Sparsity and Lasso

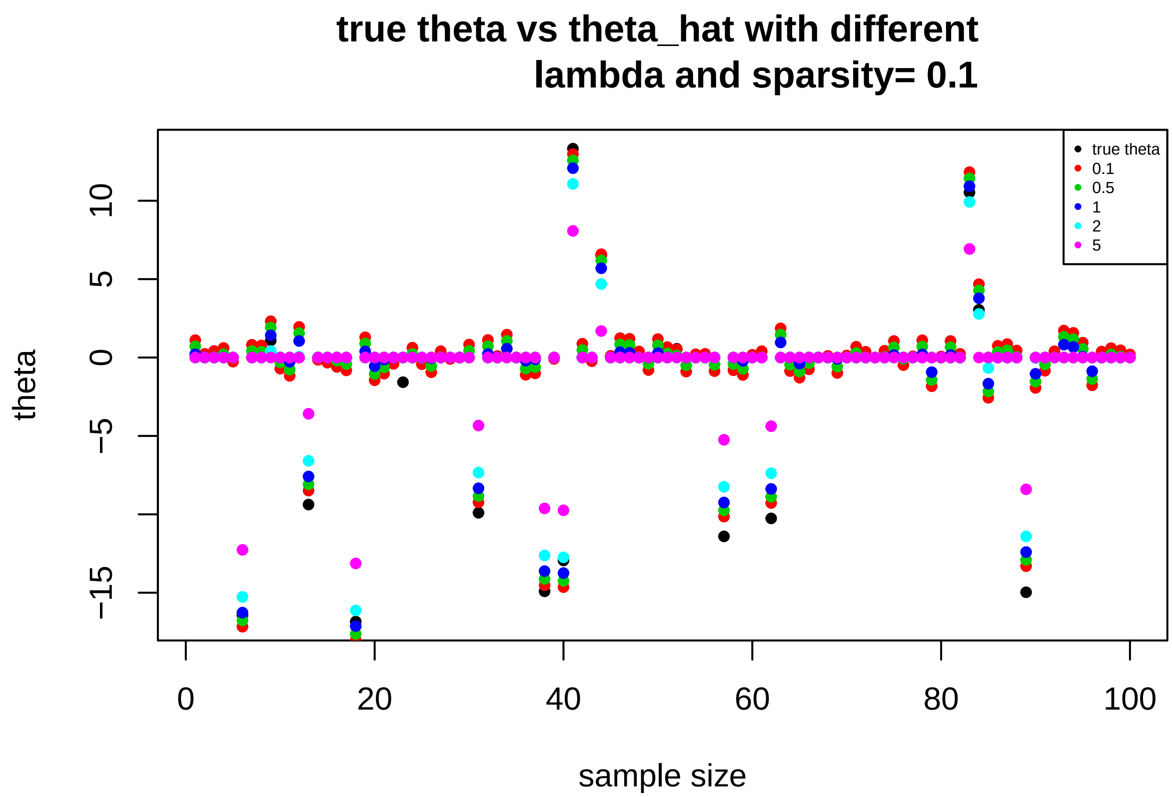
Su Chen

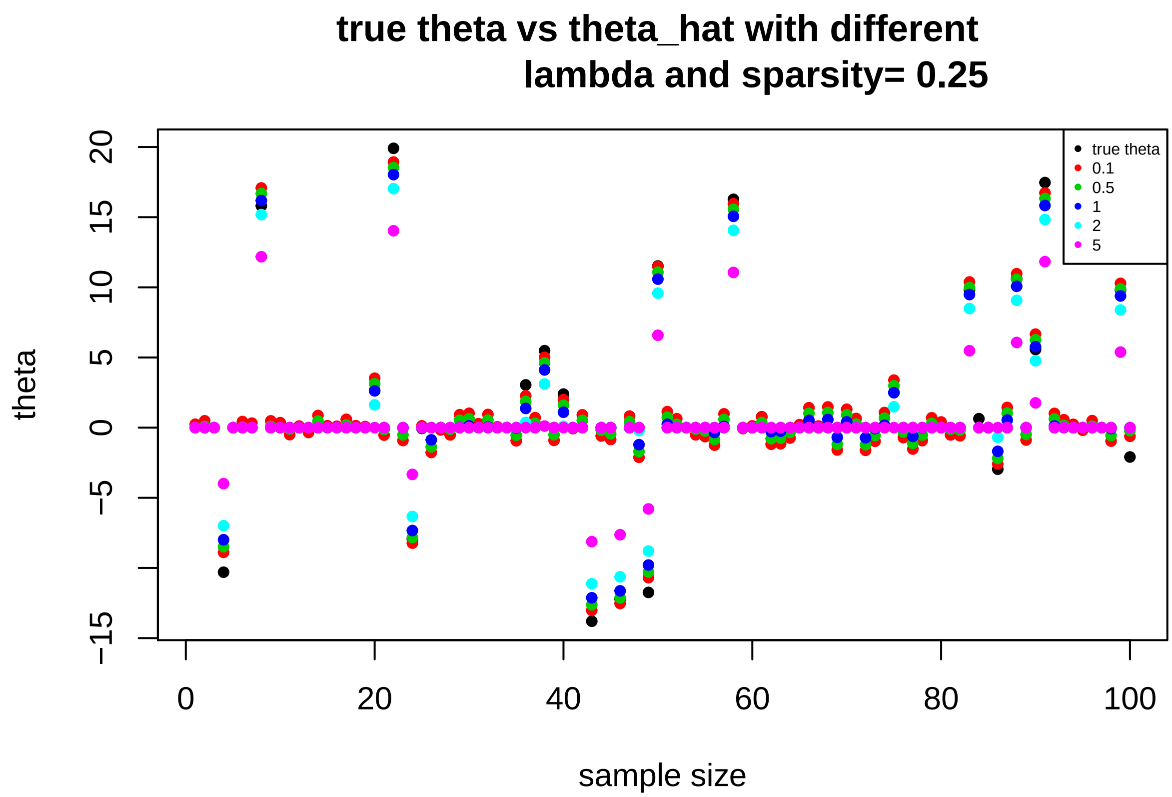
October 10, 2016

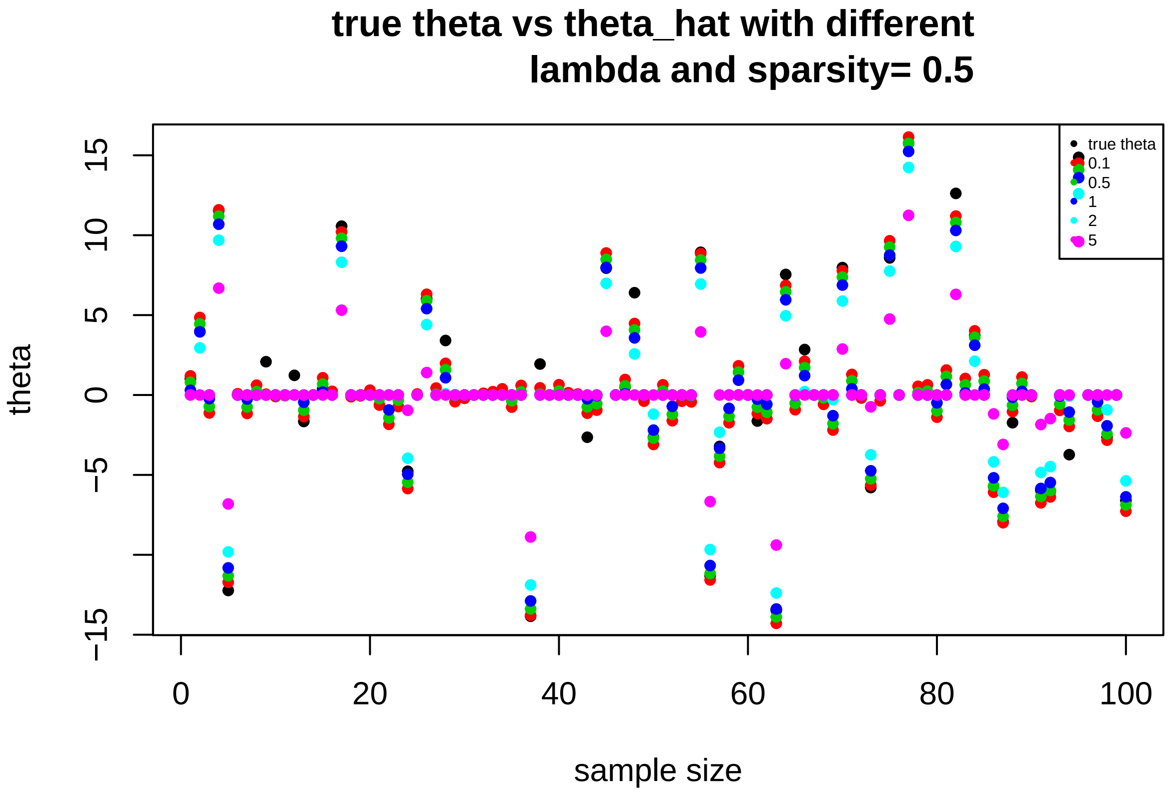
Penalized likelihood and soft thresholding Part B

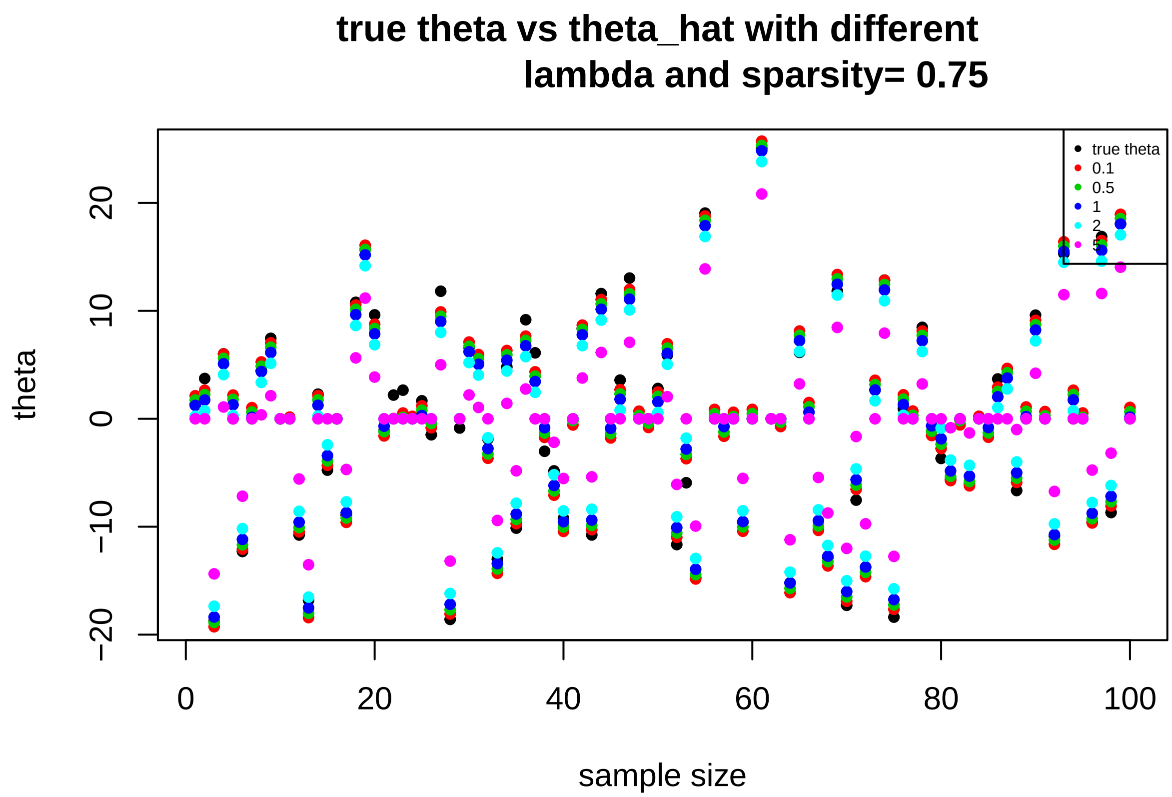
```
### function to sample normal data based on different level of sparsity
simu_data = function(n, sparsity, mu, s)
{
  ### generate Theta
  theta = rbinom(n, 1, prob = sparsity)*rnorm(n, mean=mu, sd=s)

  ### 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)
```

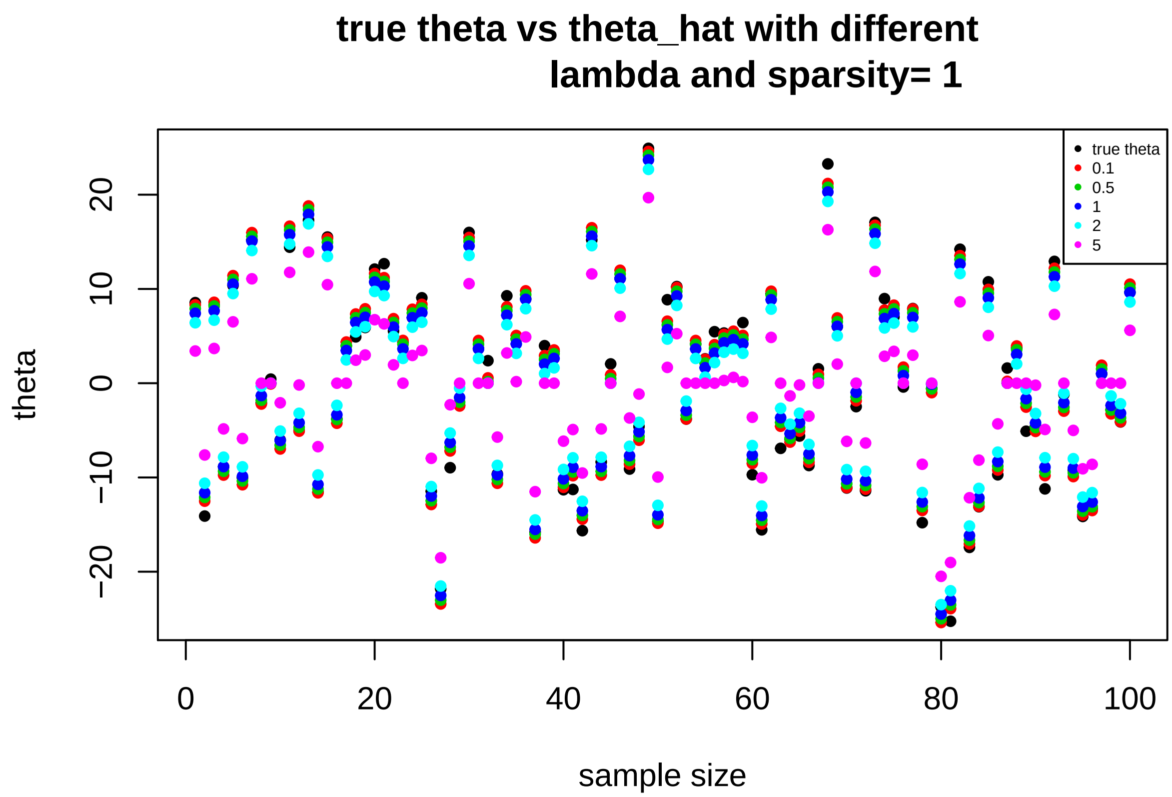
```
{
  simulation = simu_data(n, sparsity=sparsity[k], mu=0, s=10)
  theta[k,] = simulation[[1]]
  y[k,] = simulation[[2]]

  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
true theta
0.1
0.5
1
2
5
20
15
10
5
0
−5
−15
0
20
40
60
80
100


true theta vs theta_hat with different
lambda and sparsity= 0.5
theta
sample size
true theta
0.1
0.5
1
2
5
15
10
5
0
−5
−15
0
20
40
60
80
100

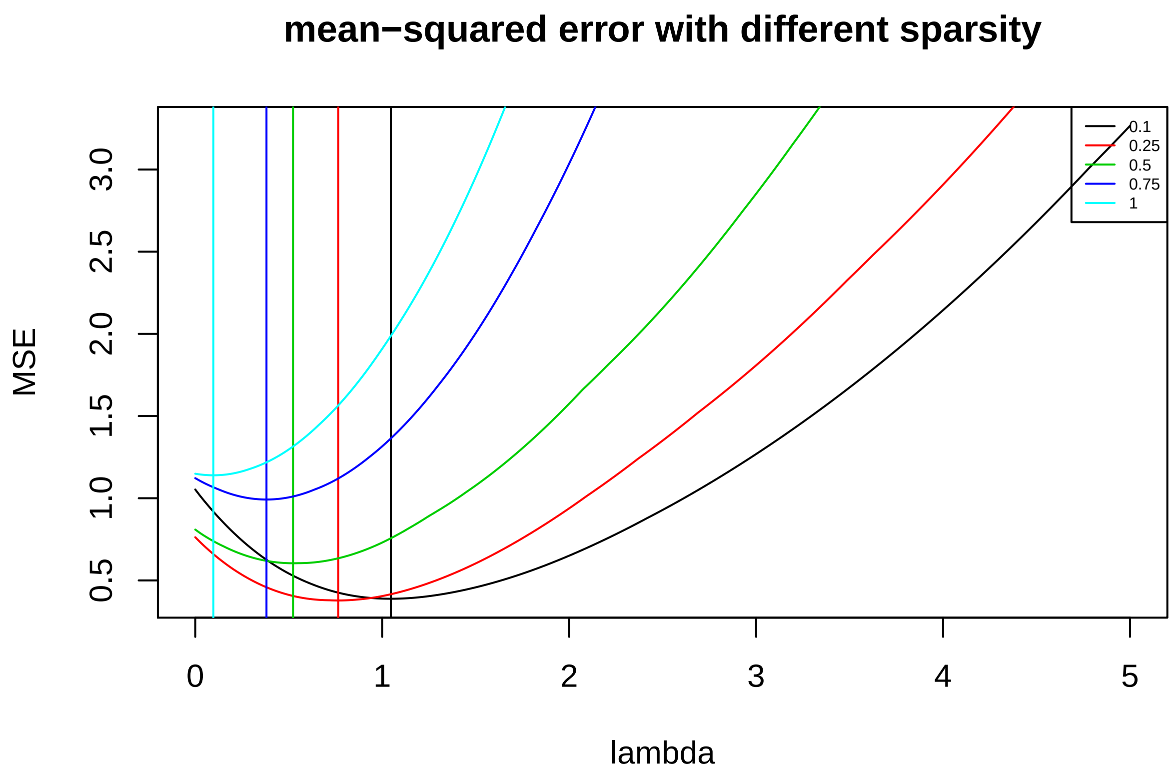

true theta vs theta_hat with different
lambda and sparsity= 0.75
theta
sample size
true theta
0.1
0.5
1
2
5
20
10
0
−10
−20
0
20
40
60
80
100



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)
```

```
  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))
```



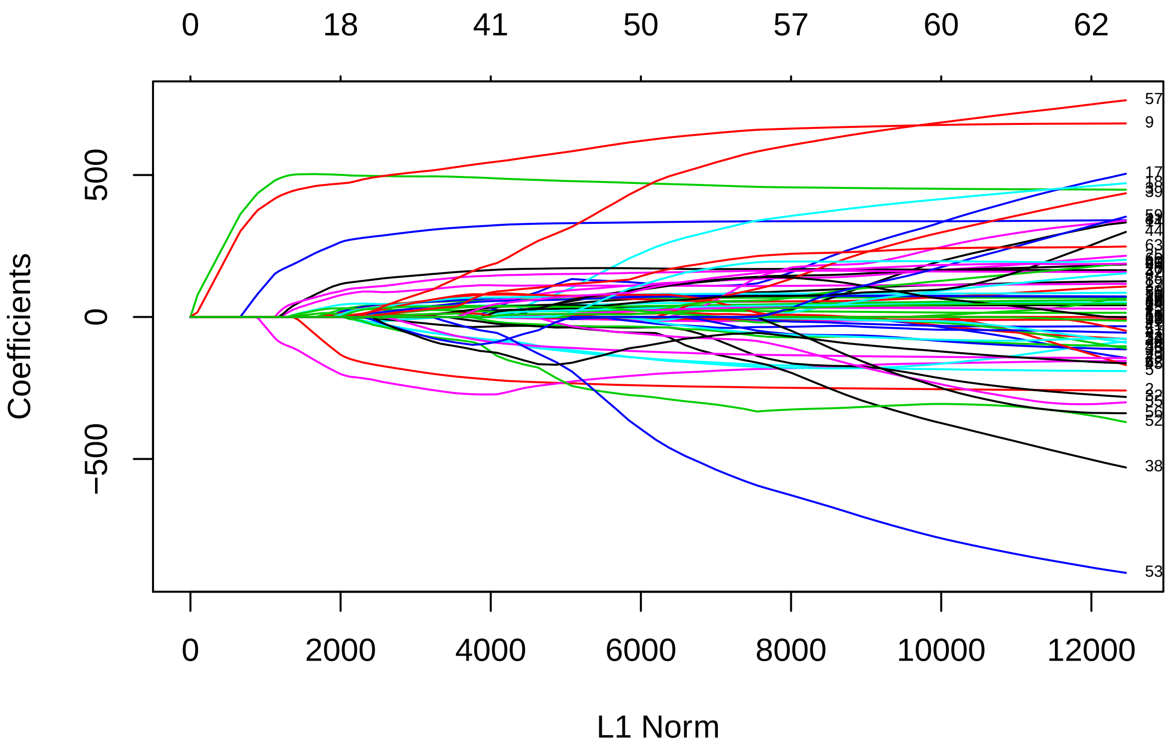
We can see from this plot that the optimal lambda increases as the sparsity level increases. When there's no sparsity at all, the optimal lambda is 0.

Lasso Part A

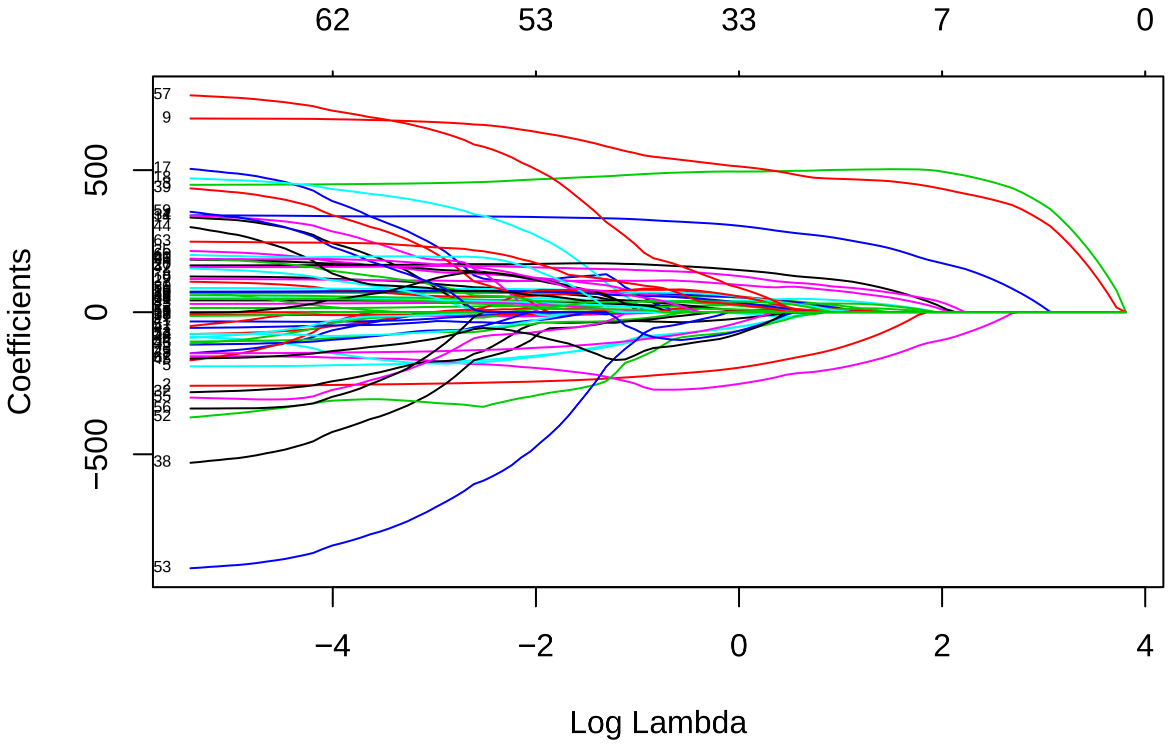
```
library(glmnet)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-5
```

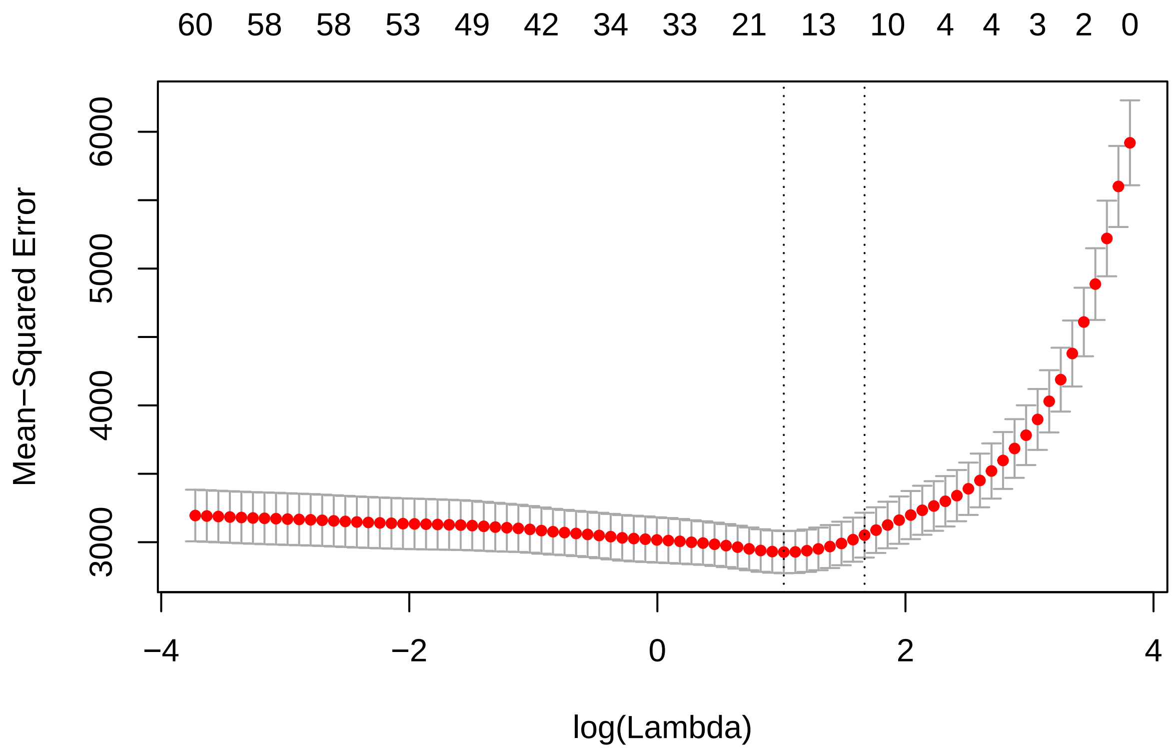
```
diabetesX = read.csv("C:/Users/schen/Dropbox/toChensu/Stats/2016Fall/Big Data/Assignment5/diabetesX.csv
diabetesY = read.csv("C:/Users/schen/Dropbox/toChensu/Stats/2016Fall/Big Data/Assignment5/diabetesY.csv
X = as.matrix(diabetesX)
Y = as.matrix(diabetesY)

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



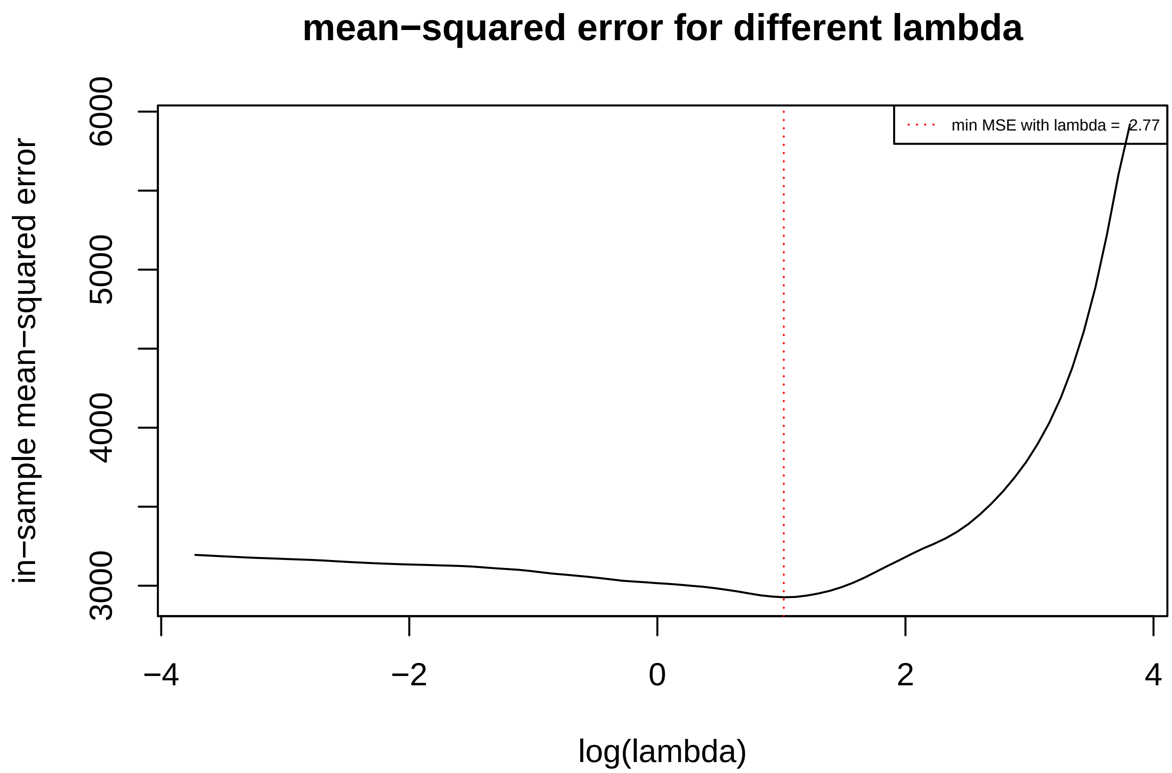

```r
plot(fit1, xvar = "lambda", label = TRUE)
```




```r
#print(fit1)

cv.fit = cv.glmnet(x = X, y = Y)
plot.cv.glmnet(cv.fit)
```




```
plot(x = log(cv.fit$lambda), y = cv.fit$cvm, type = "l",
     xlab = "log(lambda)", ylab = "in-sample mean-squared error",
     main = "mean-squared error for different lambda")
min_index = which(cv.fit$cvm == min(cv.fit$cvm))
cv.lambda = cv.fit$lambda[min_index]
abline( v = log(cv.lambda), lty = 3, col = "red")
legend("topright", cex = .5,
       paste("min MSE with lambda = ", round(cv.lambda, 2)),
       col = "red", lty = 3)
```

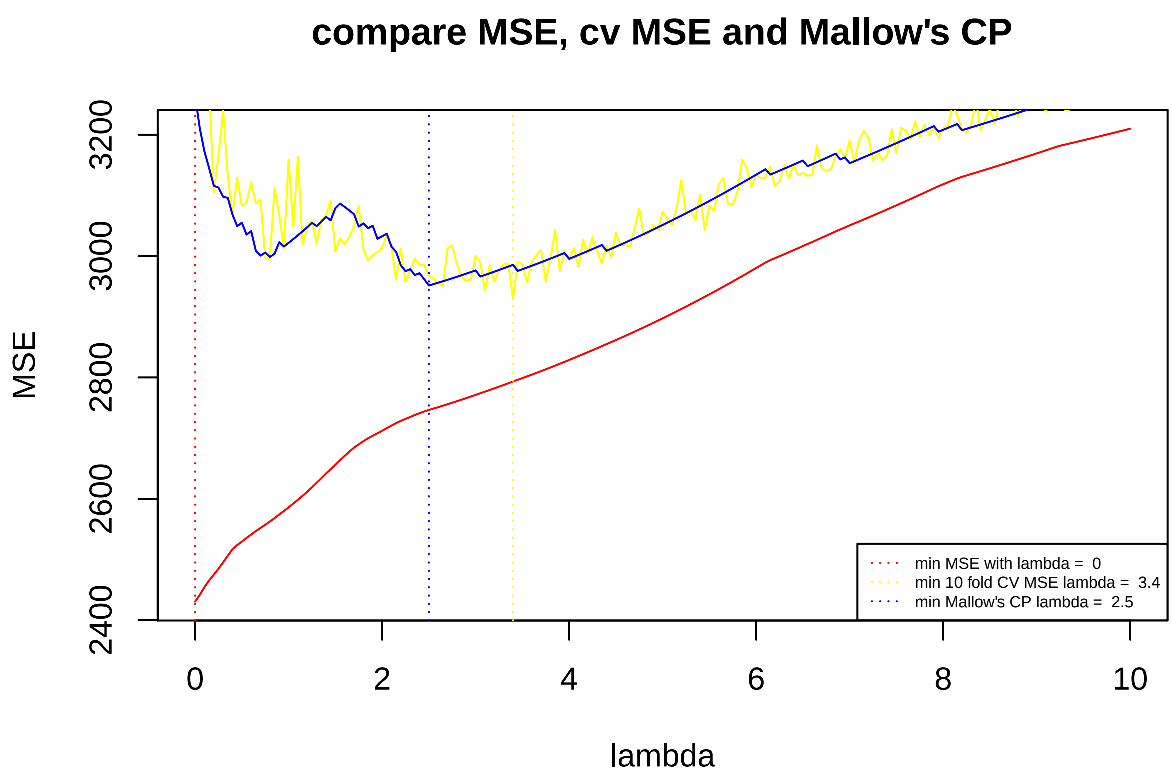


Lasso Part B and C

```
### 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)
}


### try different lambda values and plot MSE, MSE_cv and Mallow's CP
lambda_test = seq(0, 10, 0.05)
#lambda_test = cv.fit$lambda
k = 10
l = length(lambda_test)
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, lambda_test[i])
   MSE[i] = Mallow_cp(X, Y, lambda_test[i])[[1]]
   CP[i] = Mallow_cp(X, Y, lambda_test[i])[[2]]
 }

plot(x=lambda_test, y=MSE, type="l", xlab="lambda", ylab="MSE",
     main="compare MSE, cv MSE and Mallow's CP", col="red" )
min_lambda_MSE = lambda_test[MSE == min(MSE)]
abline( v = min_lambda_MSE, lty = 3, col="red")

plot_MSE_cv = rowMeans(MSE_cv)
#plot_sd = apply(MSE_cv, 1, sd)
lines(x=lambda_test, y=plot_MSE_cv, col = "yellow" )
min_lambda_cv = lambda_test[plot_MSE_cv == min(plot_MSE_cv)]
abline( v = min_lambda_cv, lty = 3, col="yellow")

lines(x=lambda_test, y=CP, col="blue")
min_lambda_CP = lambda_test[CP == min(CP)]
abline( v = min_lambda_CP, lty = 3, col="blue")
```

```r
legend("bottomright", cex = .5,
       c(paste("min MSE with lambda = ", min_lambda_MSE),
         paste("min 10 fold CV MSE lambda = ", min_lambda_cv),
         paste("min Mallow's CP lambda = ", min_lambda_CP)),
       col = c("red","yellow","blue"), lty = c(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 both similar to the optimal lambda chosen by R buildin function cv.glmnet.