0.958883   
## P.Undergrad 4.884e-03 5.655e-02 0.086 0.931222   
## Outstate -3.523e-02 2.886e-02 -1.221 0.222874   
## Room.Board 2.259e-01 7.154e-02 3.157 0.001723 \*\*   
## Books -2.342e-02 3.523e-01 -0.066 0.947022   
## Personal -7.133e-03 9.378e-02 -0.076 0.939410   
## PhD -5.904e+00 7.265e+00 -0.813 0.416918   
## Terminal -6.962e+00 8.137e+00 -0.856 0.392780   
## S.F.Ratio 1.459e+00 1.840e+01 0.079 0.936845   
## perc.alumni -1.103e+01 6.294e+00 -1.753 0.080418 .   
## Expend 4.928e-02 1.626e-02 3.031 0.002611 \*\*   
## Grad.Rate 9.541e+00 4.388e+00 2.175 0.030295 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1098 on 371 degrees of freedom  
## Multiple R-squared: 0.9216, Adjusted R-squared: 0.918   
## F-statistic: 256.6 on 17 and 371 DF, p-value: < 2.2e-16

#Calculate MSE on training set  
mean(residuals(uniTrainingLM)^2)

## [1] 1150171

#Calculate MSE on test set by calculating residuals  
mean(((test$Apps)-predict(uniTrainingLM,test))^2)

## [1] 1209637

The mean square error of the linear model is 1.209636610^{6}

# c)

#Convert data to format usable by glmnet package, split into appropriate test and training data sets  
x=model.matrix(Apps~.,uni)[,-1]  
xTest = x[indexes,]  
xTrain = x[-indexes,]  
  
y=uni$Apps  
yTest = uni[indexes,]$Apps  
yTrain = uni[-indexes,]$Apps

ridge.cv = cv.glmnet(xTrain,yTrain,alpha=0)  
lambda.cv = ridge.cv$lambda.min  
lambda.cv

## [1] 392.5473

fit.ridge = glmnet(xTrain,yTrain,alpha=0,lambda=lambda.cv)  
pred.ridge = predict(fit.ridge,newx=xTest)  
mean((yTest-pred.ridge)^2)

## [1] 2039168

The lambda minimized by cross-validation in the ridge regression model is 392.5473434, and the resulting test error is 2.03916810^{6}

# d)

lasso.cv = cv.glmnet(xTrain,yTrain,alpha=1)  
lambda.cv = lasso.cv$lambda.min  
lambda.cv

## [1] 11.17989

fit.lasso = glmnet(xTrain,yTrain,alpha=1,lambda=lambda.cv)  
pred.lasso = predict(fit.lasso,newx=xTest)  
mean((yTest-pred.lasso)^2)

## [1] 1227317

The lambda minimized by cross-validation in the LASSO model is 11.1798891, and the resulting test error is 1.227316710^{6}

# e)

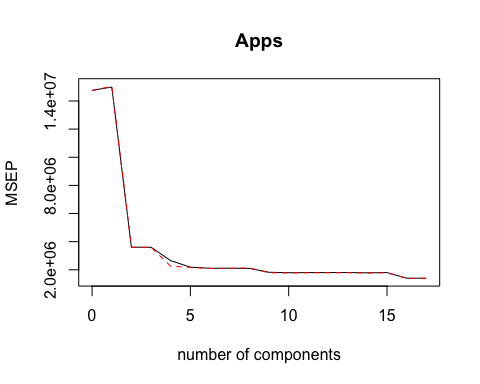
fit.pcr = pcr(Apps~.,data=train,scale=TRUE,validation="CV")  
summary(fit.pcr)

## Data: X dimension: 389 17   
## Y dimension: 389 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 3841 3872 1900 1897 1628 1480 1451  
## adjCV 3841 3878 1896 1895 1501 1474 1447  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 1453 1449 1350 1339 1342 1342 1343  
## adjCV 1450 1459 1336 1334 1337 1338 1339  
## 14 comps 15 comps 16 comps 17 comps  
## CV 1336 1342 1184 1185  
## adjCV 1331 1338 1178 1179  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
## X 31.371 57.65 65.46 71.25 76.87 81.56 85.11  
## Apps 1.626 76.68 76.87 86.56 86.72 87.01 87.01  
## 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps  
## X 88.03 90.81 93.33 95.41 97.19 98.17 98.90  
## Apps 87.21 89.06 89.20 89.20 89.23 89.24 89.37  
## 15 comps 16 comps 17 comps  
## X 99.48 99.86 100.00  
## Apps 89.40 92.10 92.16

pcr\_pred <- predict(fit.pcr, test)  
mean((pcr\_pred - test$Apps)^2)

## [1] 3623155

validationplot(fit.pcr,val.type="MSEP")



Appears the smallest Cross Validation error occurs when all components are included in the model (M = 17)

# f)

mean(((test$Apps)-predict(uniTrainingLM,test))^2) # Least Squares MSE

## [1] 1209637

mean((yTest-pred.ridge)^2) # Ridge MSE

## [1] 2039168

mean((yTest-pred.lasso)^2) # LASSO MSE

## [1] 1227317

mean((pcr\_pred - test$Apps)^2) # PCR MSE

## [1] 3623155

coef(uniTrainingLM)

## (Intercept) PrivateYes Accept Enroll Top10perc   
## -2.712196e+02 -7.848468e+02 1.263133e+00 7.895531e-02 5.411756e+01   
## Top25perc F.Undergrad P.Undergrad Outstate Room.Board   
## -1.387474e+01 2.522176e-03 4.884264e-03 -3.523260e-02 2.258838e-01   
## Books Personal PhD Terminal S.F.Ratio   
## -2.342423e-02 -7.132978e-03 -5.903813e+00 -6.961877e+00 1.458547e+00   
## perc.alumni Expend Grad.Rate   
## -1.103375e+01 4.928336e-02 9.540876e+00

coef(fit.ridge)

## 18 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) -1.310206e+03  
## PrivateYes -6.941615e+02  
## Accept 7.606190e-01  
## Enroll 7.650760e-01  
## Top10perc 2.987822e+01  
## Top25perc 2.016432e+00  
## F.Undergrad 9.180116e-02  
## P.Undergrad -1.187016e-03  
## Outstate 1.111003e-02  
## Room.Board 2.669129e-01  
## Books 2.647735e-02  
## Personal -6.344657e-02  
## PhD -9.017944e-01  
## Terminal -8.043214e+00  
## S.F.Ratio 4.032825e+00  
## perc.alumni -1.640348e+01  
## Expend 5.410772e-02  
## Grad.Rate 1.052146e+01

coef(fit.lasso)

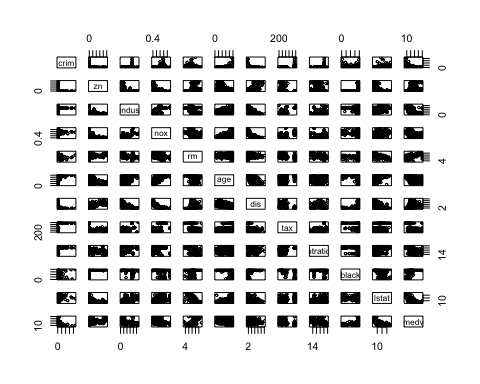
## 18 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) -3.849310e+02  
## PrivateYes -7.627745e+02  
## Accept 1.251083e+00  
## Enroll 9.984464e-02  
## Top10perc 4.803665e+01  
## Top25perc -8.839646e+00  
## F.Undergrad 2.877663e-03  
## P.Undergrad .   
## Outstate -2.213359e-02  
## Room.Board 2.001993e-01  
## Books .   
## Personal .   
## PhD -4.473273e+00  
## Terminal -6.889357e+00  
## S.F.Ratio .   
## perc.alumni -1.076730e+01  
## Expend 4.575306e-02  
## Grad.Rate 7.870912e+00

coef(fit.pcr)

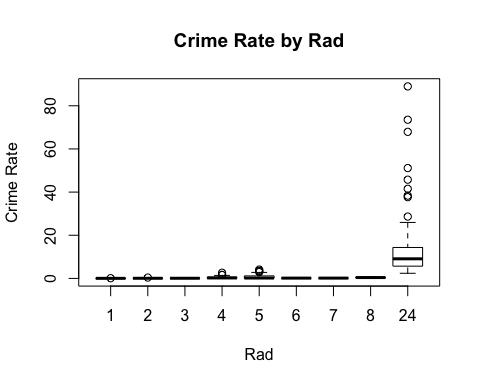
## , , 17 comps  
##   
## Apps  
## PrivateYes -351.932393  
## Accept 3164.692435  
## Enroll 78.990563  
## Top10perc 1001.943589  
## Top25perc -282.169693  
## F.Undergrad 13.136146  
## P.Undergrad 6.921946  
## Outstate -145.545916  
## Room.Board 259.997785  
## Books -3.995510  
## Personal -4.882939  
## PhD -98.764668  
## Terminal -102.093346  
## S.F.Ratio 5.981821  
## perc.alumni -138.326075  
## Expend 277.576700  
## Grad.Rate 164.551336

We can measure the accuracy of a model through the Mean Squared Error calculated using a separate test set. This measures the average squared residual and can assess how well a model fits a data set, in these cases test sets. The Least Squares Regression and the LASSO model both tend to have similar test errors, which are the lowest. The ridge regression model tends to have higher test error, with the PCR model having the worst test error on average. If these procedures were repeated for a different split between training and test sets, the trends usually hold but the values themselves change.

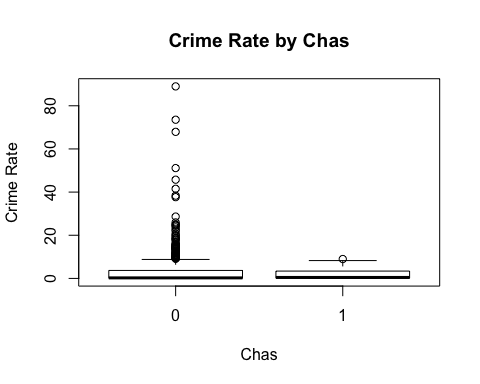
attach(Boston)  
  
  
  
#Initial Summary  
#head(Boston)  
#summary(Boston)  
  
#remove factor like variables  
Boston\_cont = subset(Boston, select=-c(rad, chas))  
  
#Pairplot  
pairs(Boston\_cont, cex = .5)



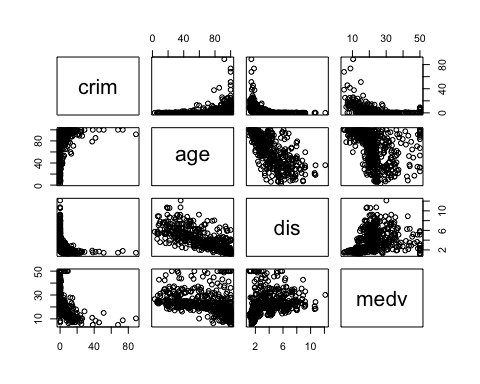
#Boxplot for factor var. rad  
boxplot(crim~rad, main="Crime Rate by Rad",   
xlab="Rad", ylab="Crime Rate")



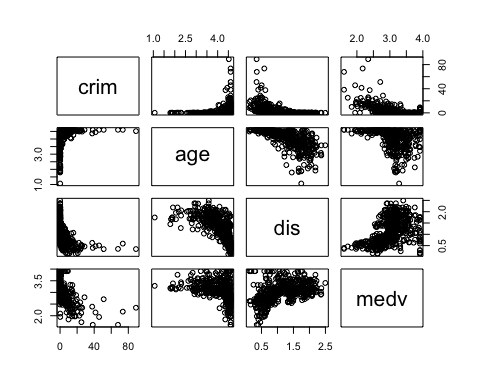
#Boxplot for factor var. chas  
boxplot(crim~chas, main="Crime Rate by Chas",   
xlab="Chas", ylab="Crime Rate")



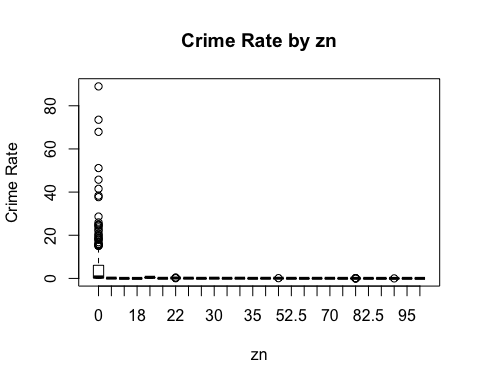
Boston\_to\_trans = subset(Boston, select=c(crim,age, dis, medv))  
  
#before transform  
pairs(Boston\_to\_trans)



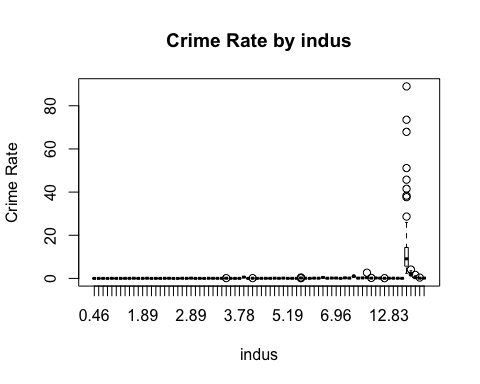
#transform  
Boston\_to\_trans[,2:4] = log(Boston\_to\_trans[,2:4])  
  
#after transform  
pairs(Boston\_to\_trans)



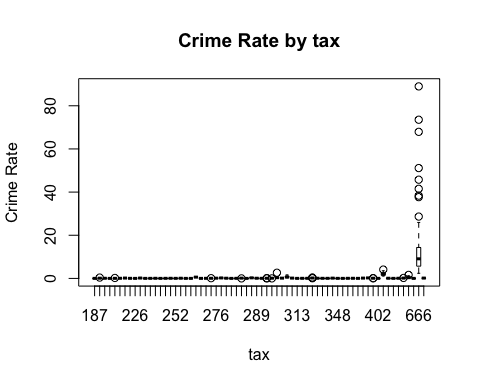
#zn indus tax and ptratio all seem to only have high crime rates at one value. See which value that is   
boxplot(crim~zn, main="Crime Rate by zn", xlab="zn", ylab="Crime Rate")



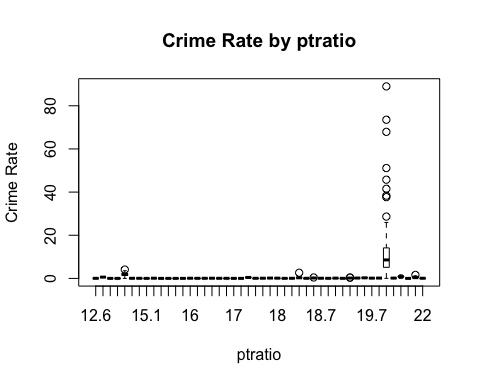
boxplot(crim~indus, main="Crime Rate by indus", xlab="indus", ylab="Crime Rate")



boxplot(crim~tax, main="Crime Rate by tax", xlab="tax", ylab="Crime Rate")



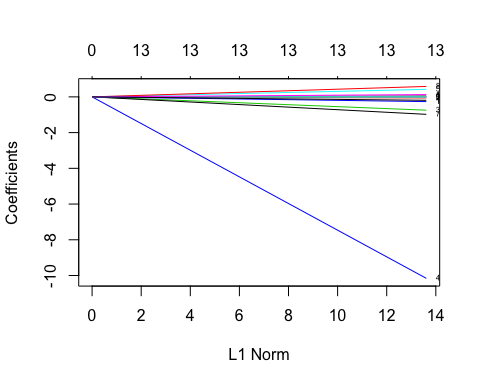
boxplot(crim~ptratio, main="Crime Rate by ptratio", xlab="ptratio", ylab="Crime Rate")



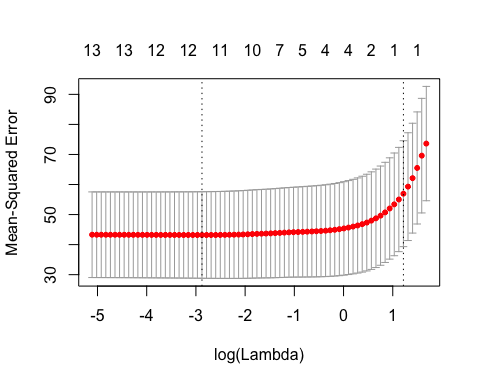
#zn = 0 , indus = 18.10 ,

Looking at the pairs plot, zn, indus, age, dis, tax, and ptratio seem to be the strongest possible contenders for covariates to be included in our model. This is of course based on their correlation with the crim variable (the top row). For the two factor covariates rad and chas, rad value of 24 seems to be correlated with larger crime rate and while the mean crim rate of chas=0 points is low, any point with a large crime rate had a chas=0. In orther words, very little of the chas=1 points had a large crime rate.

#fit lasso reg. model  
Boston\_M = as.matrix(Boston)  
lambdas=seq(1e-3,1e3,length=100)  
lasso.mod\_3=glmnet(Boston\_M[,-1],Boston\_M[ ,1],alpha=1, lambda = lambdas)  
plot(lasso.mod\_3, label=TRUE)



#Cross-validation  
set.seed(42)  
#Boston\_df = as.data.frame(Boston\_df)   
cv.out\_3=cv.glmnet(Boston\_M[,-1],Boston\_M[ ,1],alpha=1)  
plot(cv.out\_3)



#find lambda corresponding to smallest mse  
bestlam =cv.out\_3$lambda.min  
bestlam

## [1] 0.05630926

#extract coeficients for model with this lambda  
coef(cv.out\_3, s = "lambda.min")

## 14 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 12.319178096  
## zn 0.035726832  
## indus -0.068876055  
## chas -0.577832639  
## nox -6.631559478  
## rm 0.208676938  
## age .   
## dis -0.768388825  
## rad 0.512333871  
## tax .   
## ptratio -0.179631375  
## black -0.007551172  
## lstat 0.124630014  
## medv -0.154550130

x = Boston\_M[,-1]  
y=Boston\_M[ ,1]  
finalModel1=glmnet(x,y,alpha=1, lambda = 0.05630926)  
#plotres(finalModel1, which=1)  
  
  
pred.lasso = predict(finalModel1,newx=x)  
mean((y-pred.lasso)^2)

## [1] 40.48466

summary(finalModel1)

## Length Class Mode   
## a0 1 -none- numeric  
## beta 13 dgCMatrix S4   
## df 1 -none- numeric  
## dim 2 -none- numeric  
## lambda 1 -none- numeric  
## dev.ratio 1 -none- numeric  
## nulldev 1 -none- numeric  
## npasses 1 -none- numeric  
## jerr 1 -none- numeric  
## offset 1 -none- logical  
## call 5 -none- call   
## nobs 1 -none- numeric

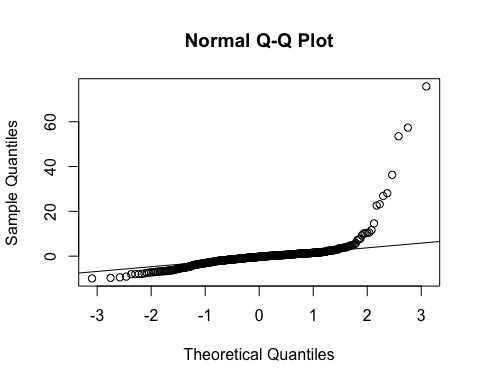
fits <- fitted(finalModel1)  
## calculate the deviance residuals  
resids <- (pred.lasso-y)  
fits

## NULL

plot(pred.lasso, resids, main = "Model Resids")  
abline(h=0, lty=2)



#QQ to see if residuals follow normal  
qqnorm(y-pred.lasso)  
qqline(y-pred.lasso)



We use cross validation via the 'cv.glmnet' funciton in order to validate our model selection and paremeter (lambda) setting. The minimizing lambda for this lasso regression is 0.0563092. This lambda corresponds with the point on the cross-validation graph whose log(Lambda) = log(0.0563092.) = -1.24942, which is the point between the two calculated vertical dotted lines. The covariates determined not useful and thus whose coefficients are set to zero in the model are age and tax. While these two variables looked as though they may have been correlated with crime rate, they could have been left out of the model due to multicollinearity. We validated our model using cross validation and considered a large amount of lambdas as values. The optimal lambda fell comfortably in between the max and min of all the lambas considered.

The residuals are not optmial, but they do have decent amounts both above and below the y=0 line. They also, for the majority of the middle instances, follow the qq line in the last plot. They only stray from it in the early and the late theoretical quantiles.