Sparsity and Lasso

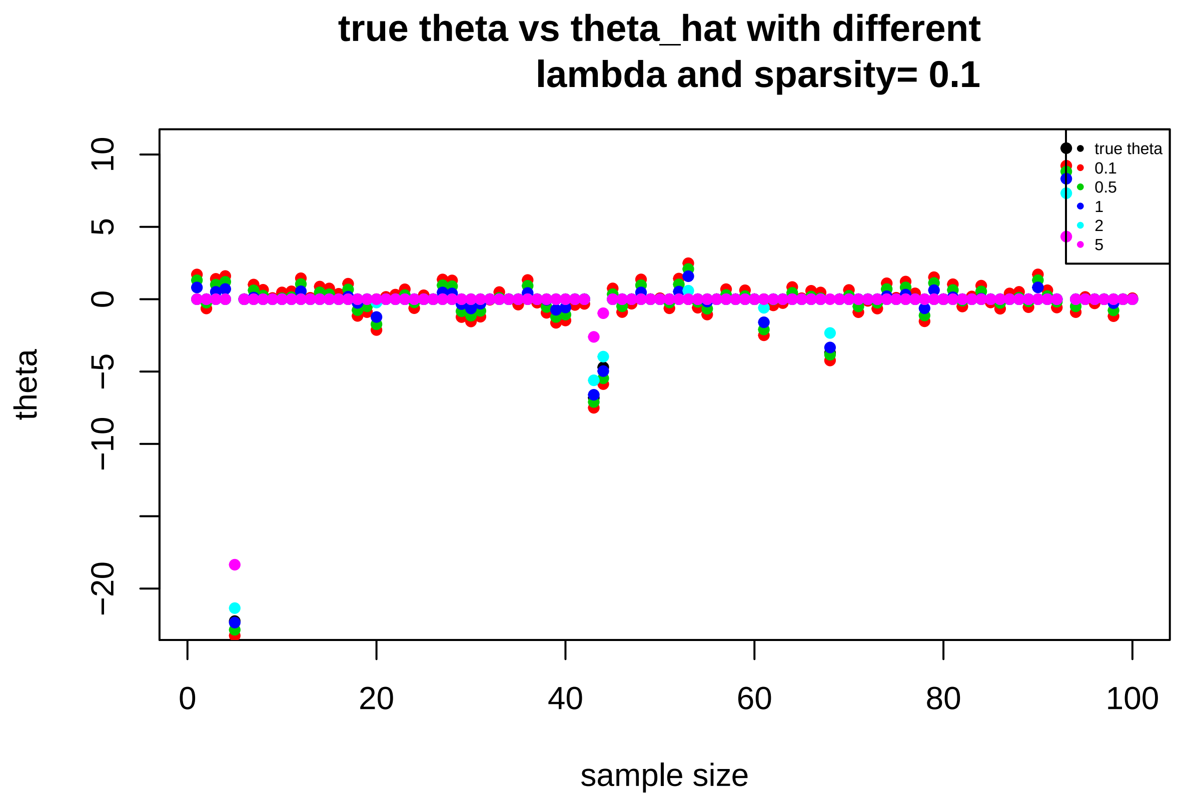
Su Chen

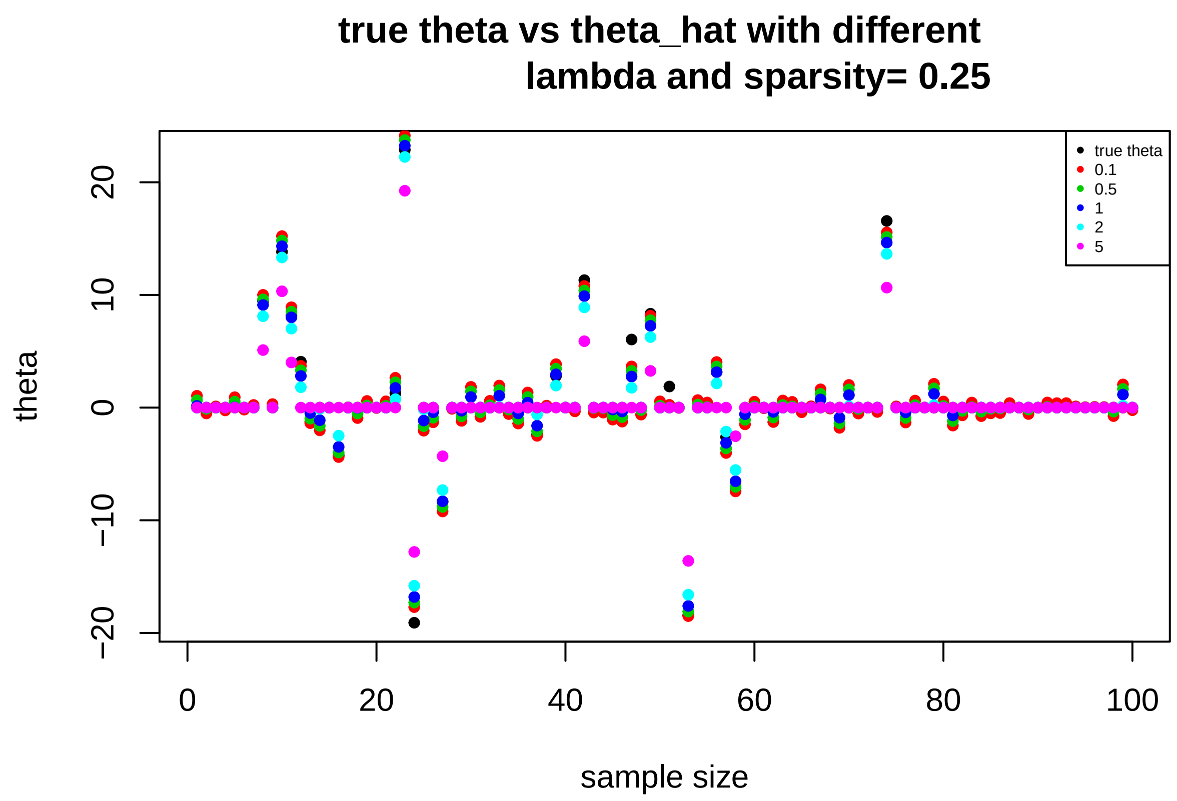
October 10, 2016

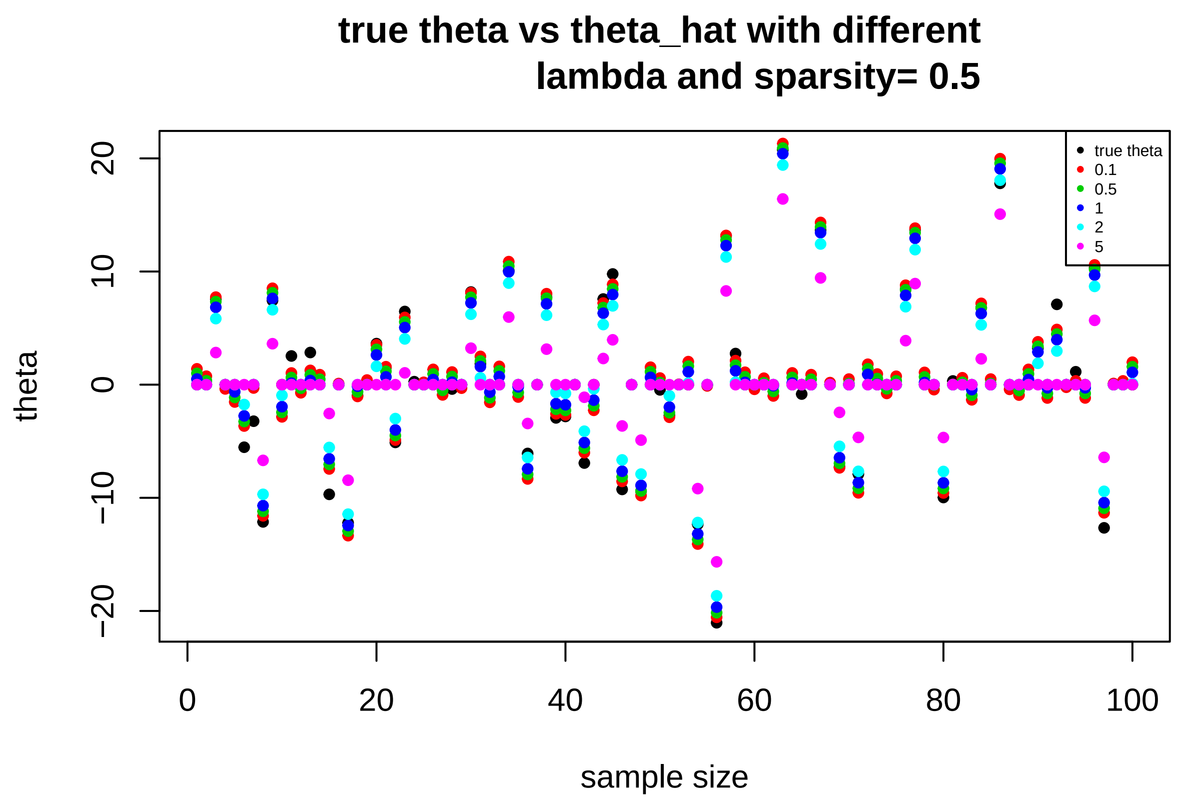
Penalized likelihood and soft thresholding Part B

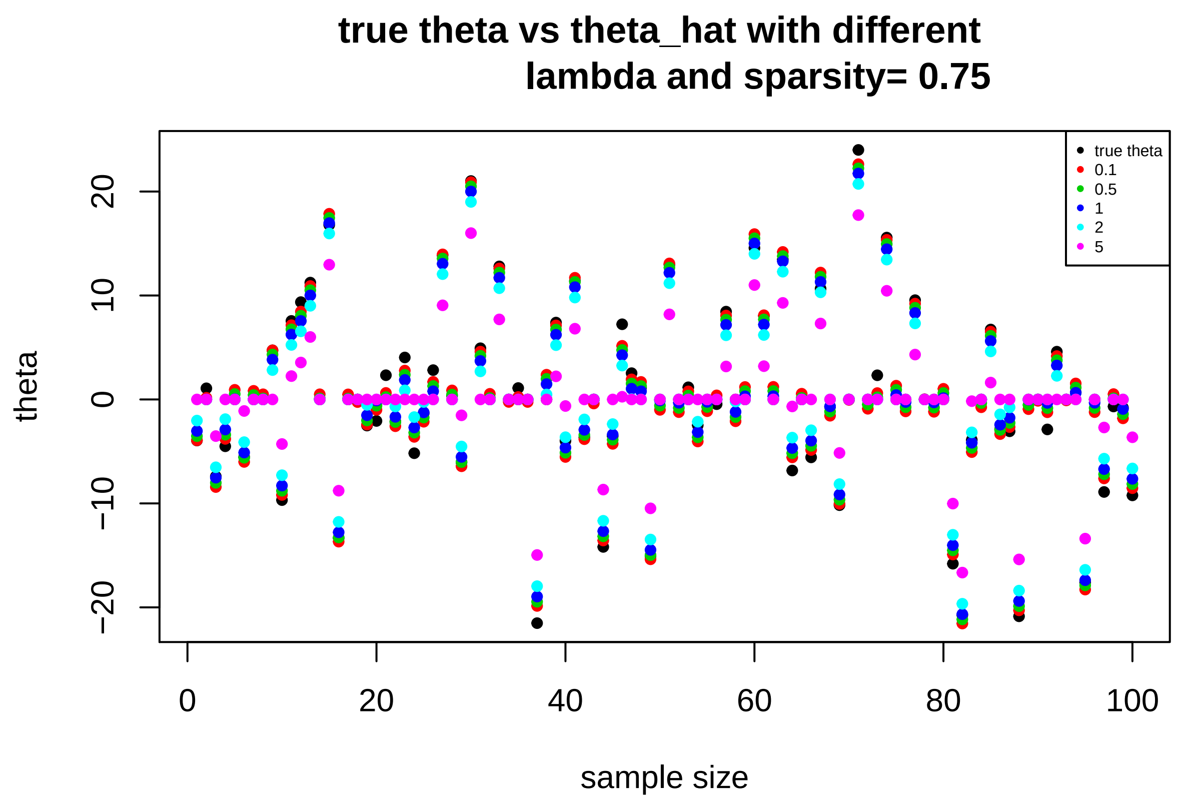
```
### function to sample normal data based on different level of sparsity
simu_data = function(n, sparsity, mu, s)
{
  ### generate Theta
  theta = rbinom(n, 1, prob = sparsity)*rnorm(n, mean=mu, sd=s)

  ### sample Y
  y = rep(0, n)
  for (i in 1:n)
  {
    y[i] = rnorm(1, mean=theta[i], sd=1) #just use sd = 1 here
  }

  return (list(theta, y))
}


### function to calculate MSE
MSE = function(y, theta, lambda)
{
  return ( mean((theta_hat(y,lambda) - theta)^2) )
}


### calculate Theta_Hat
theta_hat = function(y,lambda)
{
  z = abs(y) - lambda
  return (sign(y)*z*(z > 0)) # this is max(z, 0) element wise
}


n=100
sparsity = c(0.1, 0.25, 0.5, 0.75, 1)
p = length(sparsity)

y = matrix(0, p, n)
theta = matrix(0, p, n)

lambda_trial = c(0.1, 0.5, 1, 2, 5)
l = length(lambda_trial)


for (k in 1:p)
```

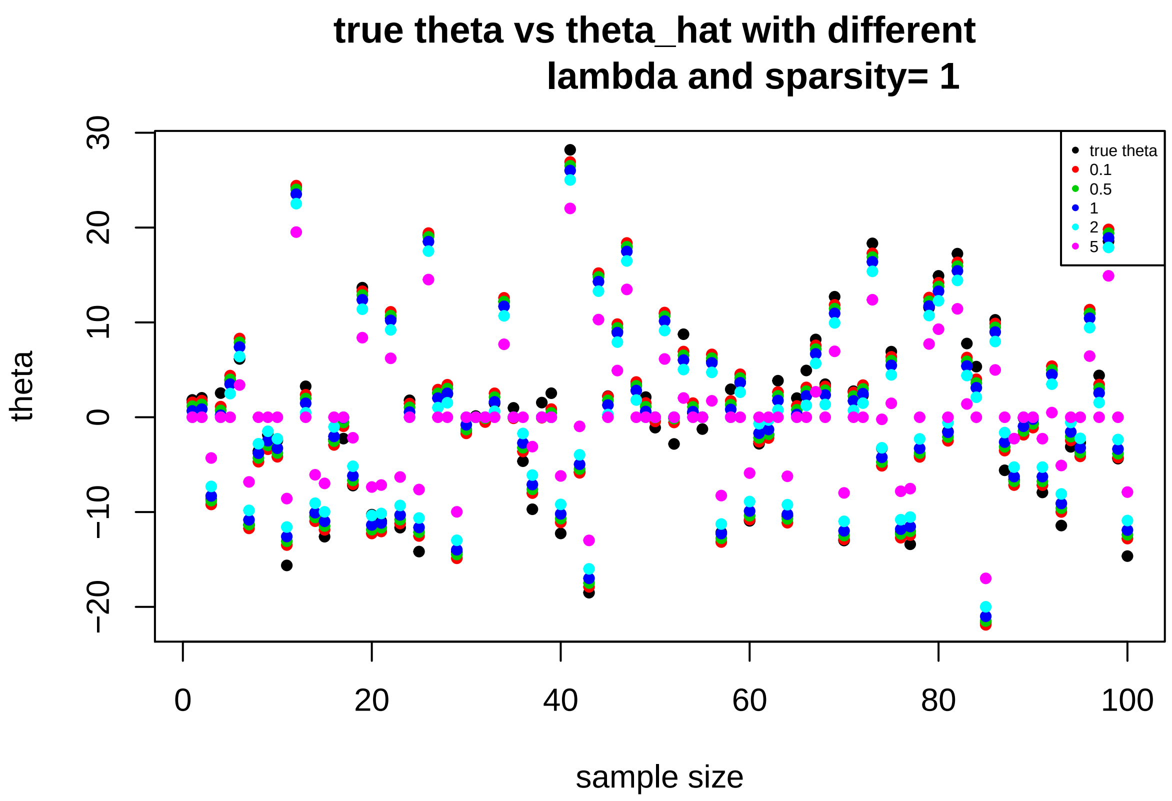
```
{
  simulation = simu_data(n, sparsity=sparsity[k], mu=0, s=10)
  theta[k,] = simulation[[1]]
  y[k,] = simulation[[2]]

  theta_hat_trial = matrix(0, nrow=l, n)

  for(i in 1:l)
  {
    for (j in 1:n)
      theta_hat_trial[i,j] = theta_hat(y=y[k,j], lambda=lambda_trial[i])
  }

  ### plot
  plot(x=1:n, y=theta[k,], type="p", xlab="sample size", ylab="theta",
       pch=20, main=paste("true theta vs theta_hat with different
                    lambda and sparsity=", sparsity[k]) )
  for (i in 1:l)
  {
    points(x=1:n, y=theta_hat_trial[i,], col=i+1, pch=20)
  }
  legend("topright", cex = .5, c("true theta",lambda_trial),
         col = c(1:(l+1)), pch = rep(20, l+1))
}
```




true theta vs theta_hat with different
lambda and sparsity= 0.25
theta
sample size
20
10
0
−10
−20
0
20
40
60
80
100
true theta
0.1
0.5
1
2
5


true theta vs theta_hat with different
lambda and sparsity= 0.5
theta
sample size
true theta
0.1
0.5
1
2
5
20
10
0
−10
−20
0
20
40
60
80
100

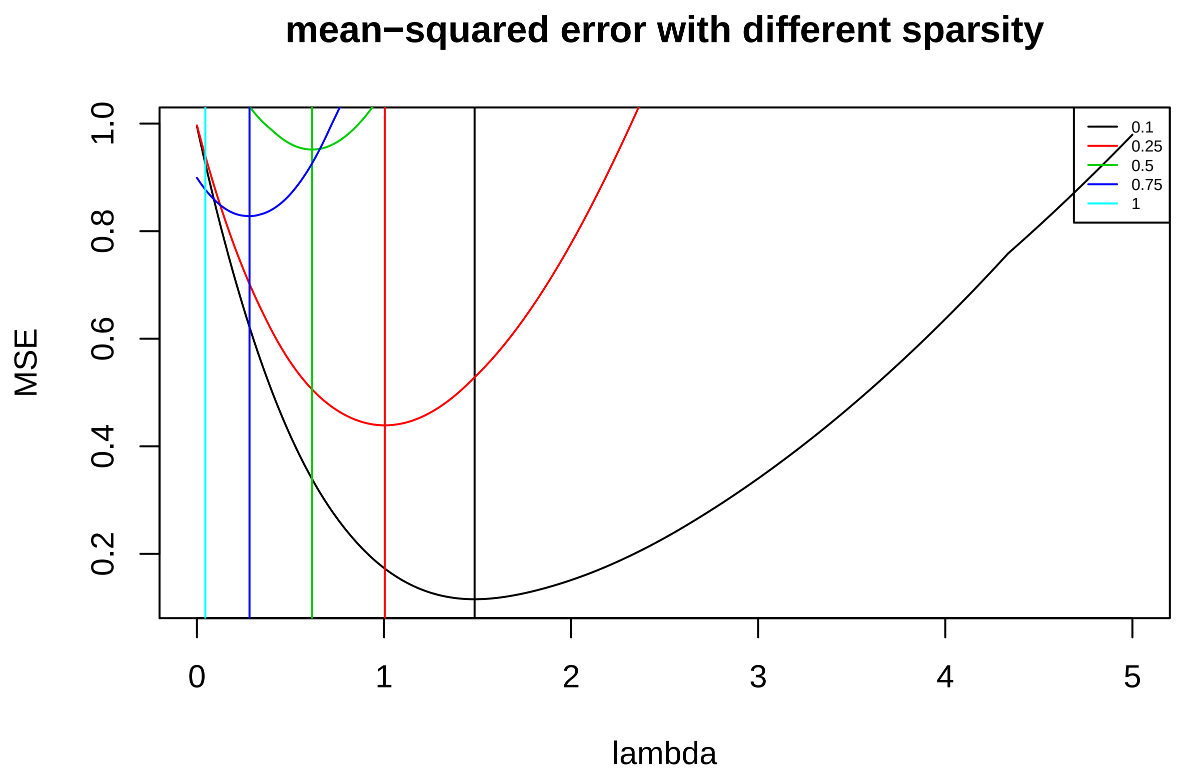

true theta vs theta_hat with different
lambda and sparsity= 0.75
theta
sample size
20
10
0
−10
−20
0
20
40
60
80
100
true theta
0.1
0.5
1
2
5



From these plots we can see as lambda increases, the soft threshold is pushing more and more theta to 0, and this shrinking effects also increases as sparsity level increases.

```
### plot MSE
k = 1
lambda_plot = seq(0,5,0.001)
q = length(lambda_plot)
MSE_plot = rep(0, q)
for (i in 1:q)
{
  MSE_plot[i] = MSE(y[k,], theta[k,], lambda_plot[i])
}

plot(x = lambda_plot, y = MSE_plot, type = "l", xlab = "lambda", ylab = "MSE",
     main = "mean-squared error with different sparsity", col = k)
abline(v = lambda_plot[MSE_plot == min(MSE_plot)], col = k)

for (k in 2:p)
{
  MSE_plot = rep(0, q)
  for (i in 1:q)
    {
        MSE_plot[i] = MSE(y[k,], theta[k,], lambda_plot[i])
    }

  lines(x = lambda_plot, y = MSE_plot, col = k)
```

```r
  abline(v = lambda_plot[MSE_plot == min(MSE_plot)], col = k)
}
legend("topright", cex = .5, as.character(sparsity), col = c(1:p), lty = rep(1, p))
```



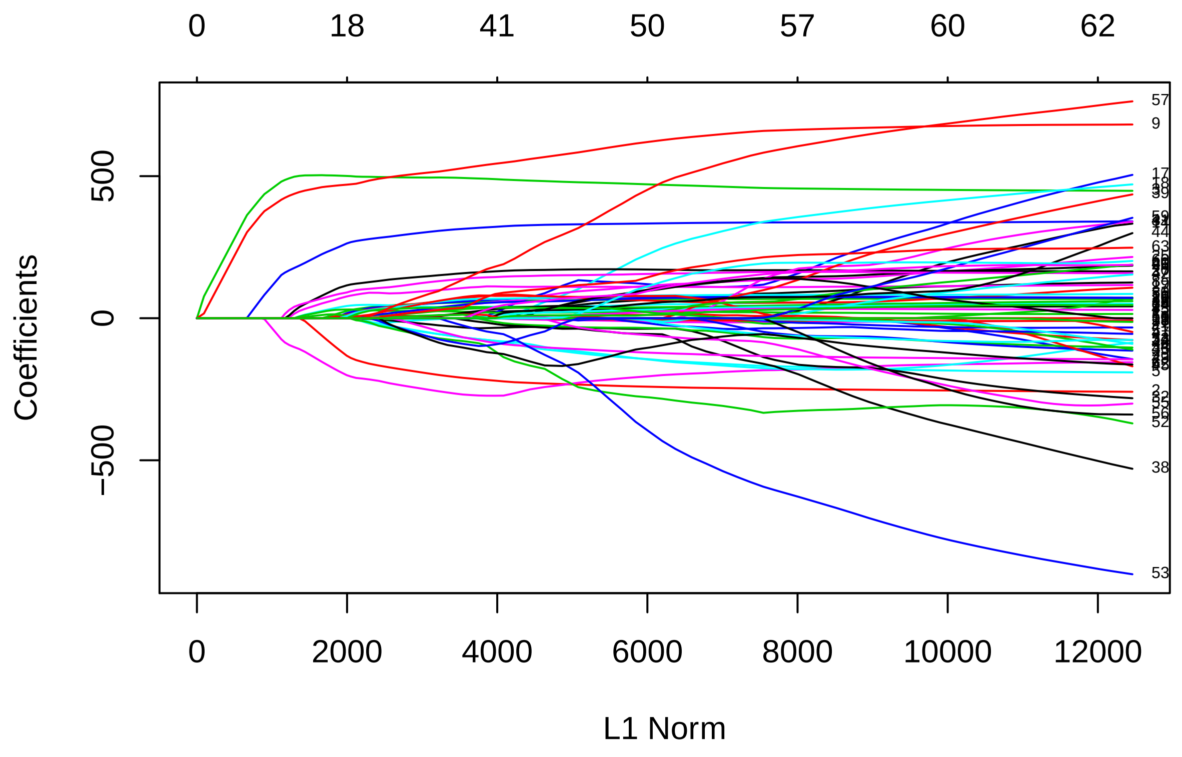
We can see from this plot that the optimal lambda increases as the sparsity level increases. When there's no sparsity at all, the optimal lambda is 0.

Lasso Part A

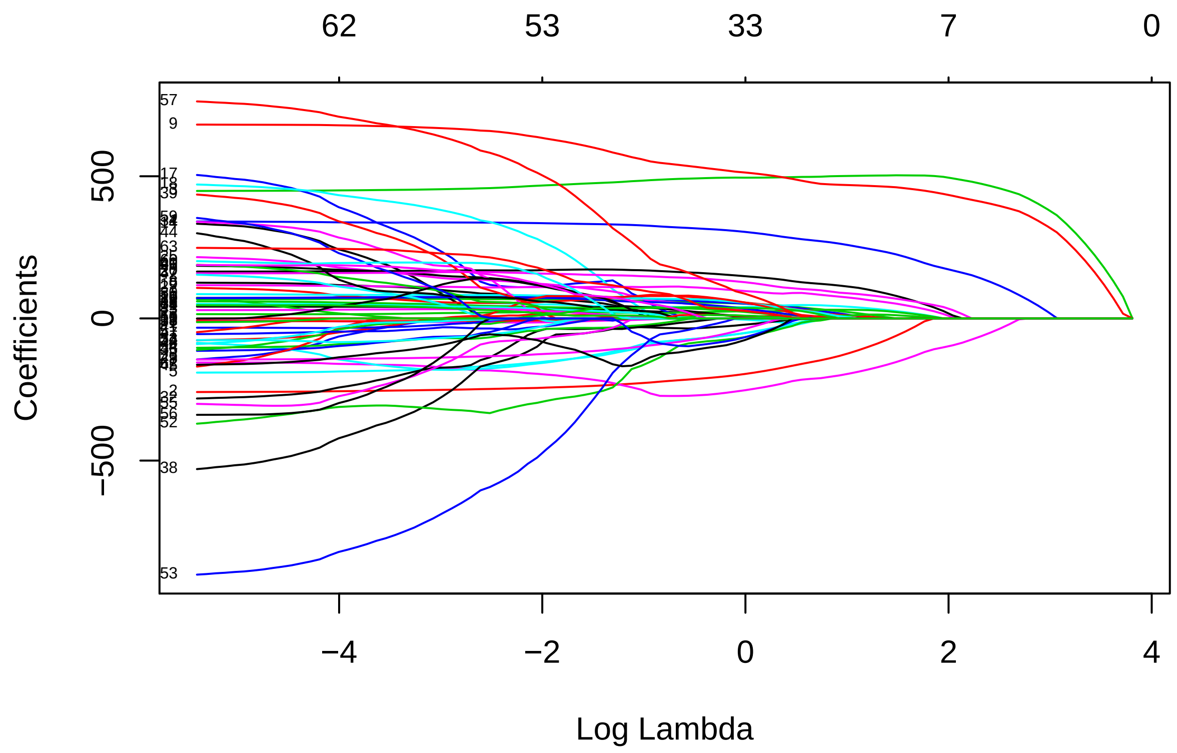
```r
library(glmnet)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-5
```

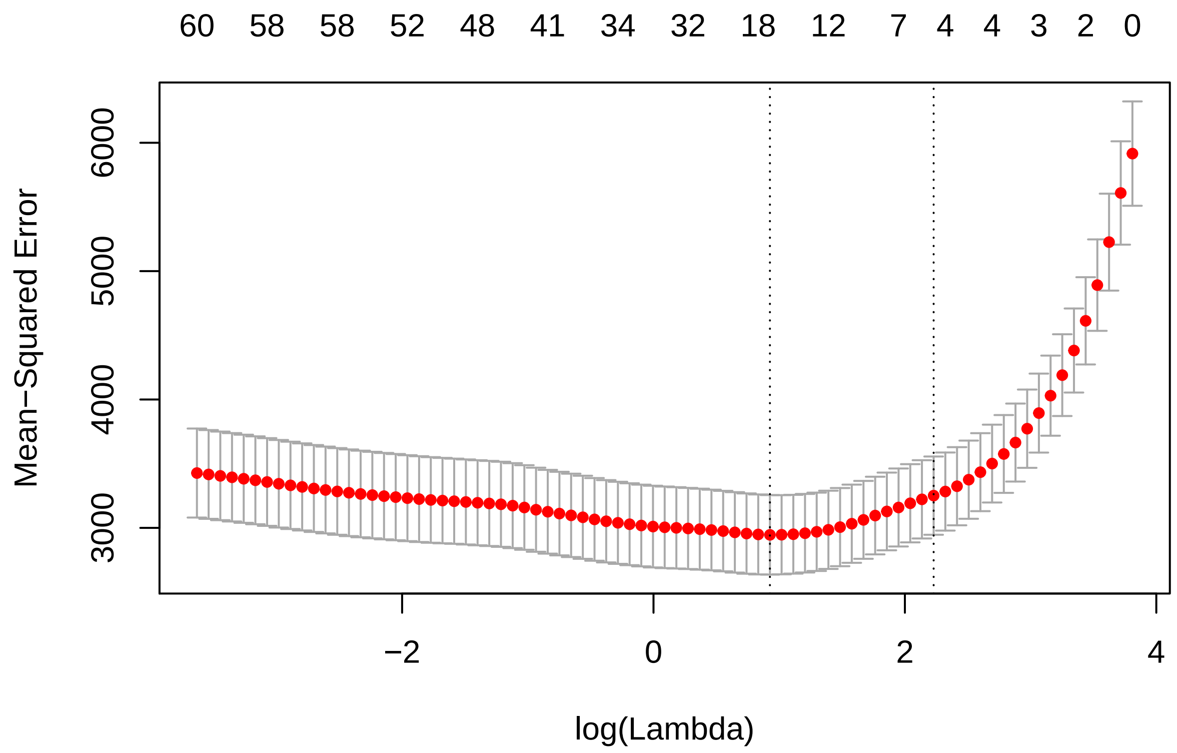
```r
diabetesX = read.csv("C:/Users/schen/Dropbox/toChensu/Stats/2016Fall/Big Data/Assignment5/diabetesX.csv
diabetesY = read.csv("C:/Users/schen/Dropbox/toChensu/Stats/2016Fall/Big Data/Assignment5/diabetesY.csv
X = as.matrix(diabetesX)
Y = as.matrix(diabetesY)

fit1 = glmnet(x = X, y = Y)
plot(fit1, xvar = "norm", label = TRUE)
```



```
plot(fit1, xvar = "lambda", label = TRUE)
```



```
#print(fit1)


### use cv.glmnet function in R for sanity check ###
cv.fit = cv.glmnet(x = X, y = Y)
plot.cv.glmnet(cv.fit)
```



Lasso Part B and C

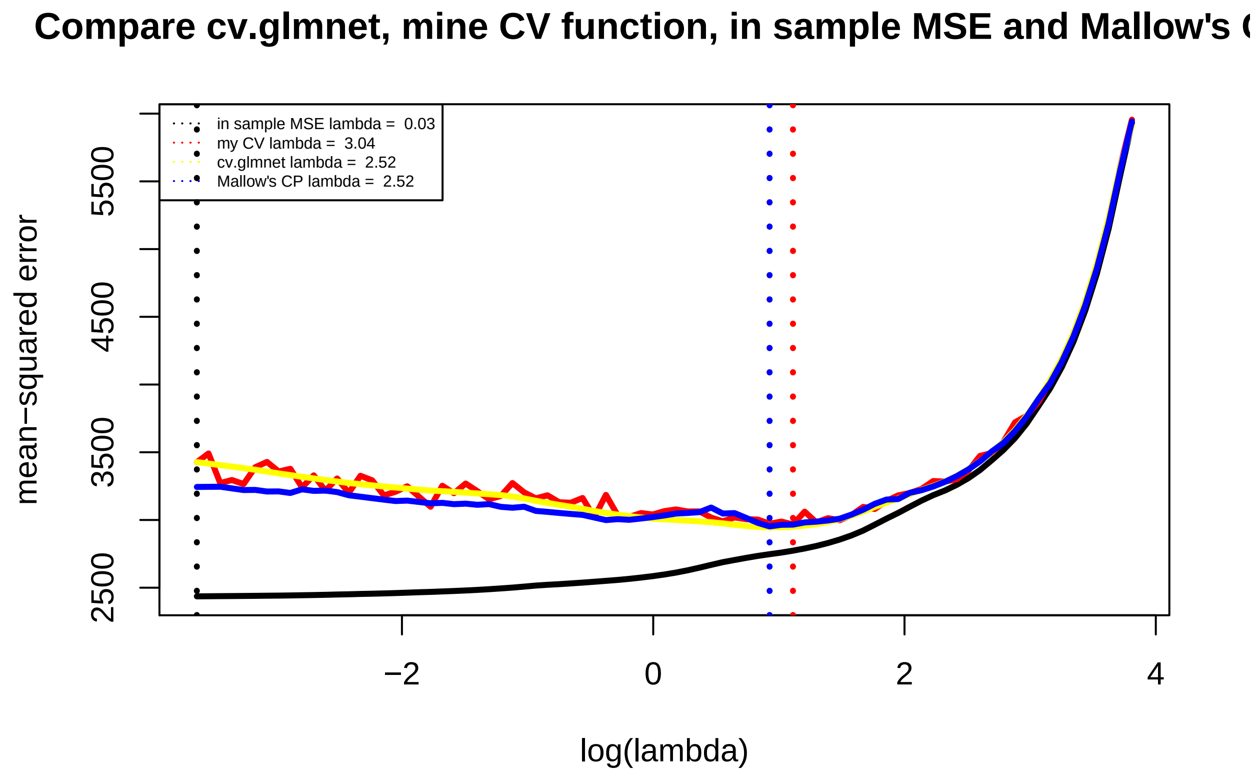
```
### function to calculate in-sample MSE and Mallow's CP
Mallow_cp = function(x, y, lambda)
{
  numobs = nrow(x)
  glmfit = glmnet(x, y, lambda=lambda)
  glm_pred = predict(glmfit, newx=x, s=lambda)
  s_lambda = sum(coef(glmfit) != 0)
  MSE = mean( (glm_pred - y)^2 )

  fit = lm(y~x)
  sigma_hat = summary(fit)[6]$sigma
  CP  = MSE + 2*s_lambda*(sigma_hat^2)/numobs
  return (list(MSE, CP))
}



### function for cross validation
cv = function(x, y, k, lambda)
{
  MSE_cv = rep(0, k)
```

```r
  data = cbind(x, y)
  numobs = nrow(data)
  numcol = ncol(data)
  cv_data = data[sample(numobs), ]

  #Create k equally size folds
  folds = cut( seq(1,numobs), breaks=k, labels=FALSE)

  #Perform k fold cross validation
  for(i in 1:k){
    #Segement your data by fold using the which() function
    test_index = which(folds==i,arr.ind=TRUE)
    test_data = cv_data[test_index, ]
    train_data = cv_data[-test_index, ]
    train_fit = glmnet(x=train_data[,-numcol], y=train_data[, numcol],
                  lambda=lambda)
    test_pred = predict(train_fit, newx=test_data[,-numcol], s=lambda)
    MSE_cv[i] = mean( (test_pred - test_data[,numcol])^2 )
  }
  return (MSE_cv)
}
```

Use cv.fit$lambda to compare optimal lambda by different ways to estimate generalization error: in sample MSE, cv.glmnet, my cross validation function and Mallow's CP:

```r
k = 10 #use 10-fold cross validation same as cv.glmnet
l = length(cv.fit$lambda)
MSE_cv = matrix(0, l, k)
MSE = rep(0, l)
CP = rep(0, l)

for (i in 1:l)
{
  MSE_cv[i, ] = cv(X, Y, k, cv.fit$lambda[i])
  MSE[i] = Mallow_cp(X, Y, cv.fit$lambda[i])[[1]]
  CP[i] = Mallow_cp(X, Y, cv.fit$lambda[i])[[2]]
}

### in sample MSE ###
plot(x=log(cv.fit$lambda), y=MSE, type = "l", xlab = "log(lambda)",
     ylab = "mean-squared error", lwd = 3,
     main = "Compare cv.glmnet, mine CV function, in sample MSE and Mallow's CP")
MSE_lambda = cv.fit$lambda[MSE == min(MSE)]
abline( v = log(MSE_lambda), lty = 3, lwd = 3)

### my CV function ###
plot_MSE_cv = rowMeans(MSE_cv)
lines(x=log(cv.fit$lambda), y=plot_MSE_cv, col="red", lwd = 3)
mycv_lambda = cv.fit$lambda[plot_MSE_cv == min(plot_MSE_cv)]
abline( v = log(mycv_lambda), lty = 3, col="red", lwd = 3)

### cv.glmnet in R ###
lines(x = log(cv.fit$lambda), y = cv.fit$cvm, col="yellow", lwd = 3)
```

```
cvglm_lambda = cv.fit$lambda[cv.fit$cvm == min(cv.fit$cvm)]
abline( v = log(cvglm_lambda), lty = 3, col="yellow", lwd = 3)

### Mallow's CP ###
lines(x=log(cv.fit$lambda), y=CP, col="blue", lwd = 3)
CP_lambda = cv.fit$lambda[CP == min(CP)]
abline( v = log(CP_lambda), lty = 3, col="blue", lwd = 3)

legend("topleft", cex = .5,
       c(paste("in sample MSE lambda = ", round(MSE_lambda, 2)),
         paste("my CV lambda = ", round(mycv_lambda, 2)),
         paste("cv.glmnet lambda = ", round(cvglm_lambda, 2)),
         paste("Mallow's CP lambda = ", round(CP_lambda, 2))),
       col = c("black","red", "yellow", "blue"), lty = c(3,3,3,3))
```



In sample MSE is always increasing as lambda increases. The 10-fold cross validation error and Mallow's CP show a very similar trend in MSE, and optimal lambdas chosen by cross validation and Mallow's CP are close to the optimal lambda chosen by R buildin function cv.glmnet.