HDS Exercise set 5

Shabbeer Hassan

Problem 1 – Solution

```
# Libraries
library(glmnet)
```

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

```
## Loaded glmnet 3.0
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(plotmo)
```

```
## Loading required package: Formula
```

```
## Loading required package: plotrix
```

```
## Loading required package: TeachingDemos
```

```
library(car)
```

```
## Loading required package: carData
```

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v tibble  2.1.3     v purrr   0.3.2
## v tidyr   1.0.0     v dplyr   0.8.3
## v readr   1.3.1     v stringr 1.4.0
## v tibble  2.1.3     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## x tidyr::pack()   masks Matrix::pack()
## x dplyr::recode() masks car::recode()
## x purrr::some()   masks car::some()
## x tidyr::unpack() masks Matrix::unpack()
```

```r
library(ggplot2)
library(MASS)
```

```
##
## Attaching package: 'MASS'

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

```r
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
library(dplyr)
library(boot)
```

```
##
## Attaching package: 'boot'

## The following object is masked from 'package:car':
##
##     logit

## The following object is masked from 'package:lattice':
##
##     melanoma
```

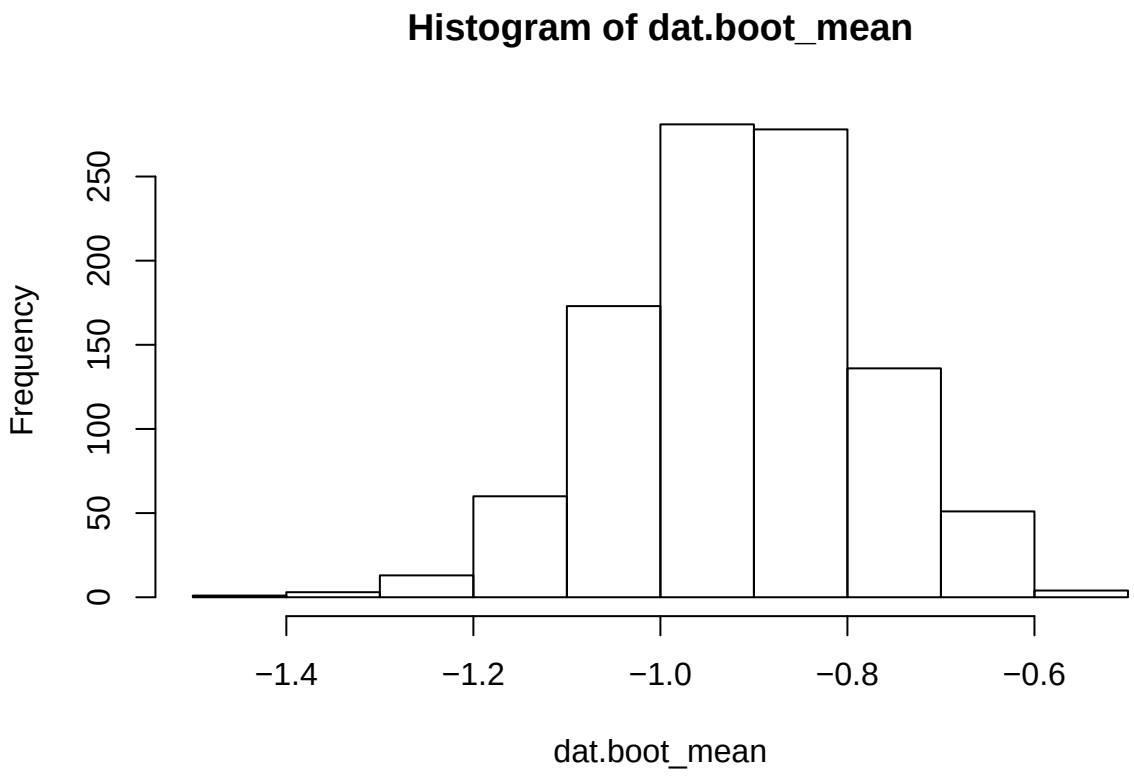
Problem 1 – Solution

```r
# Import data
dat <- read.csv("E:/Dropbox/Important_Documents/Doctoral_Work/Courses/High Dimensional Stats/2019/Week 5

# Estimate for mean
dat_mean <- mean(dat$x * dat$y)
dat_mean
```

```
## [1] -0.9129148
```

```r
# Bootstrap mean
B = 1000 # bootstrap samples
n = 100

# Function for mean of product of columns
foo <- function(data, indices){
  dt <- data[indices,]
  c(
    mean(dt[,1] * dt[,2])
  )
}

# Draw bootstrap samples off dataset and take mean of product each time
# Use function "boot" to generate boostsrap samples from our data and use the function above on each 10
dat.boot <- boot(dat, foo , R = B)
dat.boot_mean <- dat.boot$t
hist(dat.boot_mean, index=1)
```

```
## Warning in plot.window(xlim, ylim, "", ...): "index" is not a graphical
## parameter

## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## "index" is not a graphical parameter

## Warning in axis(1, ...): "index" is not a graphical parameter

## Warning in axis(2, ...): "index" is not a graphical parameter
```




```
# Confidence INtervals
boot.ci(boot.out = dat.boot, index = 1, type = "perc")
```

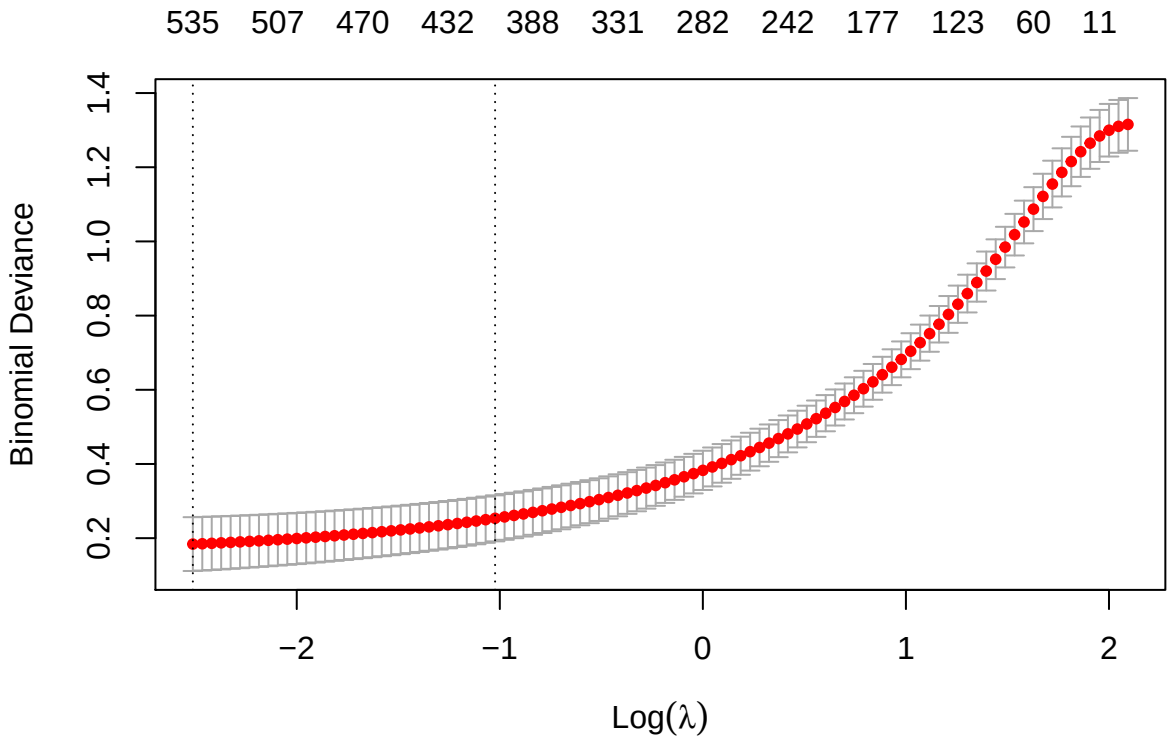
```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = dat.boot, type = "perc", index = 1)
##
## Intervals :
## Level     Percentile
## 95%   (-1.1702, -0.6574 )
## Calculations and Intervals on Original Scale
```

Problem 2 – Solution

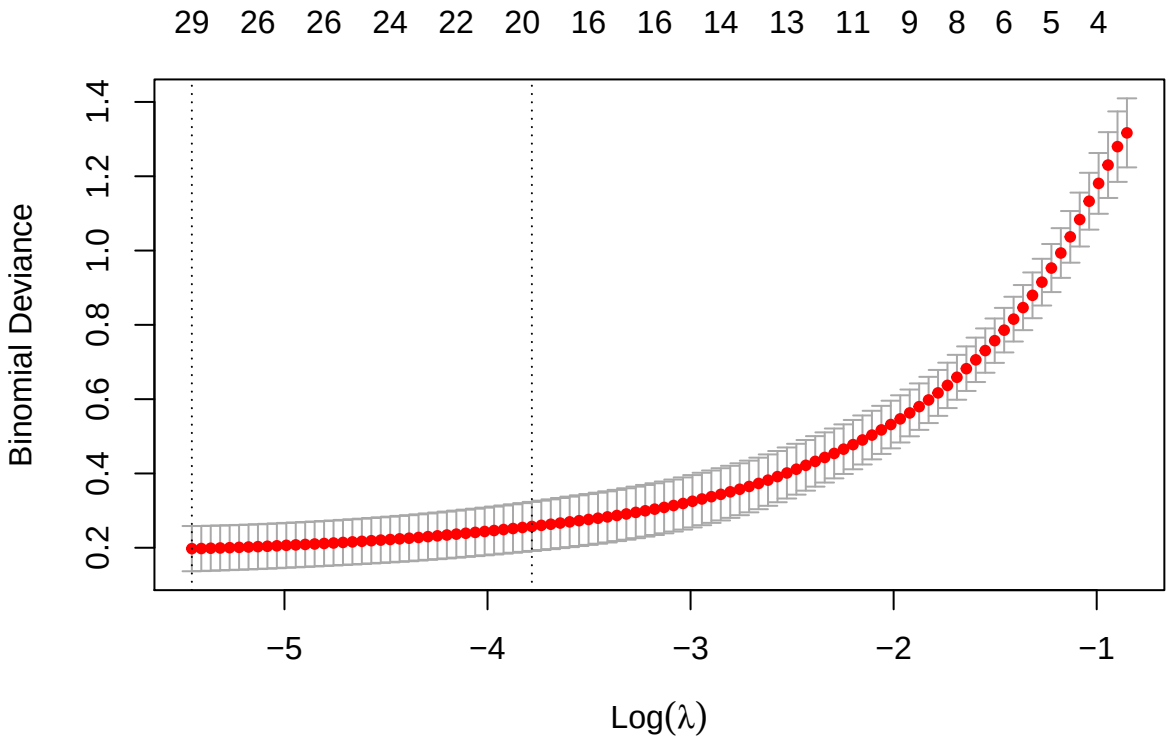
(a)

```
# Import data
leu_dat <- read.csv("E:/Dropbox/Important_Documents/Doctoral_Work/Courses/High Dimensional Stats/2019/W


# Test & Train
test <- leu_dat[c(71, 72), ]
```

```r
train <- leu_dat[-c(71, 72), ]

# Response & Predictors
X.test <- test[, -1]
X.train <- train[, -1]
Y.test <- test[, 1]
Y.train <- train[, 1]

# Ridge Reg
cv.ridge <- cv.glmnet(x = as.matrix(X.train), y  = as.matrix(Y.train), family = "binomial", type.measure
plot(cv.ridge)
```




```r
# LASSO
cv.lasso <- cv.glmnet(x = as.matrix(X.train), y  = as.matrix(Y.train), family = "binomial", type.measure
plot(cv.lasso)
```




```
## Ridge deviance plot
ridge.glmnet <- cv.ridge$glmnet.fit
ridge_mod <- as.data.frame(cbind(ridge.glmnet$dev.ratio, ridge.glmnet$lambda)) # getting dev.ratio & la
colnames(ridge_mod) <- c("dev.ratio", "lambda")
ridge_mod <- formatC(ridge_mod, digits = 4, format = "f") # limiting the decimal places
```

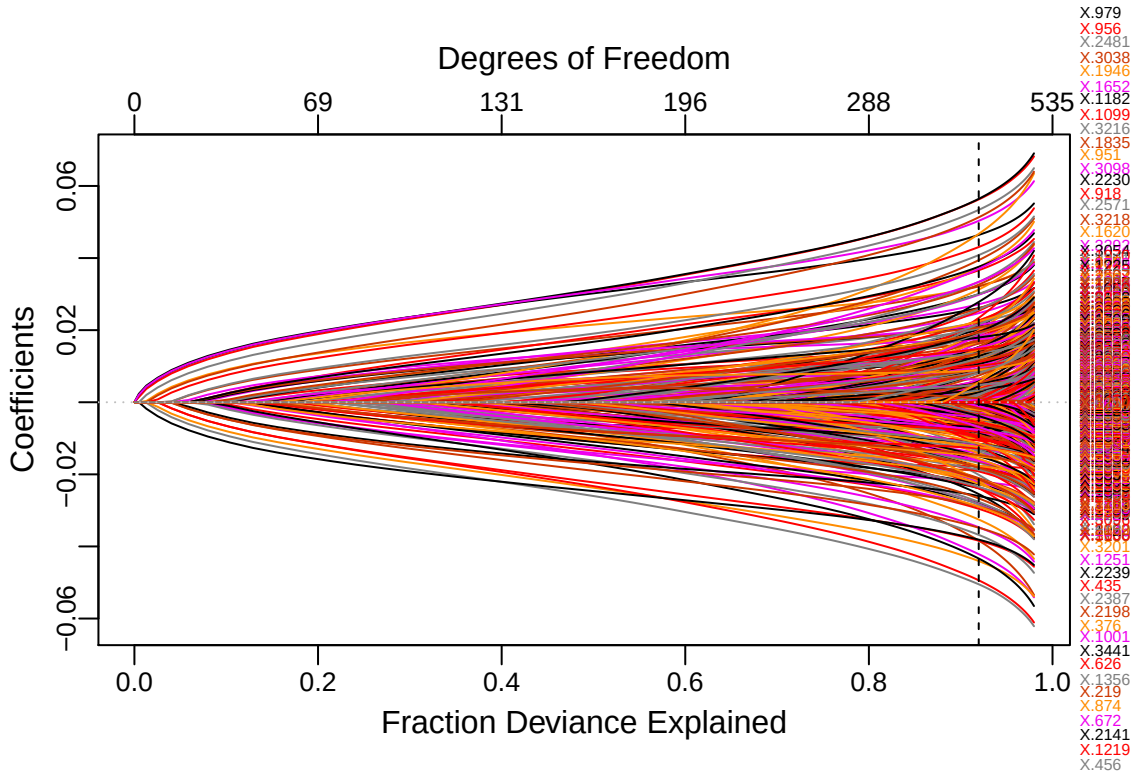
```
## Warning in formatC(ridge_mod, digits = 4, format = "f"): class of 'x' was
## discarded
```

```
## Error in is.finite(x): default method not implemented for type 'list'
```

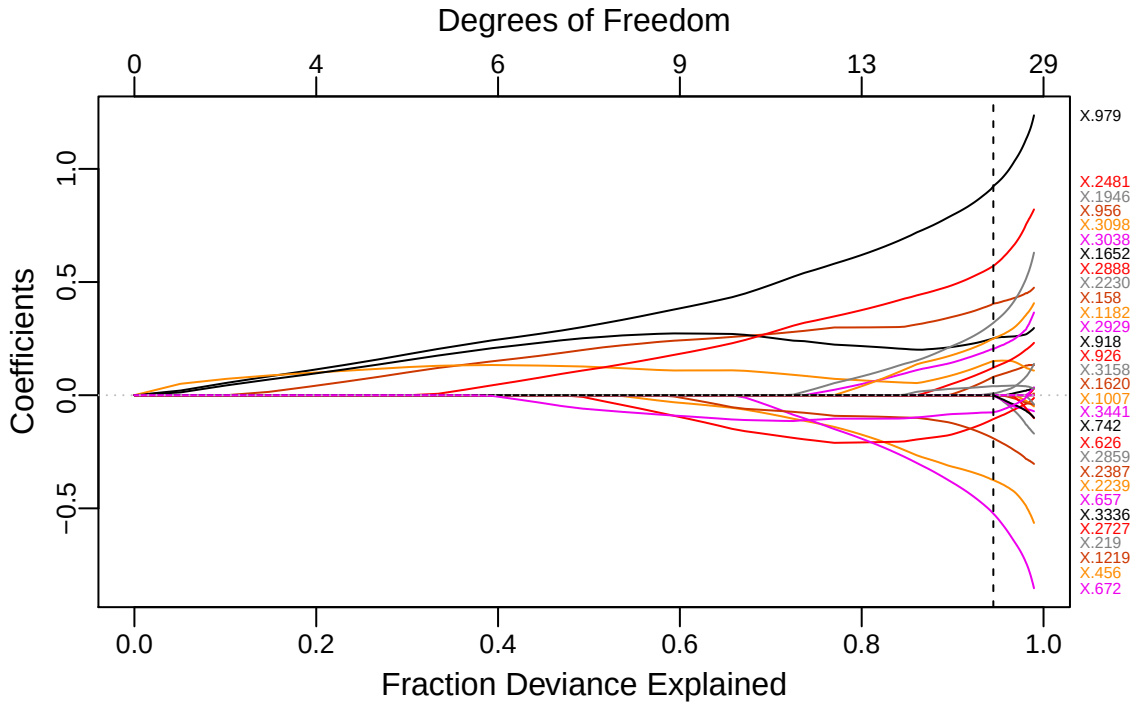
```
plot_glmnet(cv.ridge$glmnet.fit, label = T, xvar = "dev")
```

```
## Warning in TeachingDemos::spread.labs(beta[iname, ncol(beta)], mindiff =
## 1.2 * : Maximum iterations reached
```

```
abline(v = ridge_mod[ridge_mod$lambda == cv.ridge$lambda.1se, "dev.ratio"] , lty = 2)
```




```
## Lasso deviance plot
lasso.glmnet <- cv.lasso$glmnet.fit
lasso_mod <- as.data.frame(cbind(lasso.glmnet$dev.ratio, lasso.glmnet$lambda)) # getting dev.ratio & la
colnames(lasso_mod) <- c("dev.ratio", "lambda")

plot_glmnet(cv.lasso$glmnet.fit, label = T, xvar = "dev")
abline(v = lasso_mod[lasso_mod$lambda == cv.lasso$lambda.1se, "dev.ratio"] , lty = 2)
```




```
## Prediction - Ridge
response_ridge <- predict(cv.ridge, as.matrix(X.test), s = "lambda.1se", type = "response")
response_ridge
```

```
##             1
## 71 0.01487969
## 72 0.89859321
```

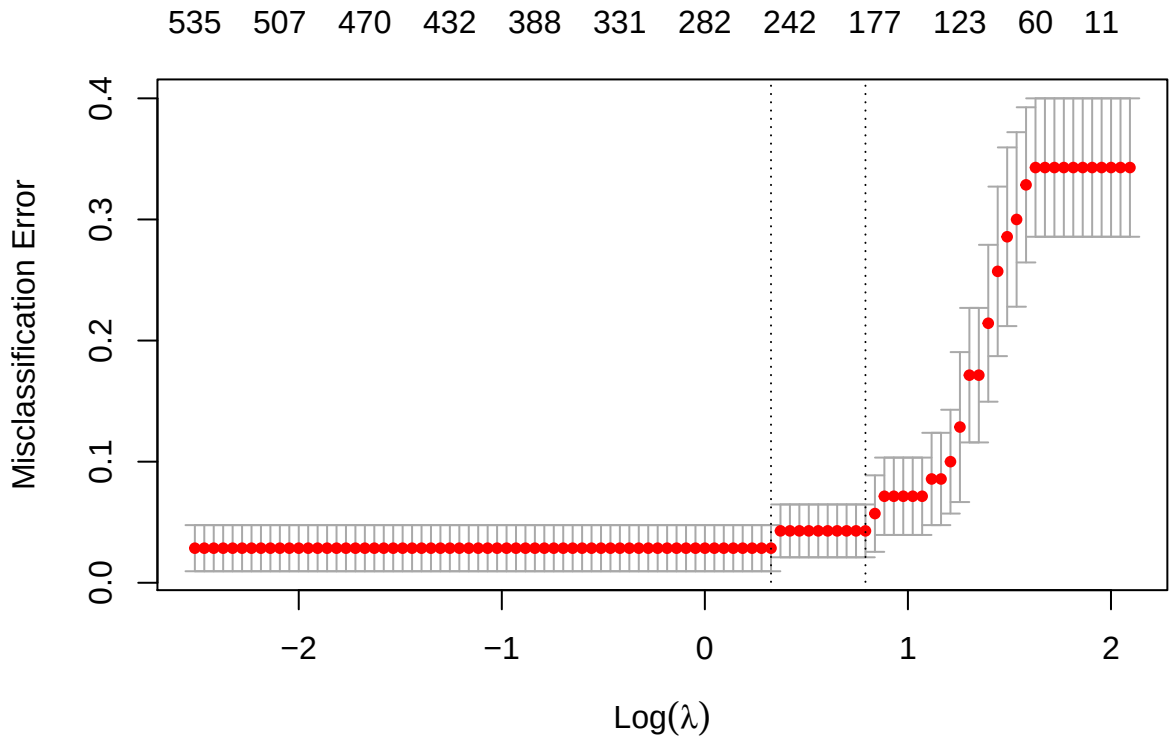
```
## Prediction - Lasso
response_lasso <- predict(cv.lasso, as.matrix(X.test), s = "lambda.1se", type = "response")
response_lasso
```

```
##             1
## 71 0.04625341
## 72 0.97238049
```

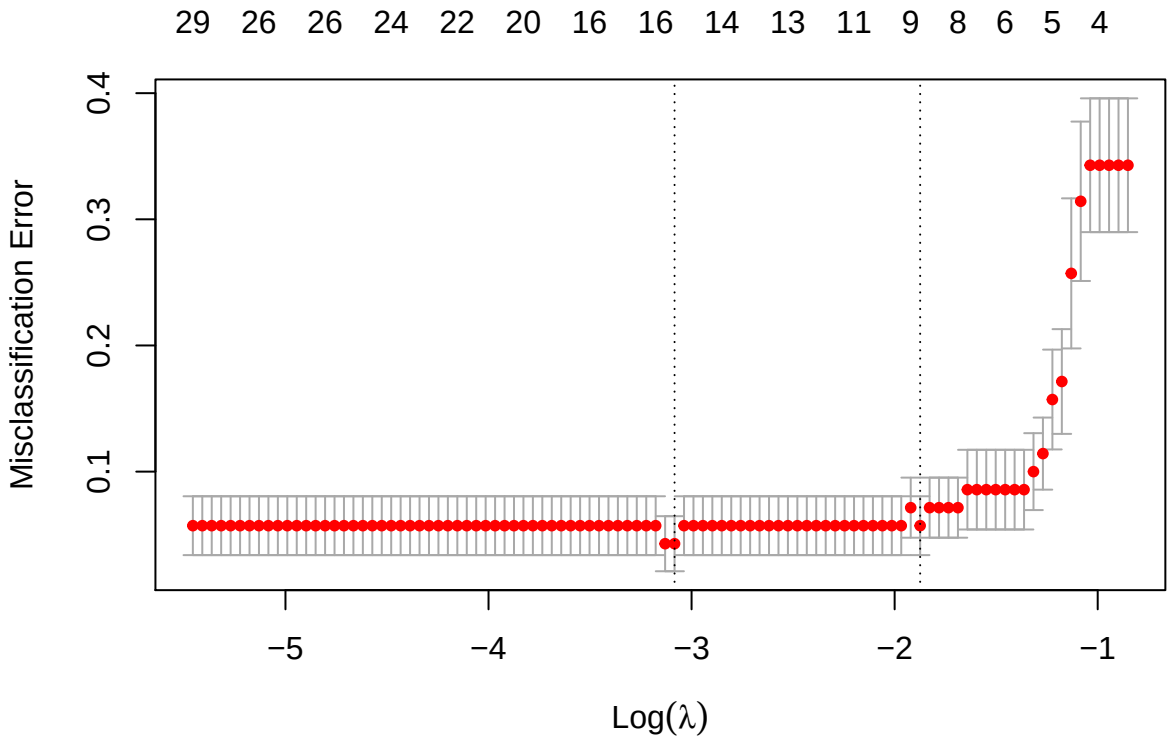
For both ridge and lasso the predictions are åretty much on the spot though ridge gives better prediction probability than lasso.

(b)

```
# Ridge Reg
cv.ridge <- cv.glmnet(x = as.matrix(X.train), y  = as.matrix(Y.train), family = "binomial", type.measur
plot(cv.ridge)
```




```
# LASSO
cv.lasso <- cv.glmnet(x = as.matrix(X.train), y  = as.matrix(Y.train), family = "binomial", type.measur
plot(cv.lasso)
```




```
## Ridge deviance plot
ridge.glmnet <- cv.ridge$glmnet.fit
ridge_mod <- as.data.frame(cbind(ridge.glmnet$dev.ratio, ridge.glmnet$lambda)) # getting dev.ratio & la
colnames(ridge_mod) <- c("dev.ratio", "lambda")
ridge_mod <- formatC(ridge_mod, digits = 4, format = "f") # limiting the decimal places
```

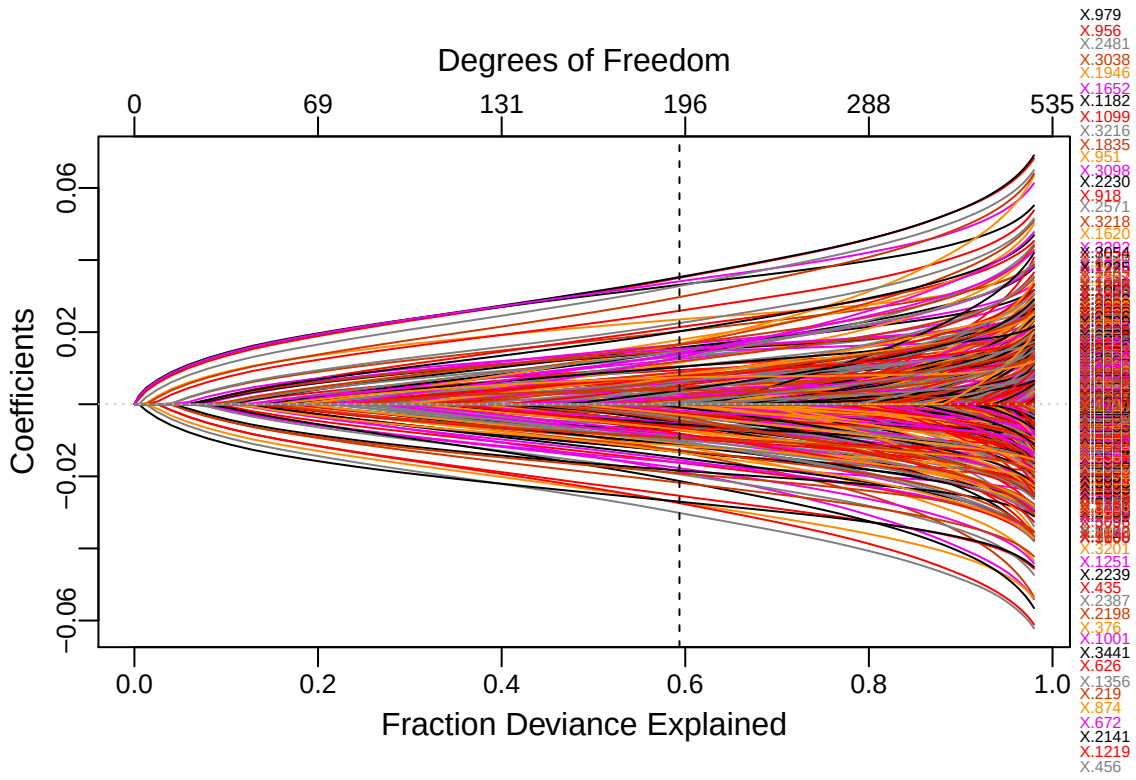
```
## Warning in formatC(ridge_mod, digits = 4, format = "f"): class of 'x' was
## discarded
```

```
## Error in is.finite(x): default method not implemented for type 'list'
```

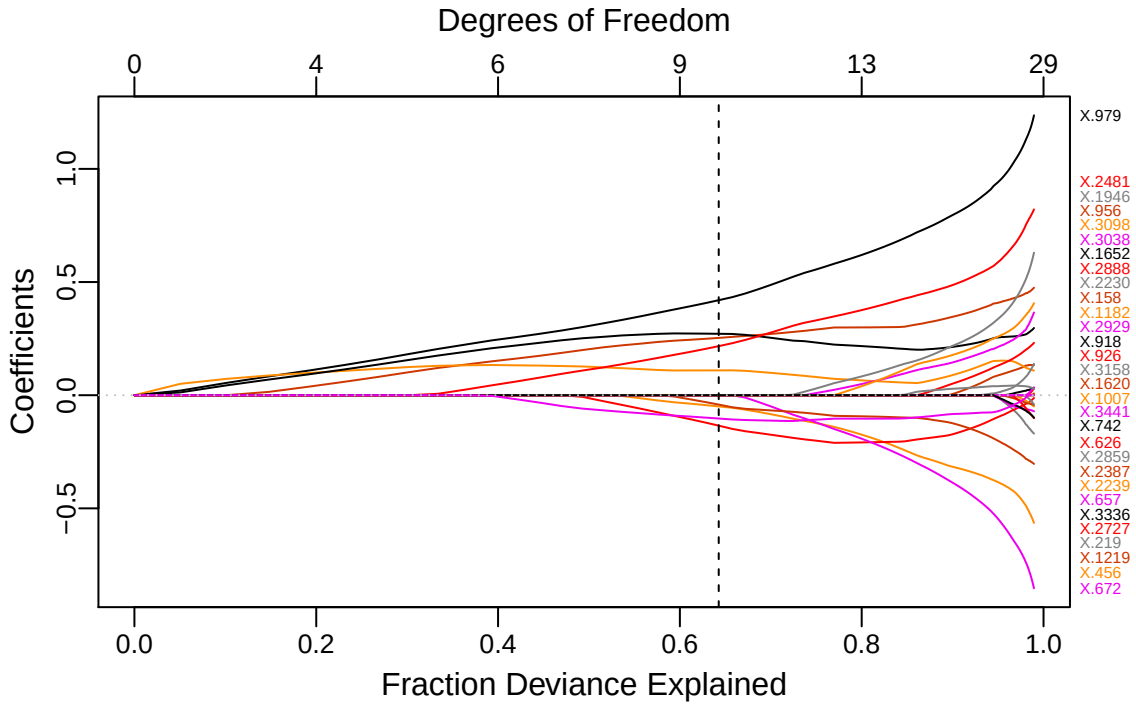
```
plot_glmnet(cv.ridge$glmnet.fit, label = T, xvar = "dev")
```

```
## Warning in TeachingDemos::spread.labs(beta[iname, ncol(beta)], mindiff =
## 1.2 * : Maximum iterations reached
```

```
abline(v = ridge_mod[ridge_mod$lambda == cv.ridge$lambda.1se, "dev.ratio"] , lty = 2)
```




```
## Lasso deviance plot
lasso.glmnet <- cv.lasso$glmnet.fit
lasso_mod <- as.data.frame(cbind(lasso.glmnet$dev.ratio, lasso.glmnet$lambda)) # getting dev.ratio & la
colnames(lasso_mod) <- c("dev.ratio", "lambda")

plot_glmnet(cv.lasso$glmnet.fit, label = T, xvar = "dev")
abline(v = lasso_mod[lasso_mod$lambda == cv.lasso$lambda.1se, "dev.ratio"] , lty = 2)
```




```
## Prediction - Ridge
response_ridge <- predict(cv.ridge, as.matrix(X.test), s = "lambda.1se", type = "response")
response_ridge
```

```
##            1
## 71 0.1318155
## 72 0.6493725
```

```
## Prediction - Lasso
response_lasso <- predict(cv.lasso, as.matrix(X.test), s = "lambda.1se", type = "response")
response_lasso
```

```
##            1
## 71 0.1564105
## 72 0.7994742
```

For both ridge and lasso the predictions are åretty much on the spot though ridge gives better prediction probability than lasso. But when using "type.measure"=class, the prediction probabilities have imporved quite a bit

Problem 3 – Solution

```r
# Fit lasso model
fit_lasso <- cv.glmnet(type.measure = "class", x = as.matrix(leu_dat[,-1]), y = as.matrix(leu_dat[,1]),
plot_glmnet(fit_lasso)
```

```
## Error: the class of "x" is "cv.glmnet" but expected the class to be one of
## "glmnet" "multnet"
```

```r
# Predict
prob_lasso <- predict(fit_lasso, as.matrix(leu_dat[,-1]), s = "lambda.1se", type='response')

# Get MSE
lasso.mse <- mean((as.matrix(leu_dat[,1]) - prob_lasso)^2)
lasso.mse
```

```
## [1] 0.00761202
```

```r
# extract optimal lambda
lambda_opt <- fit_lasso$lambda.1se

# manually plugging lambda into glmnet to get BEST MODEL
fit_lasso_2 <- glmnet(x = as.matrix(leu_dat[,-1]), y = as.matrix(leu_dat[,1]), lambda = lambda_opt)


# Make a function for above and bootstrap
foo <- function(x, y){
# Fit lasso model
mod <- cv.glmnet(type.measure = "class", x = as.matrix(x), y = as.matrix(y), alpha = 1, family = 'binom
# Get non-zero coefficients
non-zero <- names(fit_lasso_2$beta[, 1][fit_lasso_2$beta[, 1] > 1e-10])
}


# Run Bootstrap
dat.boot <- boot(dat, foo , R = B)
```

```
## Error in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one multinomial or binomial
```

```r
N = 100
mat <- matrix(0, ncol = N, nrow = ncol(leu_dat[,-1]))
rownames(mat) <- colnames((leu_dat[,-1]))
for (i in 1:N){
  # Fit lasso model
  mod <- cv.glmnet(type.measure = "class", x = as.matrix(leu_dat[,-1]), y = as.matrix(leu_dat[,1]), alp
  # Get non-zero coefficients
  non_zero <- as.vector(names(mod$beta[, 1][mod$beta[, 1] > 1e-10]))
  for(i in colnames(mat)[colnames(mat) %in% names(non_zero)]) {
    # fill these matrix columns with the designated values from your vector
    mat[ , i] <- 1
  }
  i = i+1
}
```

Problem 4 – Solution

(a)

```
# Function to generate a pair of correlated variables from lecture 6
gen.cor.pair <- function(r, n){ #r is target correlation, n is sample size
u <- rnorm(n, 0, sqrt(abs(r))) #temporary variable that is used for generating x1 and x2
x1 <- scale(u + rnorm(n, 0, sqrt(1 - abs(r))))
x2 <- scale(sign(r)*u + rnorm(n, 0, sqrt(1 - abs(r))))
cbind(x1, x2) #return matrix
}

# Testing function
n = 10000
r = c(-0.9, -0.5, 0, 0.5, 0.9)
r.est1 <- cor(gen.cor.pair(r = r[1], n = n))[[1, 2]]
r.est2 <- cor(gen.cor.pair(r = r[2], n = n))[[1, 2]]
r.est3 <- cor(gen.cor.pair(r = r[3], n = n))[[1, 2]]
r.est4 <- cor(gen.cor.pair(r = r[4], n = n))[[1, 2]]
```

(b)

```
n = 200
r = c(0, -0.5, 0.99, 0.999)
R = 50 #replicates of data set
r.est.1 <- replicate(R, gen.cor.pair(r = r[1], n = n)) #returns cor(x,y) for R indep. data sets
r.est.2 <- replicate(R, gen.cor.pair(r = r[1], n = n)) #returns cor(x,y) for R indep. data sets
r.est.3 <- replicate(R, gen.cor.pair(r = r[1], n = n)) #returns cor(x,y) for R indep. data sets
r.est.4 <- replicate(R, gen.cor.pair(r = r[1], n = n)) #returns cor(x,y) for R indep. data sets
```

JUNK CODE

dat_boot <- replicate(B, dat[sample(1:n, n, replace = T),][1,2]) require(boot) yourCVGLM<- cv.glmnet(type.measure = "class", y = as.matrix(leu_dat[,1]), x = as.matrix(leu_dat[,-1]), family="binomial", K=100) names(yourCVGLM$beta[,1][yourCVGLM$beta[, 1] > 1e-10])