HDS Exercise set 4

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)

tr.ind <- read.csv("E:/Dropbox/Important_Documents/Doctoral_Work/Courses/High Dimensional Stats/2019/We

# Train & Test dataset
data(Boston)
train <- Boston[tr.ind$X1,]
test <- Boston[-tr.ind$X1,]

# Predictor variables
x <- model.matrix(medv~., train)[,-1]

# Outcome variable
y <- train$medv
```

(a)

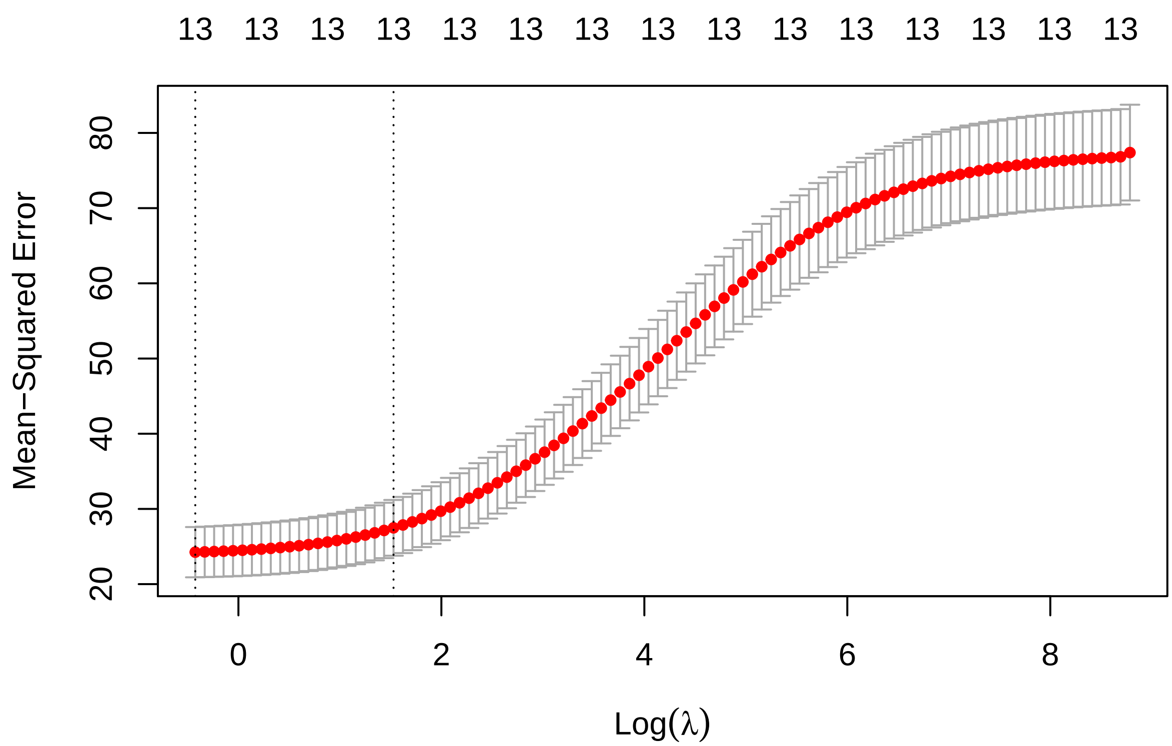
```r
# Finding the best lambda using cross-validation
cv_glmnet <- cv.glmnet(x, y, alpha = 0)

c(cv_glmnet$lambda.min, cv_glmnet$lambda.1se)
```

```
## [1] 0.6536359 4.6112717
```

```r
round(log(c(cv_glmnet$lambda.min, cv_glmnet$lambda.1se)), 2)
```

```
## [1] -0.43  1.53
```

```r
plot(cv_glmnet) # PLotting model
```



We have seen previously that traditional model selection methodsare often unstable and have low prediction accuracy, especially for high-dimensional data which often include many correlated predictor variables. For analyzing these type of data, penalized regression methods such as LASSO selection is much better as they can produce more stable models with higher prediction accuracy as evidenced by our ability to choose lamda which can minimize the regularized loss function, leading to minimizing multicollinearity issues

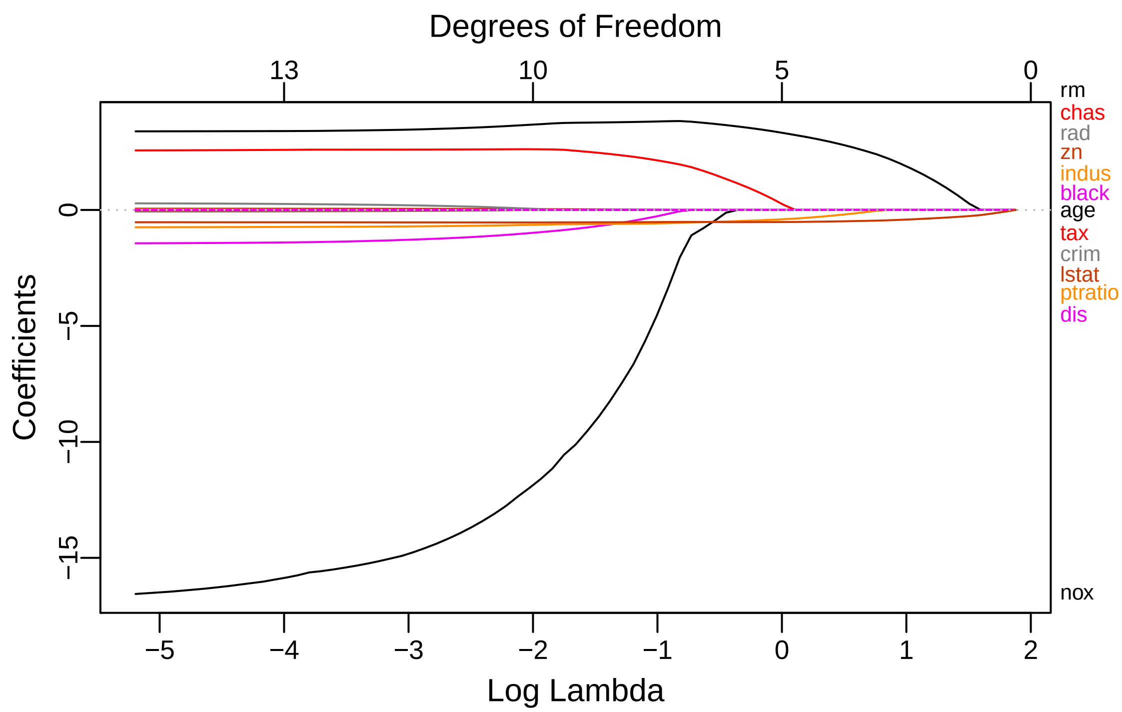
The lambda.min option refers to value of at the lowest CV error. The error at this value of is the average of the errors over the k folds and hence this estimate of the error is uncertain. The lambda.1se represents the value of in the search that was simpler than the best model (lambda.min), but which has error within 1 standard error of the best model.

In other words, using the value of lambda.1se as the selected value for results in a model that is slightly simpler than the best model but which cannot be distinguished from the best model in terms of error given the uncertainty in the k-fold CV estimate of the error of the best model.

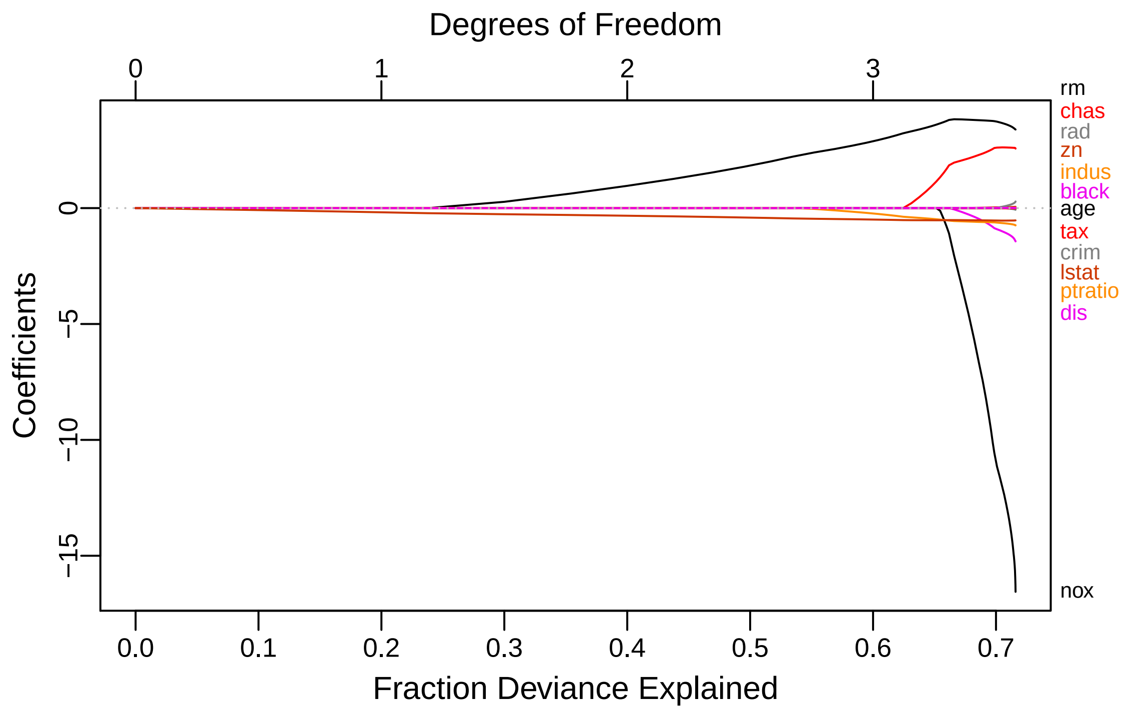
Hence, if we use lambda.min, we might get the best model that may be too complex and slightly overfitted. BUt lambda.1se gives us the simplest model that has comparable error to the best model given the uncertainty.

(b)

```r
# Running glmnet
fit <- glmnet(x, y, family = "gaussian", alpha = 1)

# PLotting coef vs log(lamda)
plot_glmnet(fit, xvar = "lambda", label = TRUE)
```




```r
plot_glmnet(fit, xvar = "dev", label = TRUE)
```




```r
# Actual coef from above model at some tuning parameter
coef(fit, s = 0.1)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  28.403161269
## crim         -0.016567429
## zn            0.040899530
## indus         .
## chas          2.613527714
## nox         -13.073951231
## rm            3.591125700
## age           .
## dis          -1.107719427
## rad           0.111105654
## tax          -0.005523527
## ptratio      -0.669325672
## black         0.007645459
## lstat        -0.538426624
```

```r
# Run lm
fit_lm <- lm(medv~., train)
summary(fit_lm)
```

```
##
```

```
## Call:
## lm(formula = medv ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.2569  -2.7731  -0.5727   1.4788  25.8668
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  35.600400   6.200853   5.741 2.11e-08 ***
## crim         -0.072705   0.053498  -1.359 0.175051
## zn            0.057290   0.016161   3.545 0.000449 ***
## indus         0.035552   0.080274   0.443 0.658138
## chas          2.550825   0.955445   2.670 0.007960 **
## nox         -16.904750   4.524228  -3.736 0.000219 ***
## rm            3.381526   0.538233   6.283 1.03e-09 ***
## age          -0.003004   0.016036  -0.187 0.851492
## dis          -1.452771   0.242733  -5.985 5.56e-09 ***
## rad           0.300110   0.084084   3.569 0.000410 ***
## tax          -0.013904   0.004780  -2.909 0.003871 **
## ptratio      -0.753162   0.158022  -4.766 2.80e-06 ***
## black         0.008903   0.003348   2.659 0.008212 **
## lstat        -0.530554   0.064581  -8.215 4.67e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.767 on 335 degrees of freedom
## Multiple R-squared:  0.716,  Adjusted R-squared:  0.7049
## F-statistic: 64.95 on 13 and 335 DF,  p-value: < 2.2e-16
```

```r
vif(fit_lm)
```

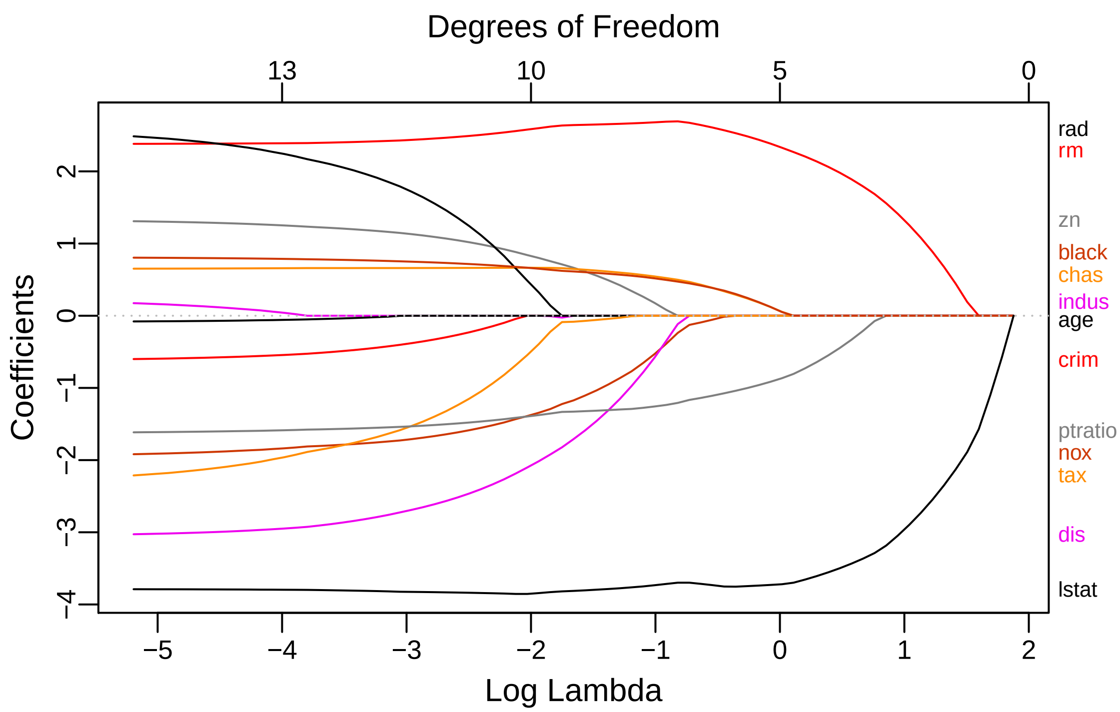
```
##     crim       zn    indus     chas      nox       rm      age      dis
## 2.281083 2.434418 4.580449 1.101435 4.451223 2.027185 3.234339 4.103866
##      rad      tax  ptratio    black    lstat
## 8.248746 9.766272 1.792897 1.305496 3.356000
```

We see that age and indus have been shrinked to zero and the ones remaining are included in the model. In lm(), we see that lstat, dis, rm and ptratio are amongst the most highly signififcant variables. However, in glmnet() we see that "dis" is not to be seen at all despite having 5.56e-09 p-value.

(c)

```r
Boston$chas<-as.numeric(Boston$chas)

#Standardize covariates before fitting
Boston.X.std<- scale(dplyr::select(Boston,-medv))
X.train<- as.matrix(Boston.X.std)[tr.ind$X1,]
X.test<-  as.matrix(Boston.X.std)[-tr.ind$X1,]
Y.train<- Boston[tr.ind$X1, "medv"]
Y.test<- Boston[-tr.ind$X1, "medv"]
```

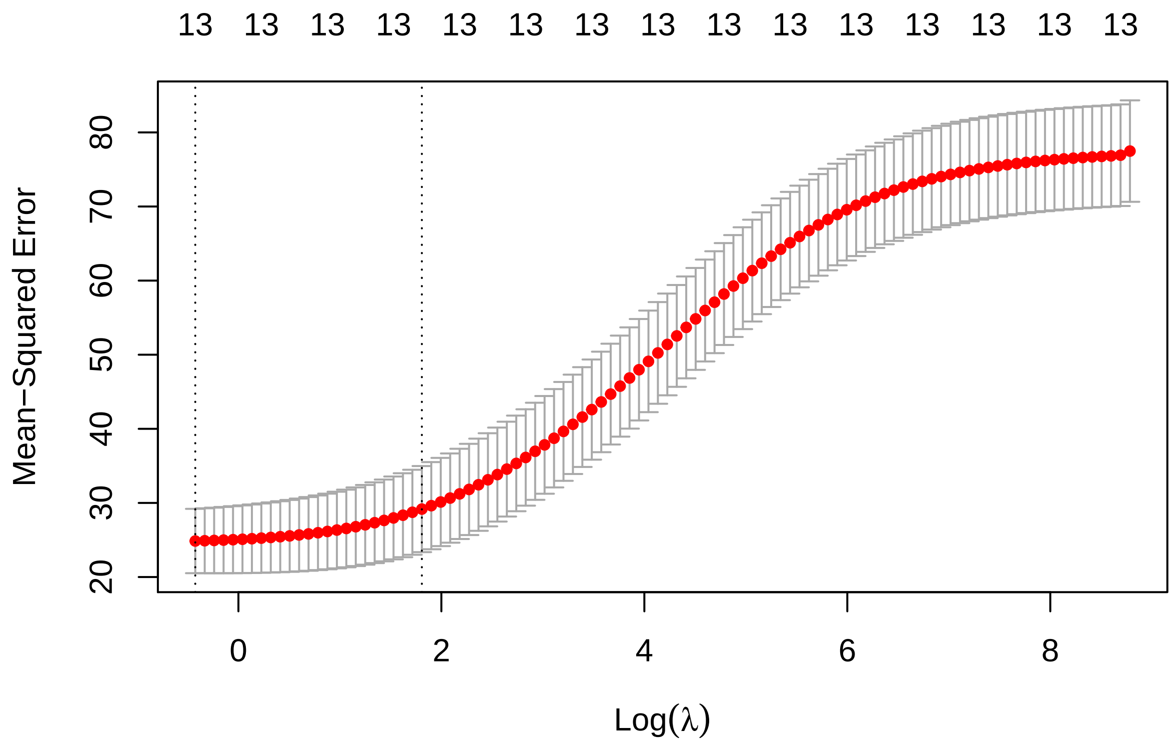
```r
fit_std<- glmnet(x=X.train, y=Y.train, family = "gaussian", alpha = 1)
plot_glmnet(fit_std, xvar = "lambda", label=TRUE)
```



Lasso regression puts constraints on the size of the coefficients associated to each variable. However, this value will depend on the magnitude of each variable. Hence the need to center and reduce, or standardize, the variables. The result of centering the variables means that there is no longer an intercept.We see more variables pop in the selected variables in glmnet() model, and due to this centering we can rank the coefficient importance by the relative magnitude of post-shrinkage coefficient estimates.

(d)

```r
# Ridge Reg

# Cross Validation to find lamda.min
cv1<- cv.glmnet(x=X.train, y=Y.train, family = "gaussian", alpha = 0, nfolds = 10)
plot(cv1)
```



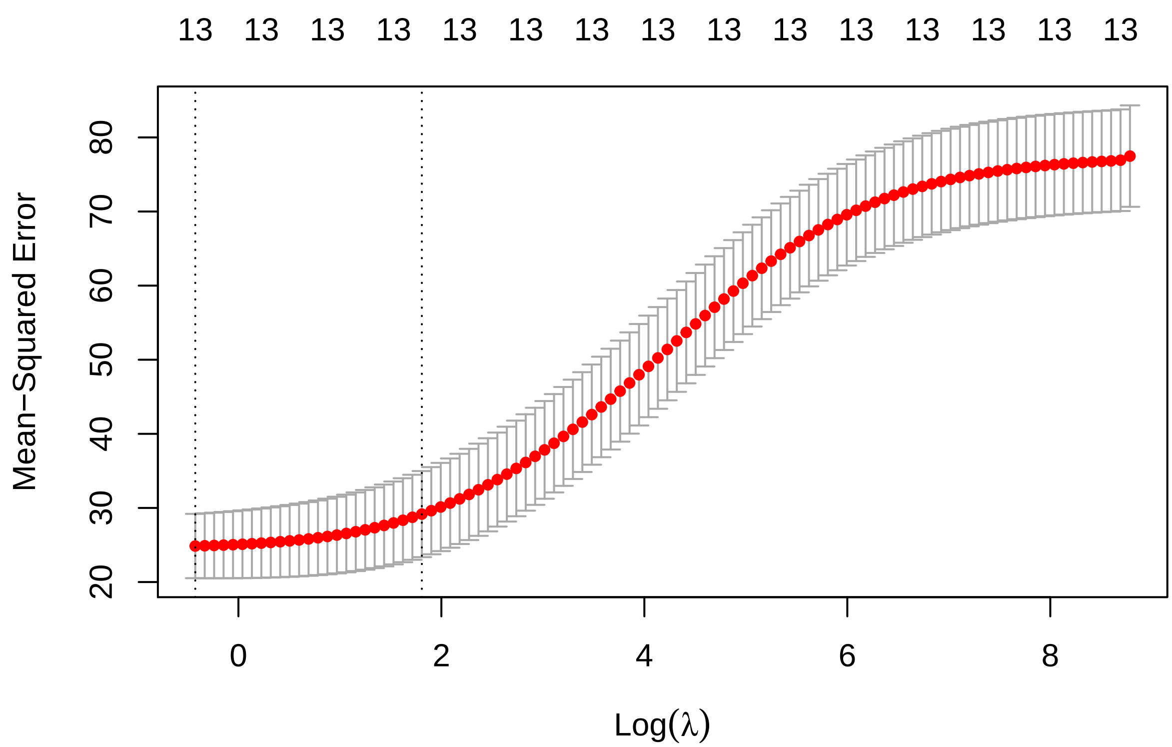

```
# Predictions
pred1.min<- predict(fit_std, newx = X.test, s = cv1$lambda.min)

# MSPE (prediction error)
mean((Y.test-pred1.min)^2)
```

```
## [1] 31.64782
```

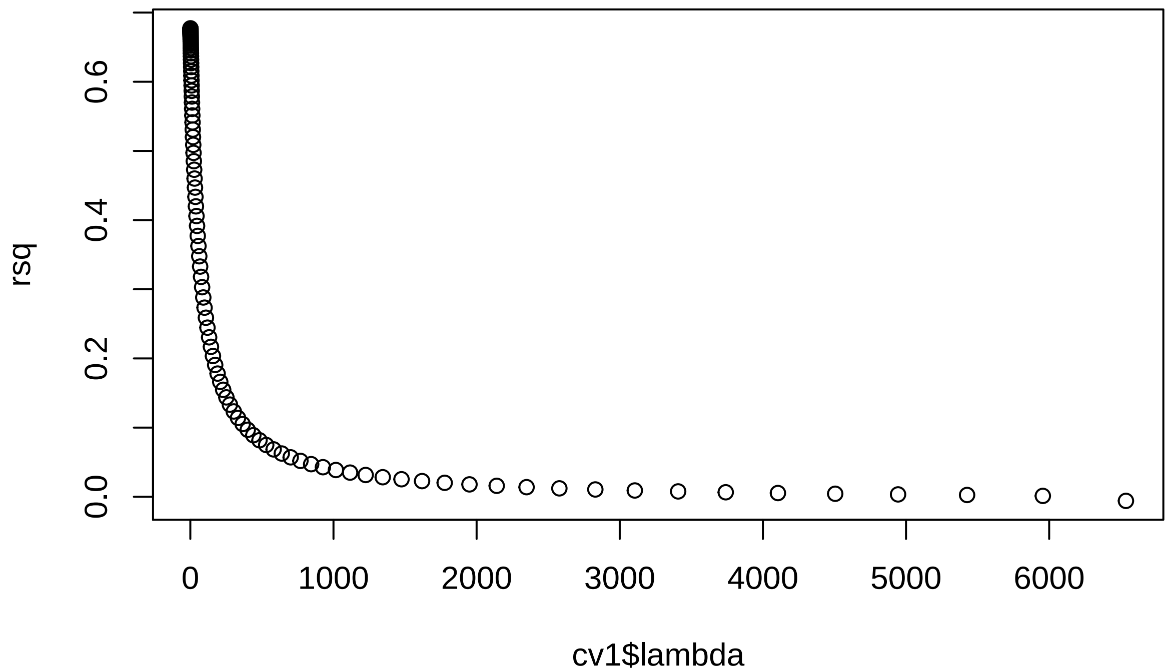
```
# Deviance explained after standardization
plot(cv1, xvar = "dev", label = TRUE)
```

```
## Warning in plot.window(...): "xvar" is not a graphical parameter

## Warning in plot.window(...): "label" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "xvar" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "label" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "xvar" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "label" is not
## a graphical parameter
```

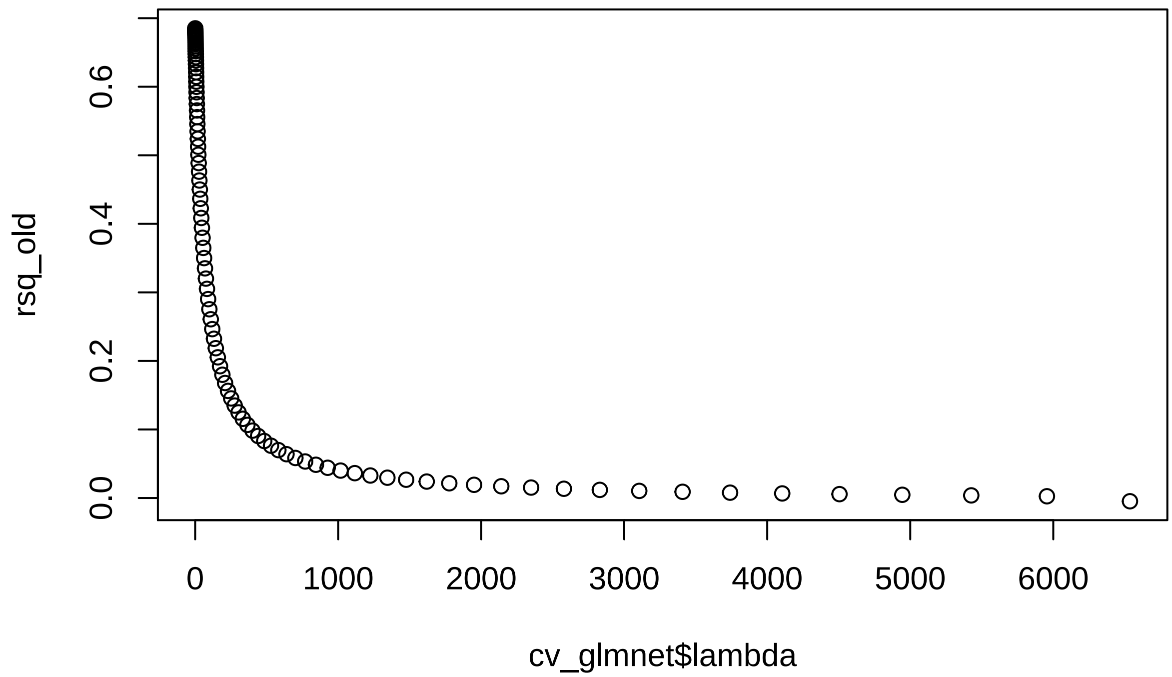
```
## Warning in axis(side = side, at = at, labels = labels, ...): "xvar" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "label" is not
## a graphical parameter

## Warning in box(...): "xvar" is not a graphical parameter

## Warning in box(...): "label" is not a graphical parameter

## Warning in title(...): "xvar" is not a graphical parameter

## Warning in title(...): "label" is not a graphical parameter
```




```r
# Standardized vars
rsq = 1 - cv1$cvm/var(Y.train)
plot(cv1$lambda,rsq)
```




```r
# Non-standardized vars
rsq_old = 1 - cv_glmnet$cvm/var(y)
plot(cv_glmnet$lambda,rsq_old)
```

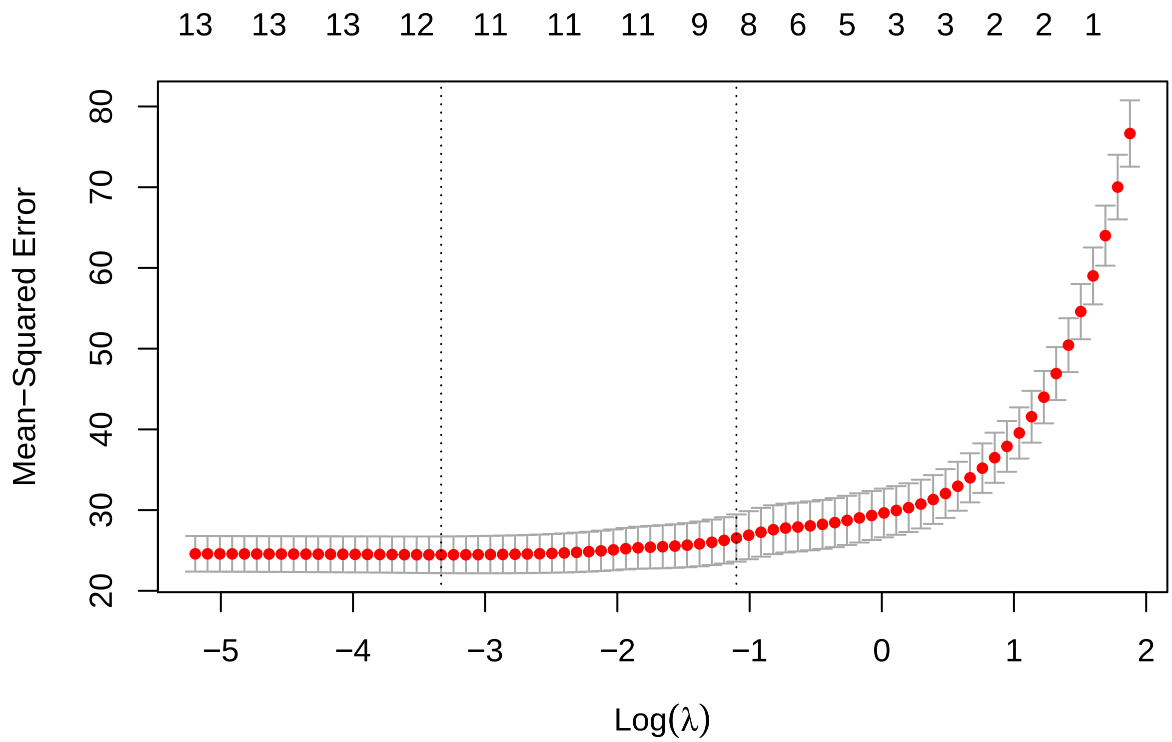


Variance explained is around 70% with standarized variables compared to around 60% or lesser with original ones.

Problem 2 – Solution

(e)

```
# LASSO

library(coefplot)

# Cross Validation to find lamda.min
cv2<- cv.glmnet(x=X.train, y=Y.train, family = "gaussian", alpha = 1, nfolds = 10)
plot(cv2)
```




```r
# Predictions
pred2.1se<- predict(fit_std, newx = X.test, s=cv2$lambda.1se)

# MSPE (prediction error)
mean((Y.test-pred2.1se)^2)
```

```
## [1] 28.0744
```

The MSE is smaller in lamda.min model than lamda.1se

(f).

```r
# Run lm
fit_lm <- lm(medv~., train)
mse1 <- mean((test$medv - predict.lm(fit_lm, test)) ^ 2)
mse1
```

```
## [1] 23.21952
```

```r
# Using lambda.min (Ridge)
pred1<- predict(fit_std, newx = X.test, s=cv1$lambda.min)
mse2 <- mean((Y.test-pred1)^2)
mse2
```

```
## [1] 31.64782
```

```r
# Using lambda.1se (LASSO)
pred2<- predict(fit_std, newx = X.test, s=cv2$lambda.1se)
mse3 <- mean((Y.test-pred2)^2)
mse3
```

```
## [1] 28.0744
```

LM has lowest mse, followed by ridge and then LASSO

(g).

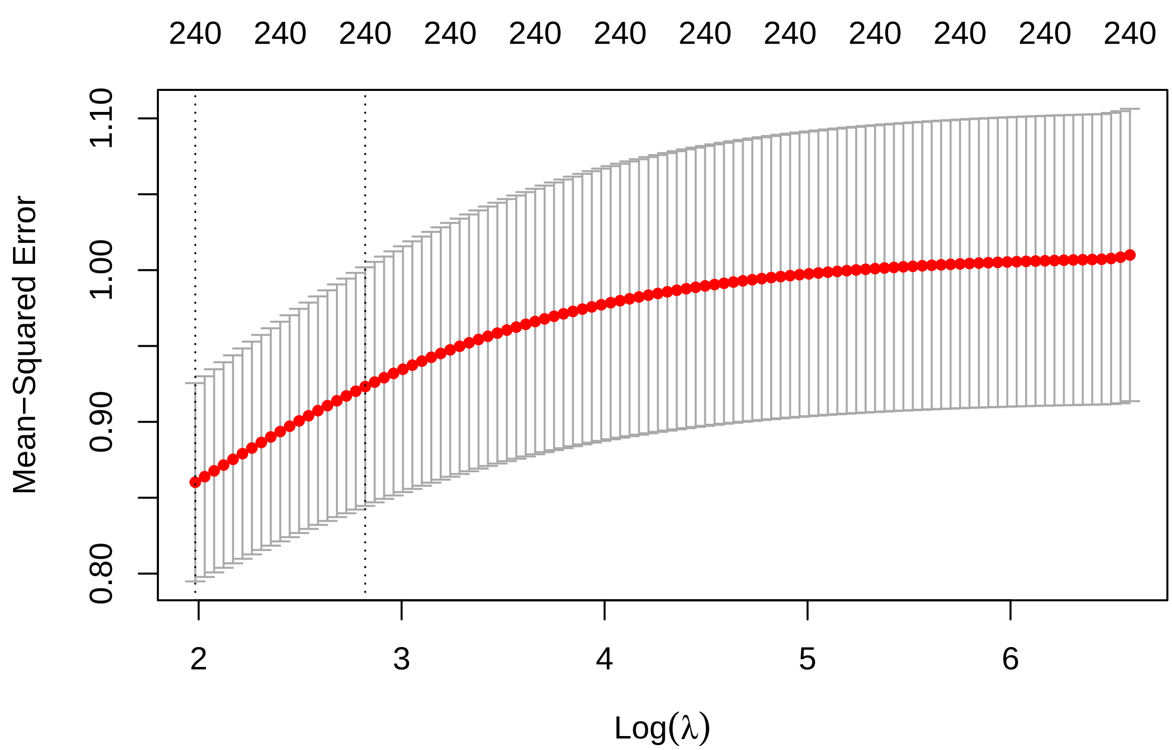
```r
# Using lamda.min (ridge)
cv1<- cv.glmnet(x=X.train, y=Y.train, family = "gaussian", alpha = 0, nfolds = 10) # find lambda
lambda_min <- glmnet(x=X.train, y=Y.train, alpha = 0, lambda= cv1$lambda.min) # do ridge reg
coef_min <- data.frame(as.matrix(coef(lambda_min, s = "lambda_min")))

# Using lamda.1se (LASSO)
cv2<- cv.glmnet(x=X.train, y=Y.train, family = "gaussian", alpha = 1, nfolds = 10) # find lambda
lambda_1se <- glmnet(x=X.train, y=Y.train, alpha = 1, lambda = cv2$lambda.1se) # do LASSO
coef_1se <- data.frame(as.matrix(coef(lambda_1se, s = "lambda.1se")))

# LM
std_data <- data.frame(cbind(X.train, Y.train))
fit_lm2 <- lm(Y.train~., std_data)
coef_lm <- fit_lm2$coefficients # from before

# Coefficient table
coef_tab <- data.frame(ridge = coef_min, lasso = coef_1se, linear = coef_lm) %>% rename(ridge = X1, las
coef_tab
```

```
##                  ridge      lasso      linear
## (Intercept) 22.4354673 22.4798224 22.46853527
## crim        -0.4916785  0.0000000 -0.62537611
## zn           0.9475160  0.0000000  1.33613876
## indus       -0.2310895  0.0000000  0.24389761
## chas         0.7216320  0.3391200  0.64789422
## nox         -1.3652870 -0.0145085 -1.95888316
## rm           2.5735501  2.5709178  2.37591808
## age         -0.2181462  0.0000000 -0.08457213
## dis         -2.2604464  0.0000000 -3.05911429
## rad          1.2479972  0.0000000  2.61313858
## tax         -1.0317161  0.0000000 -2.34331835
## ptratio     -1.4388625 -1.0749887 -1.63055461
## black        0.7978840  0.3481508  0.81280178
## lstat       -3.3308022 -3.7498080 -3.78871520
```

Problem 3 – Solution

(a).

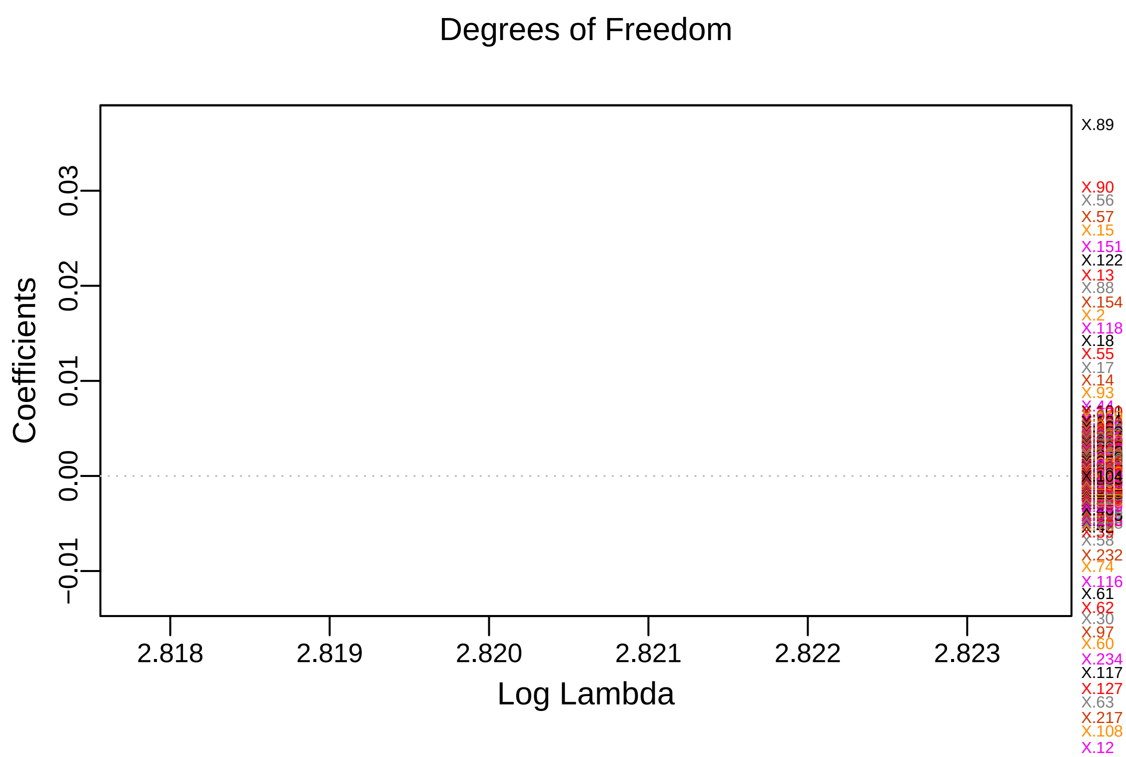
```r
new_dat <- read.csv("E:/Dropbox/Important_Documents/Doctoral_Work/Courses/High Dimensional Stats/2019/W

# Train & Test dataset
train <- new_dat[which(new_dat$train=='1'), ]
test <- new_dat[which(new_dat$train=='0'), ]

#Standardize covariates before fitting
train_std <- scale(dplyr::select(train,-c(y1, y2, train)))
test_std <- scale(dplyr::select(test,-c(y1, y2, train)))

# Outcome vars
Y1.train<- train$y1
Y2.train<- train$y2
Y1.test<- train$y1
Y2.test<- train$y2
Y.test<- Boston[-tr.ind$X1, "medv"]

# Using lamda.min (ridge)
cv.y1_ridge <- cv.glmnet(x=train_std, y=Y1.train, family = "gaussian", alpha = 0, nfolds = 10) # find l
y1_ridge <- glmnet(x=train_std, y=Y1.train, alpha = 0, lambda= cv.y1_ridge$lambda.1se) # do ridge reg
plot(cv.y1_ridge) # CV plot
```



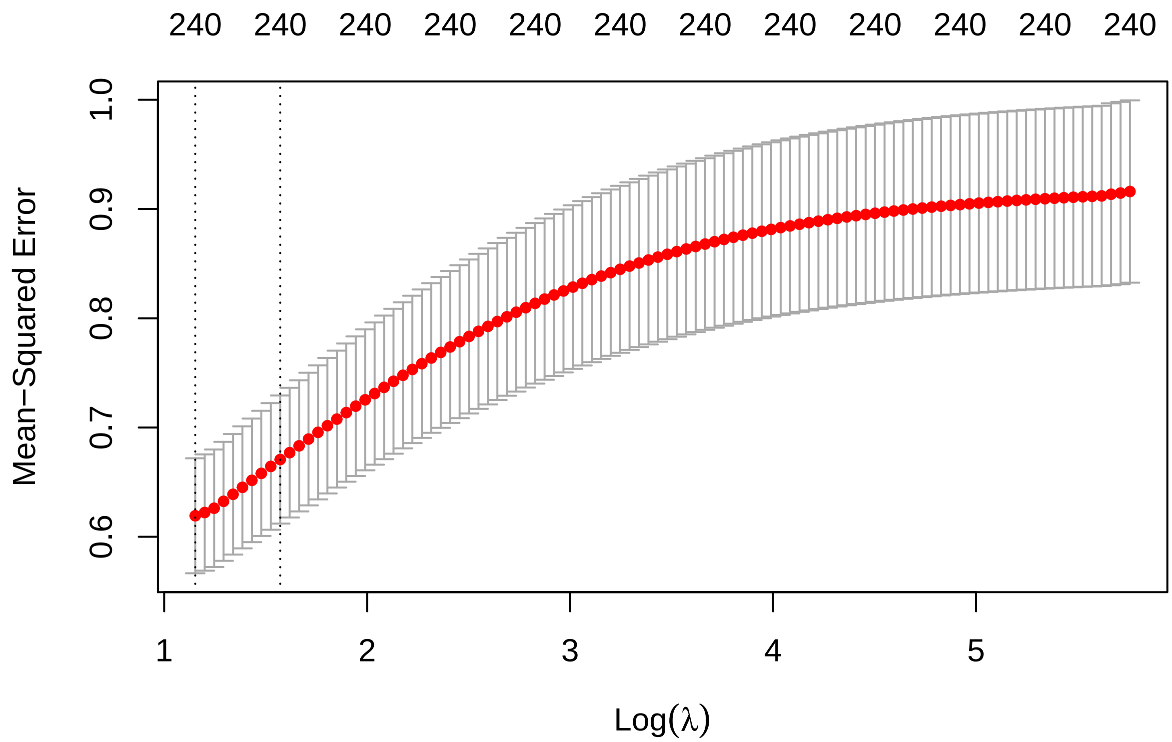

```r
plot_glmnet(y1_ridge, xvar = "lambda", label=TRUE) # Coef plot
```

```
## Warning in TeachingDemos::spread.labs(beta[iname, ncol(beta)], mindiff =
```

```
## 1.2 * : Maximum iterations reached
```

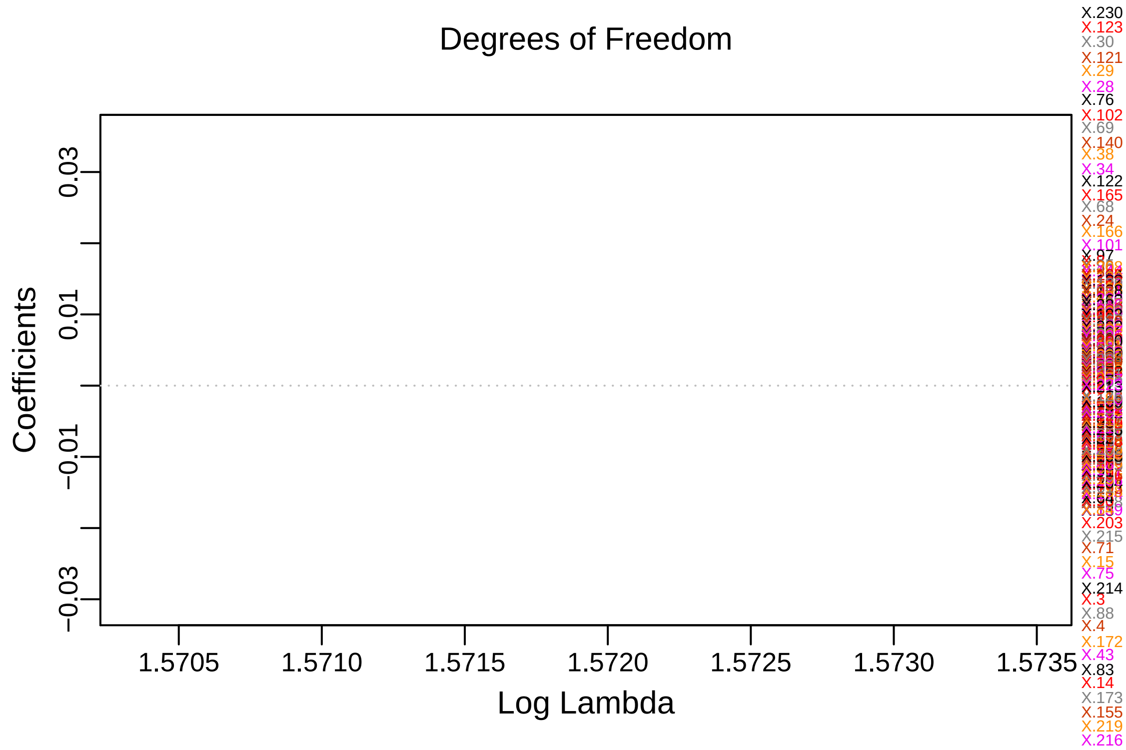



```
cv.y2_ridge <- cv.glmnet(x=train_std, y=Y2.train, family = "gaussian", alpha = 0, nfolds = 10) # find l
y2_ridge <- glmnet(x=train_std, y=Y2.train, alpha = 0, lambda= cv.y2_ridge$lambda.1se) # do ridge reg
plot(cv.y2_ridge) # CV plot
```

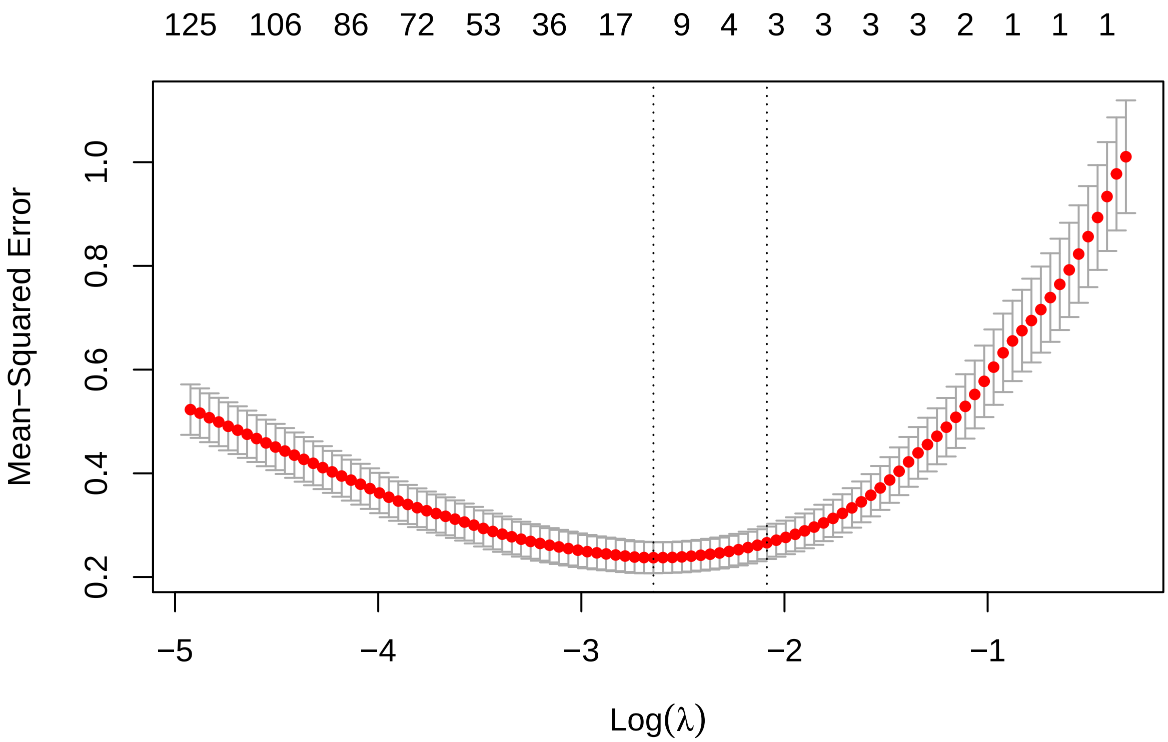



```r
plot_glmnet(y2_ridge, xvar = "lambda", label=TRUE) # Coef plot
```

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

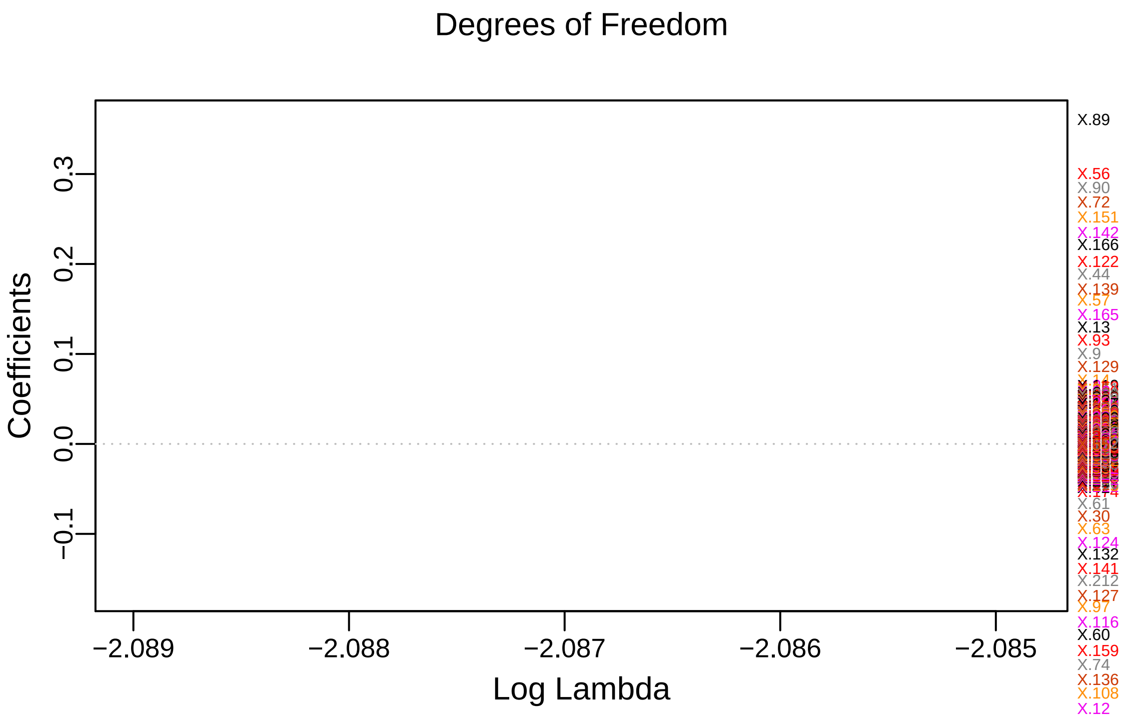



```r
# Using lamda.1se (LASSO)
cv.y1_lasso <- cv.glmnet(x=train_std, y=Y1.train, family = "gaussian", alpha = 1, nfolds = 10) # find l
y1_lasso <- glmnet(x=train_std, y=Y1.train, alpha = 0, lambda= cv.y1_lasso$lambda.1se) # do ridge reg
plot(cv.y1_lasso) # CV plot
```

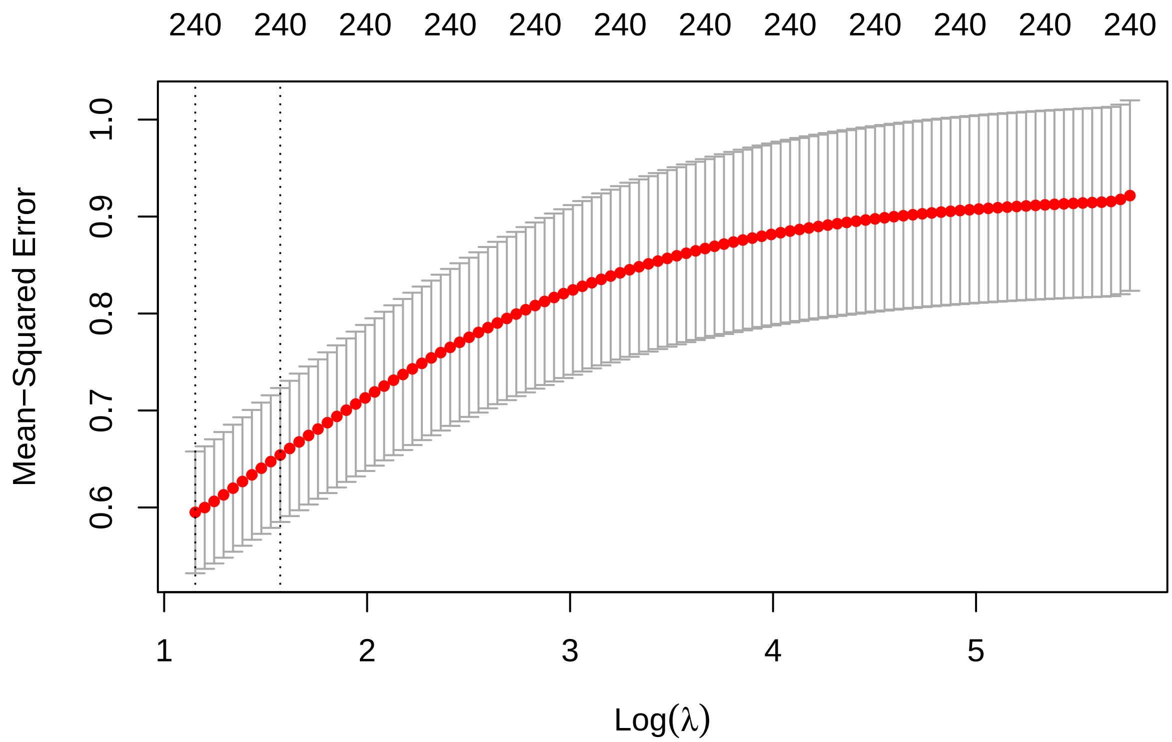



```r
plot_glmnet(y1_lasso, xvar = "lambda", label=TRUE) # Coef plot
```

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

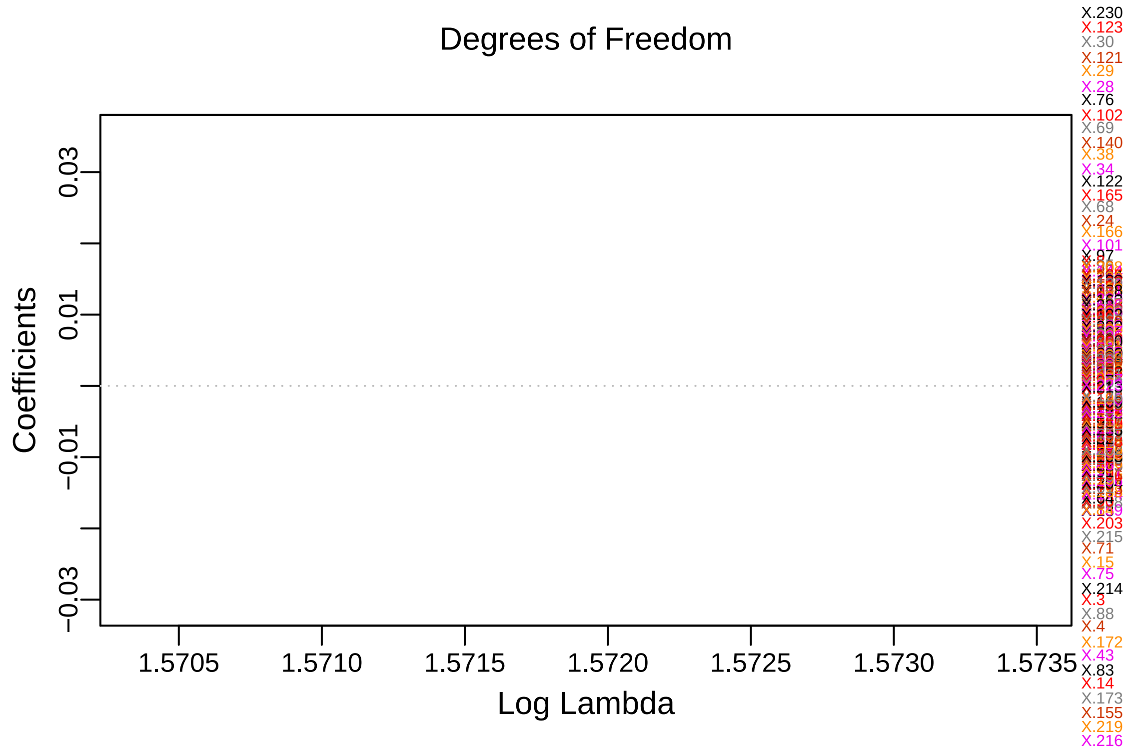



```
cv.y2_ridge <- cv.glmnet(x=train_std, y=Y2.train, family = "gaussian", alpha = 0, nfolds = 10) # find l
y2_ridge <- glmnet(x=train_std, y=Y2.train, alpha = 0, lambda= cv.y2_ridge$lambda.1se) # do ridge reg
plot(cv.y2_ridge) # CV plot
```




```r
plot_glmnet(y2_ridge, xvar = "lambda", label=TRUE) # Coef plot
```

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



Junk COde

imp <- as.data.frame(varImp(cv_glmnet)) imp <- data.frame(overall = imp$Overall, names = rownames(imp))imp[order(imp$overall,decreasing = T),]

varImp <- function(object, lambda = NULL, …) {

skipping a few lines

beta <- predict(object, s = lambda, type = "coef") if(is.list(beta)) { out <- do.call("cbind", lapply(beta, function(x) x[,1])) out <- as.data.frame(out) } else out <- data.frame(Overall = beta[,1]) out <- abs(out[rownames(out) != "(Intercept)",,drop = FALSE]) out }

Compare the models and see which variables agree

var_step = names(fit_lm$coefficients)[-1]var_lasso = colnames(train)[which(coef(fit, s = cv.lasso$lambda.min)!=0)-1] intersect(var_step,var_lasso)