Lab 4: One last time through the hot New Jersey night. . .

Alice Chang

Load/Clean Data

```
library(MASS)
library(glmnet)
```

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

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

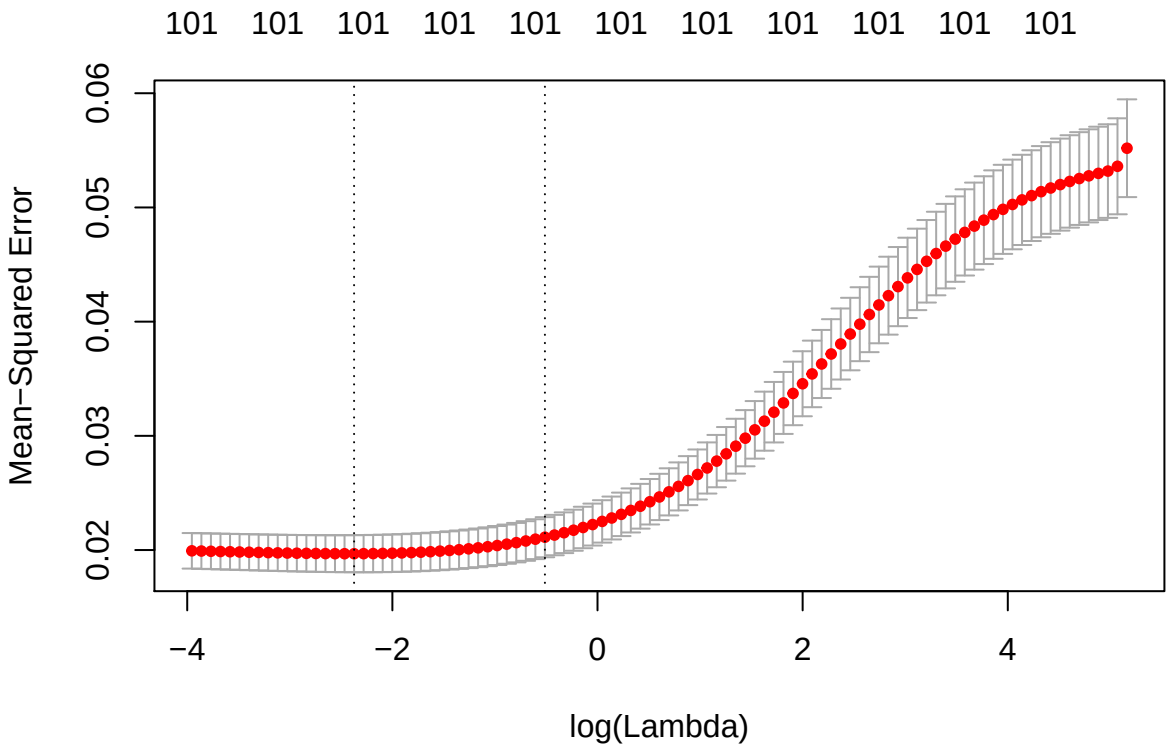
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
d <- read.csv("http://andrewpbray.github.io/data/crime-train.csv")
d_c <- d %>%
  select_if(function(col) !any(col == "?")) %>%
   select(-c("communityname"))
```

Ridge Regression, Lasso, and "Violent Crime" Dataset

```
#RFit ridge regression model
X <- model.matrix(ViolentCrimesPerPop ~ ., data = d_c)[, -1]
Y <- d_c$ViolentCrimesPerPop
grid <- seq(from = 1e4, to = 1e-2, length.out = 100)
rm1 <- glmnet(x = X, y = Y, alpha = 0,
lambda = grid, standardize = TRUE)
#find best lambda for ridge regrssion
set.seed(1)
cv.out = cv.glmnet(X, Y, alpha = 0)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min
bestlam
```

```
## [1] 0.09316328
```

```
#get training MSE for ridge
ridge.pred = predict(rm1, s = bestlam, X)
MSE <- mean((ridge.pred-Y)^2)
MSE
```

```
## [1] 0.01626048
```

```
#Explore coefficient estimates of fitted ridge regression model
dim(coef(rm1))
```

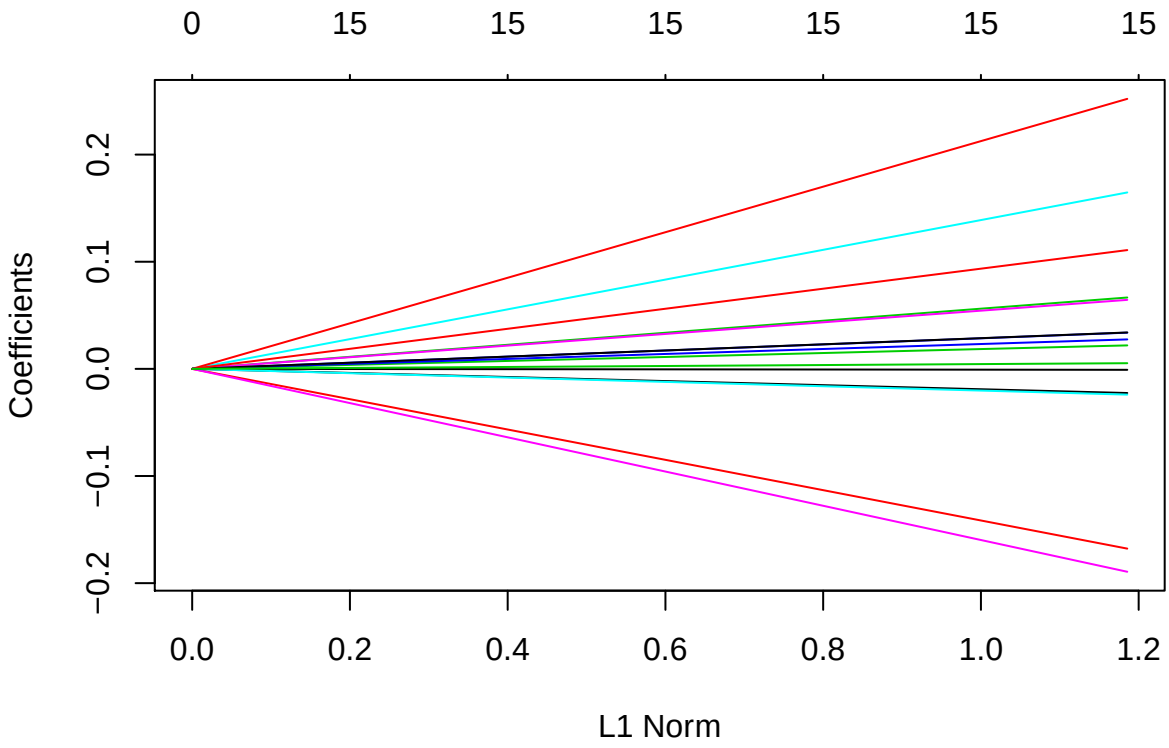
```
## [1] 102 100
```

```
out = glmnet(X, Y, alpha = 0)
predict(out, type = "coefficients", s = bestlam) [1:102]
```

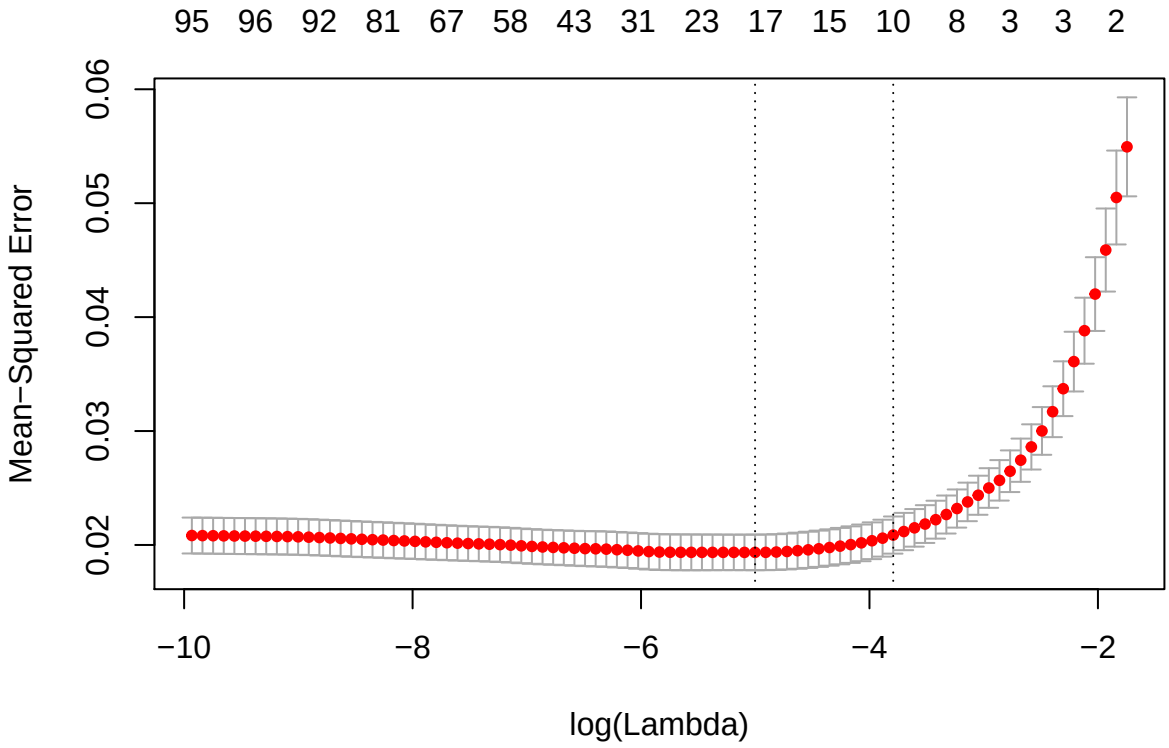
```
##   [1]  0.2926251258 -0.0009840049 -0.0089532785  0.0107257573  0.0881313801
##   [6] -0.0809924872 -0.0097613247  0.0249184099 -0.0167323578 -0.0424777337
##  [11] -0.0210728677 -0.0037276253  0.0045345663  0.0288314914  0.0111580638
##  [16] -0.0083971778 -0.0021684949 -0.0410728874  0.0158726018  0.0060979436
##  [21] -0.0042782298  0.0141126169  0.0036881350  0.0117392882 -0.0195762981
##  [26] -0.0369458250  0.0126426091  0.0209890264  0.0061716125  0.0181549379
##  [31] -0.0064758179  0.0129587917  0.0379860807 -0.0009452666  0.0029978888
##  [36] -0.0016885610 -0.0032285623  0.0002300394  0.0128756662 -0.0171153730
##  [41]  0.0741235554  0.0335372410  0.0297143686  0.0474480627  0.0285925238
##  [46] -0.0575675526 -0.0719363169 -0.0584535116 -0.0469717896  0.0040757124
##  [51] -0.0287398275  0.0498169124  0.1043712318 -0.0360273550  0.0013725887
##  [56] -0.0218248799  0.0401878127  0.0164555386 -0.0005306383 -0.0013544761
##  [61]  0.0100202759  0.0134804599  0.0043867696 -0.0025103805  0.0229989489
```

```
##  [66]  0.0107137952  0.0189613676 -0.0062389567  0.0134601558 -0.0108334888
##  [71]  0.0345685665  0.0285771521 -0.0098681167  0.0218983869 -0.0519022728
##  [76]  0.0037365717  0.0655494867 -0.0073943496 -0.0026126843  0.0078845140
##  [81] -0.0062504949 -0.0123998642 -0.0046269971 -0.0017396240 -0.0270369417
##  [86]  0.0067894553  0.0151727643  0.0223585552  0.0386952467 -0.0200876804
##  [91] -0.0371127662  0.0743617819  0.0980715729  0.0040268340 -0.0121838366
##  [96]  0.0029776425  0.0258272553  0.0011568491  0.0075609826 -0.0021111548
## [101]  0.0115852775  0.0248638884
```

```r
#Fit LASSO
lasso.mod <- glmnet(x = X, y = Y, alpha = 1,
lambda = grid, standardize = TRUE)
plot(lasso.mod)
```




```r
#find best lambda for LASSO
set.seed(1)
cv.out1 = cv.glmnet(X, Y, alpha = 1)
plot(cv.out1)
```




```
bestlam1 = cv.out1$lambda.min
bestlam1
```

```
## [1] 0.006727142
```

```
#Get training MSE for LASSO
lasso.pred = predict(lasso.mod, s = bestlam1, X)
MSE_lasso = mean ((lasso.pred - Y)^2)
MSE_lasso
```

```
## [1] 0.01828638
```

```
#Explore coefficient estimates of LASSO and selected variables
out = glmnet(X, Y, alpha = 1, lambda = grid)
lasso.coef = predict(out, type = "coefficients", s = bestlam1)[1:102,]
lasso.coef[lasso.coef != 0]
```

```
##          (Intercept)                state        racePctWhite
##        0.3375478525       -0.0008941009       -0.1677564554
##            pctUrban        PctNotHSGrad      MalePctDivorce
##        0.0219176399        0.0338060595        0.1646460132
##         PctKids2Par          PctWorkMom             PctIlleg
##       -0.1893645798       -0.0225404877        0.2520075851
##    PctPersDenseHous          HousVacant        PctHousOccup
##        0.0665861854        0.0274969788       -0.0240420801
##    PctVacantBoarded       NumInShelters           NumStreet
##        0.0644010940        0.0339653803        0.1108561645
## LemasPctOfficDrugUn
##        0.0052804459
```

1. LASSO selected 15 variables.

2. a) Using an optimal λ of 0.09316328, the training value for ridge regression is 0.01626048

b) Using an optimal λ of 0.006727142, the training MSE for LASSO is 0.01828638.

3. The training MSE for LASSO is slightly higher than the training MSE for ridge regression. The better performance of ridge regression may be due to the the fact that we are analyzing a training dataset with many predictors (102), all with coefficients of approximately similar sizes. As shown above, in the fitting of the ridge regression model on the training dataset, all of the coefficients are small and close to zero.

Problem Set 2

2.a) iii. Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

b) iii. Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

3. Lasso Regression

a) iv. Steadily decreases: As s increases from 0, there is less contraint and greater flexibility in the model. As βs increase from 0 towards their least squares estimates, the training error, RSS, should generally continue to decrease from its initial high value as well.

b) ii. Decreases initially, and then eventually starts increasing in a U shape: As s increases from 0, and the βs in the model increase from 0, the greater flexibility of the model may produce less test error in the RSS. However, over time, as the βs in the model approach their least squares estimates, the model will begin to overfit

c) iii. Steadily increases: Variances generally increases with greater model flexibility as moe βs are incorporated and increased.

d) iv. Steadily decreases: Bias generally decreases with greater model flexibility as more βs are incorporated and increased.

e) v. Remain constant: Irreducible error is independent of model selection.

4. Ridge Regression

a) iii. Steadily increases: As λ increases from 0, there is greater constraint and heavier penalty in the model. Subseuqently, the training error, RSS, should generally continue to increase as well as βs are reduced towards zero from their OLS estimates and begin to fit the training data less accurately.

b) ii. Decreases initially, and then eventually starts increasing in a U shape: As λ increases from 0, the βs in the model will shrink from their least squares estimates towards zero, generating a decrease in overfitting as well as the test error, RSS. However, over time, the model may become too simple, subsquently causing an eventual inrease in test error.

c) iv. Steadily decreases: With less flexibility and more constraints to the model, variance will decrease. When λ is 0, βs correspond to the least square estimates that reflect the high variance of the training data. Consequently, as βs shrink towards zero, variance decreases as well.

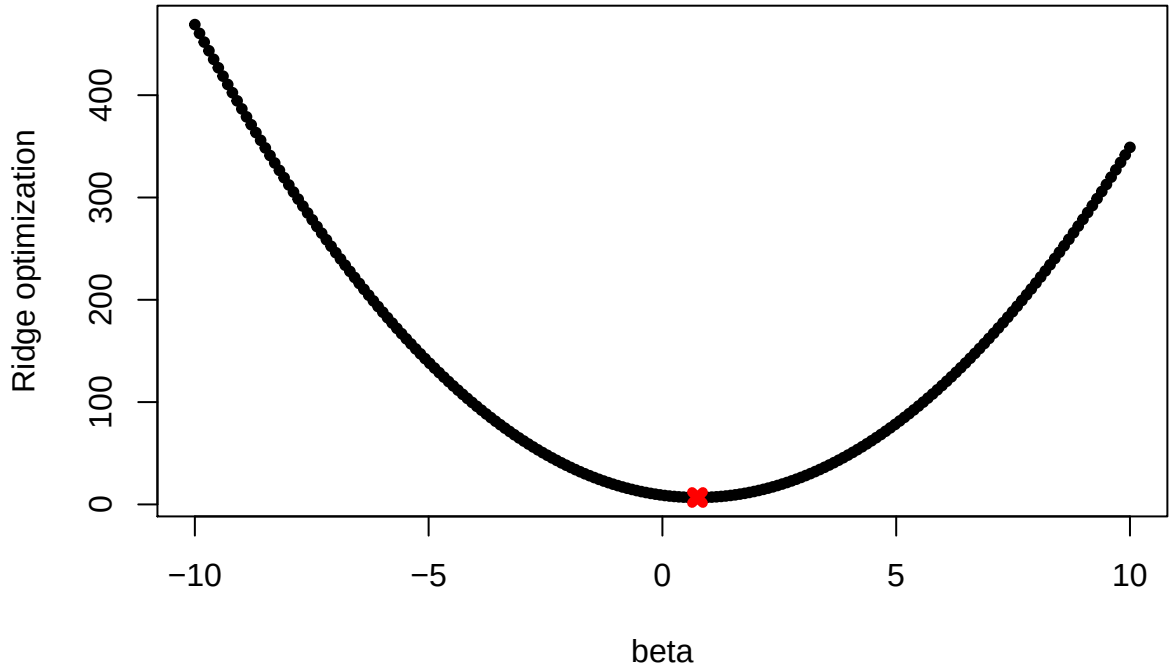
d) iii. Steadily increases: Conversely, bias generally increases with less model flexibility as βs are reduced from their least square estimates – which carry minimal bias – towards zero.

e) v. Remain constant: Irreducible error remains the same regardless of maodel selection.

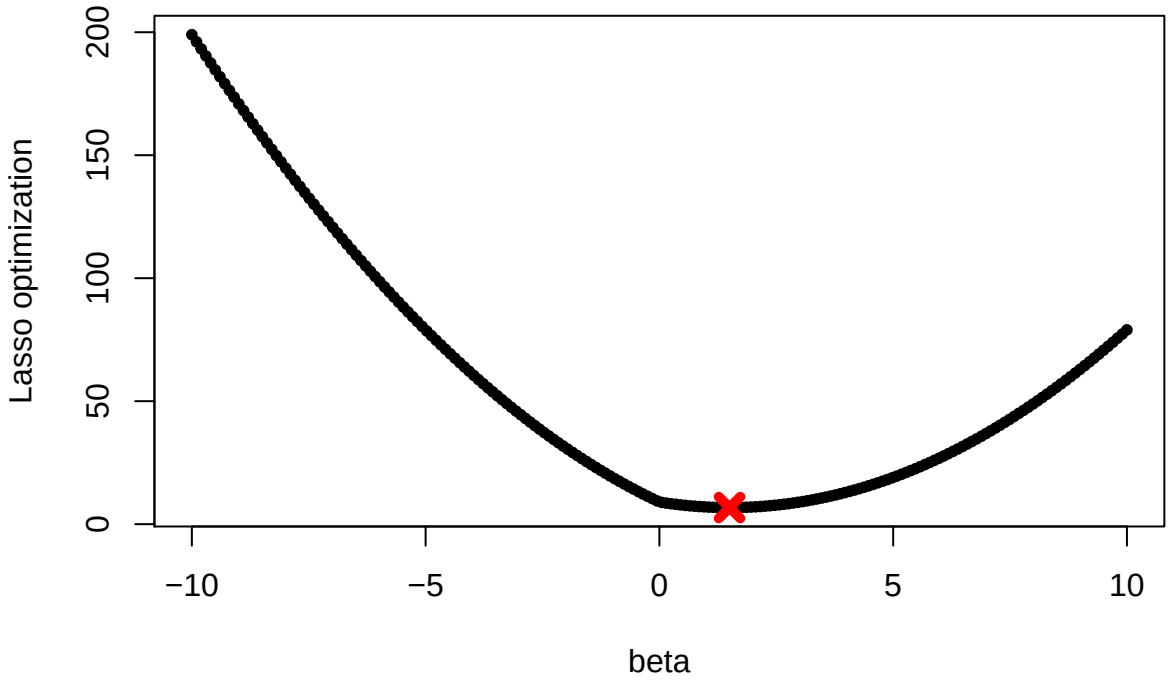
6.

```
#Part a)
y = 3
lambda = 3
betas = seq( from=-10, to=+10, length.out=200 )
```

```r
func = (y - betas)^2 + lambda * betas^2
plot(betas, func, pch = 20, xlab = "beta", ylab = "Ridge optimization")
#plot point
pt.beta = y/(1 + lambda)  #show that this is true
y_opt = (y - pt.beta)^2 + lambda * pt.beta^2
points(pt.beta, y_opt, col = "red", pch = 4, lwd = 5, cex = pt.beta)
```




```r
#Part b)
y1 = 3
lambda1 = 3
betas1 = seq( from=-10, to=+10, length.out=200 )
func1 = (y1-betas1)^2 + lambda1 * abs(betas1)
plot(betas1, func1, pch = 20, xlab = "beta", ylab = "Lasso optimization")
#plot point
pt.beta1 = y1 - lambda1/2  #show that this is true since y1 > lambda1/2
y_opt1 = (y-pt.beta1)^2 + lambda1 * abs(pt.beta1)
points(pt.beta1, y_opt1, col = "red", pch = 4, lwd = 5, cex = pt.beta1)
```


Lasso optimization
0
50
100
150
200
−10
−5
0
5
10
beta