Lasso estimation in multiple linear regression

Gregoire Gasparini, Aurora Hofman, Sarah Musiol, Beatriu Tort

03 de marzo de 2020

1. Lasso for the Boston Housing data

The Boston House-price dataset concerns housing values in 506 suburbs of Boston corresponding to year 1978. They are available here: https://archive.ics.uci.edu/ml/datasets/Housing

The Boston House-price corrected dataset (available in boston.Rdata) con- tains the same data (with some corrections) and it also includes the UTM coor- dinates of the geographical centers of each neighborhood.

1.1 Lasso estimation using the package 'glmnet'

After loading the right package, the response and explanatory variables from the Boston Housing data are set.

```
#install.packages("glmnet")
library(glmnet)
```

```
## Loading required package: Matrix
## Loaded glmnet 3.0-2
```

```
library(Matrix)

boston <- load("boston.Rdata")

response <- "CMEDV"

explanatory <- c("CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B"

# glmnet cannot handle factors -> "CHAS" is a factor
boston.c$CHAS <- as.numeric(boston.c$CHAS)
```

Fitting the Lasso Regression model.

```
lasso_boston <- glmnet(x = as.matrix(boston.c[, explanatory]),
                y = boston.c$CMEDV,
                alpha = 1,     # specifying alpha = 1: Lasso Regression
                standardize = FALSE,
                intercept = FALSE)
lasso_boston$beta
```

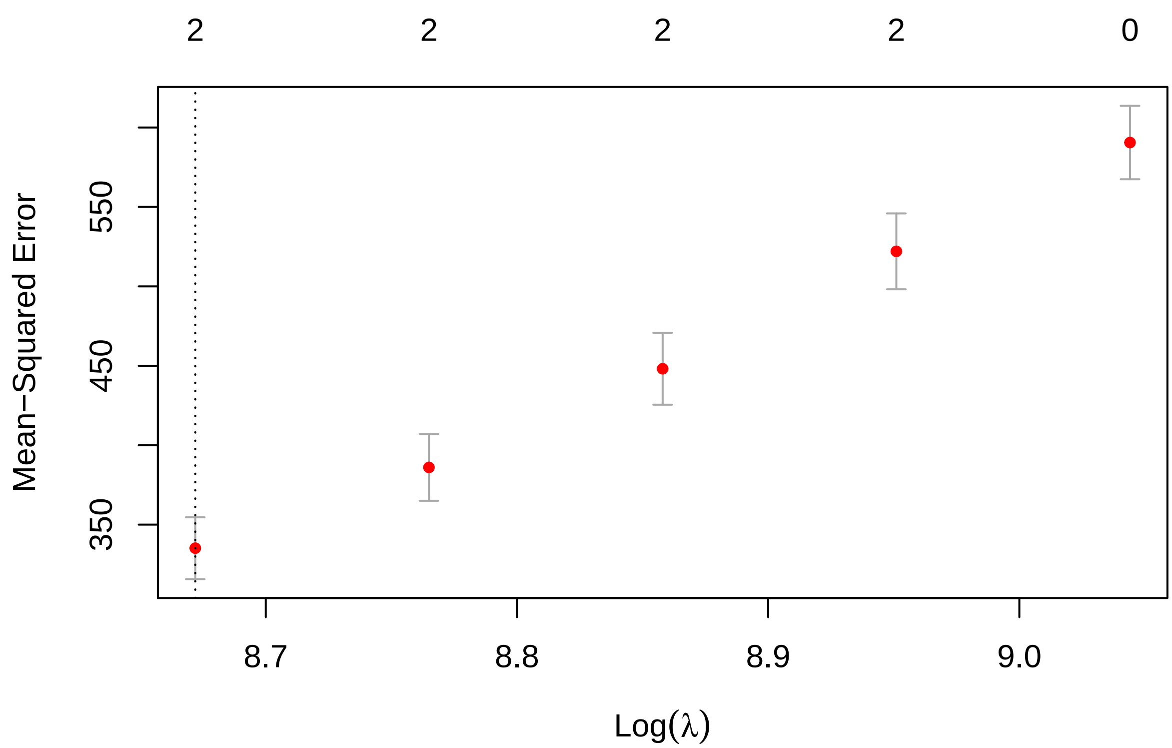
```
## 13 x 5 sparse Matrix of class "dgCMatrix"
##         s0          s1          s2          s3          s4
## CRIM     . .           .           .           .
## ZN       . .           .           .           .
## INDUS    . .           .           .           .
## CHAS     . .           .           .           .
```

```
## NOX      . .           .           .           .
## RM       . .           .           .           .
## AGE      . .           .           .           .
## DIS      . .           .           .           .
## RAD      . .           .           .           .
## TAX      . 0.002623633 0.002307334 0.002022783 0.001760304
## PTRATIO  . .           .           .           .
## B        . 0.001737741 0.007119139 0.012018732 0.016486342
## LSTAT    . .           .           .           .
```

```r
cv_lasso_boston <- cv.glmnet(x = as.matrix(boston.c[, explanatory]),
                y = boston.c$CMEDV,
                alpha = 1,     # specifying alpha = 1: Lasso Regression
                standardize = FALSE,
                intercept = FALSE)
cv_lasso_boston
```

```
##
## Call:  cv.glmnet(x = as.matrix(boston.c[, explanatory]), y = boston.c$CMEDV,      alpha = 1, standard
##
## Measure: Mean-Squared Error
##
##     Lambda Measure    SE Nonzero
## min   5837   335.2 19.47       2
## 1se   5837   335.2 19.47       2
```

```r
plot(cv_lasso_boston)
```



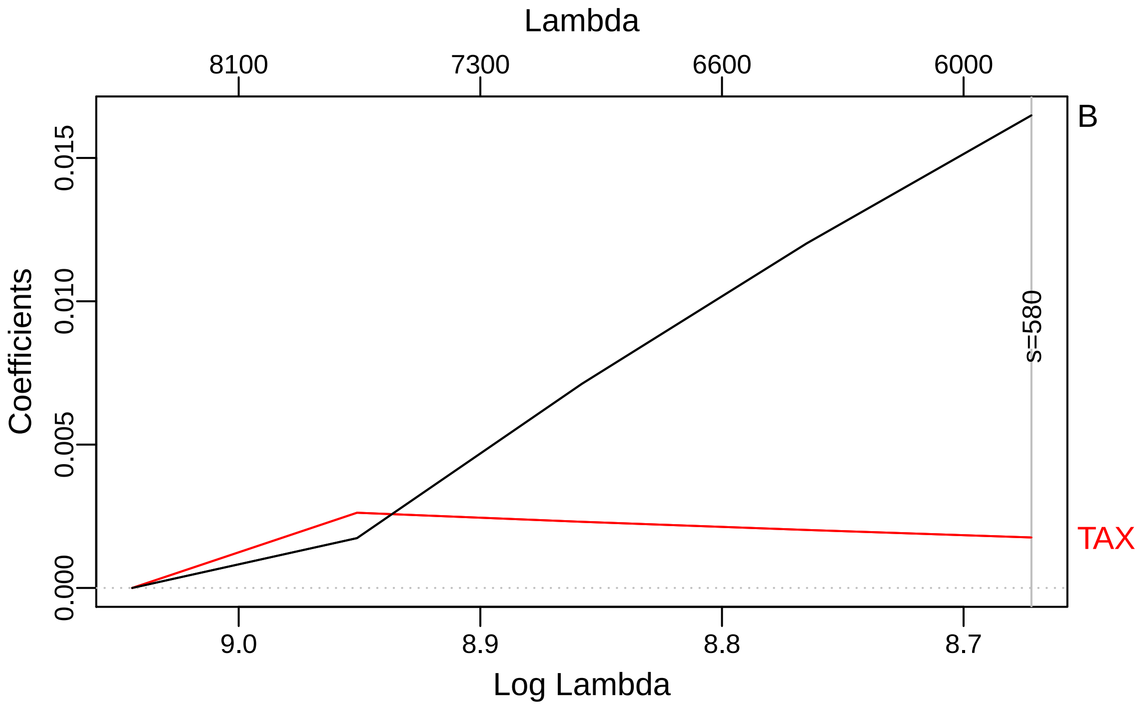

```r
library(plotmo) #plot glmnet with coefficient names
```

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

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

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

```r
plot_glmnet(lasso_boston, s=cv_lasso_boston$lambda.min)
```



The Lasso Estimation does variable selection automatically. This results in a model with two non-zero explanatory variables. The variables affecting the model are the full-value property tax rates ('TAX') and the proportion of blacks by town ('B').

1.2 Ridge Regression model using glmnet

We fit the dataset using glmnet and ridge regression and plot the results. Task description:

```r
ridge_boston <- glmnet(x = as.matrix(boston.c[, explanatory]),
                y = boston.c$CMEDV,
                alpha = 0,     # specifying alpha = 0: Ridge Regression
                standardize = FALSE,
                intercept = FALSE)
ridge_boston$beta
```

```
## 13 x 5 sparse Matrix of class "dgCMatrix"
##                   s0           s1           s2           s3           s4
## CRIM    5.121190e-35 3.129122e-05 3.217069e-05 3.283579e-05 3.323520e-05
## ZN      3.363982e-34 3.267988e-04 3.533479e-04 3.817224e-04 4.120141e-04
## INDUS   2.226496e-34 1.816651e-04 1.932905e-04 2.051945e-04 2.173131e-04
## CHAS    2.474348e-35 2.191053e-05 2.349778e-05 2.516203e-05 2.690234e-05
## NOX     1.216239e-35 1.049404e-05 1.122691e-05 1.199023e-05 1.278256e-05
## RM      1.475444e-34 1.314780e-04 1.410854e-04 1.511744e-04 1.617426e-04
## AGE     1.462027e-33 1.242872e-03 1.327772e-03 1.415833e-03 1.506813e-03
## DIS     9.122115e-35 8.292515e-05 8.914248e-05 9.569996e-05 1.026014e-04
## RAD     1.862984e-34 1.464932e-04 1.552873e-04 1.641726e-04 1.730775e-04
## TAX     8.553691e-33 7.129210e-03 7.601712e-03 8.088988e-03 8.589147e-03
## PTRATIO 4.098485e-34 3.567226e-04 3.819446e-04 4.082736e-04 4.356713e-04
```

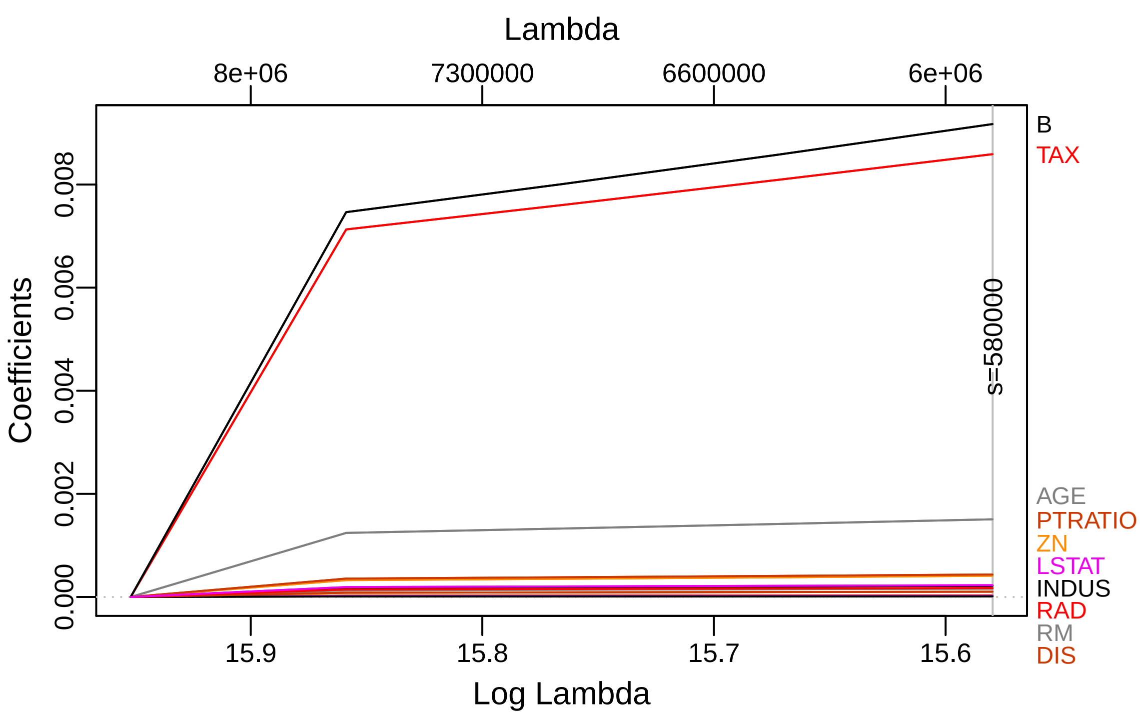
```
## B       8.399607e-33 7.464300e-03 8.007256e-03 8.576883e-03 9.172900e-03
## LSTAT   2.389677e-34 1.922832e-04 2.042857e-04 2.165093e-04 2.288740e-04
```

The ridge regression of boston data using cv.glmnet

```r
cv_ridge_boston <- cv.glmnet(x = as.matrix(boston.c[, explanatory]),
                y = boston.c$CMEDV,
                alpha = 0,     # specifying alpha = 0: Ridge Regression
                standardize = FALSE,
                intercept = FALSE)
cv_ridge_boston$glmnet.fit$beta
```

```
## 13 x 5 sparse Matrix of class "dgCMatrix"
##                   s0           s1           s2           s3           s4
## CRIM    5.121190e-35 3.129122e-05 3.217069e-05 3.283579e-05 3.323520e-05
## ZN      3.363982e-34 3.267988e-04 3.533479e-04 3.817224e-04 4.120141e-04
## INDUS   2.226496e-34 1.816651e-04 1.932905e-04 2.051945e-04 2.173131e-04
## CHAS    2.474348e-35 2.191053e-05 2.349778e-05 2.516203e-05 2.690234e-05
## NOX     1.216239e-35 1.049404e-05 1.122691e-05 1.199023e-05 1.278256e-05
## RM      1.475444e-34 1.314780e-04 1.410854e-04 1.511744e-04 1.617426e-04
## AGE     1.462027e-33 1.242872e-03 1.327772e-03 1.415833e-03 1.506813e-03
## DIS     9.122115e-35 8.292515e-05 8.914248e-05 9.569996e-05 1.026014e-04
## RAD     1.862984e-34 1.464932e-04 1.552873e-04 1.641726e-04 1.730775e-04
## TAX     8.553691e-33 7.129210e-03 7.601712e-03 8.088988e-03 8.589147e-03
## PTRATIO 4.098485e-34 3.567226e-04 3.819446e-04 4.082736e-04 4.356713e-04
## B       8.399607e-33 7.464300e-03 8.007256e-03 8.576883e-03 9.172900e-03
## LSTAT   2.389677e-34 1.922832e-04 2.042857e-04 2.165093e-04 2.288740e-04
```

```r
plot_glmnet(ridge_boston, s=cv_ridge_boston$lambda.min)
```



As one can see rigde regression has a lot more non-zero expenatory variables than Lasso. However a lot of them are very close to zero. The one with the largest beta coefficients are the same as for Lasso namely TAX and B.

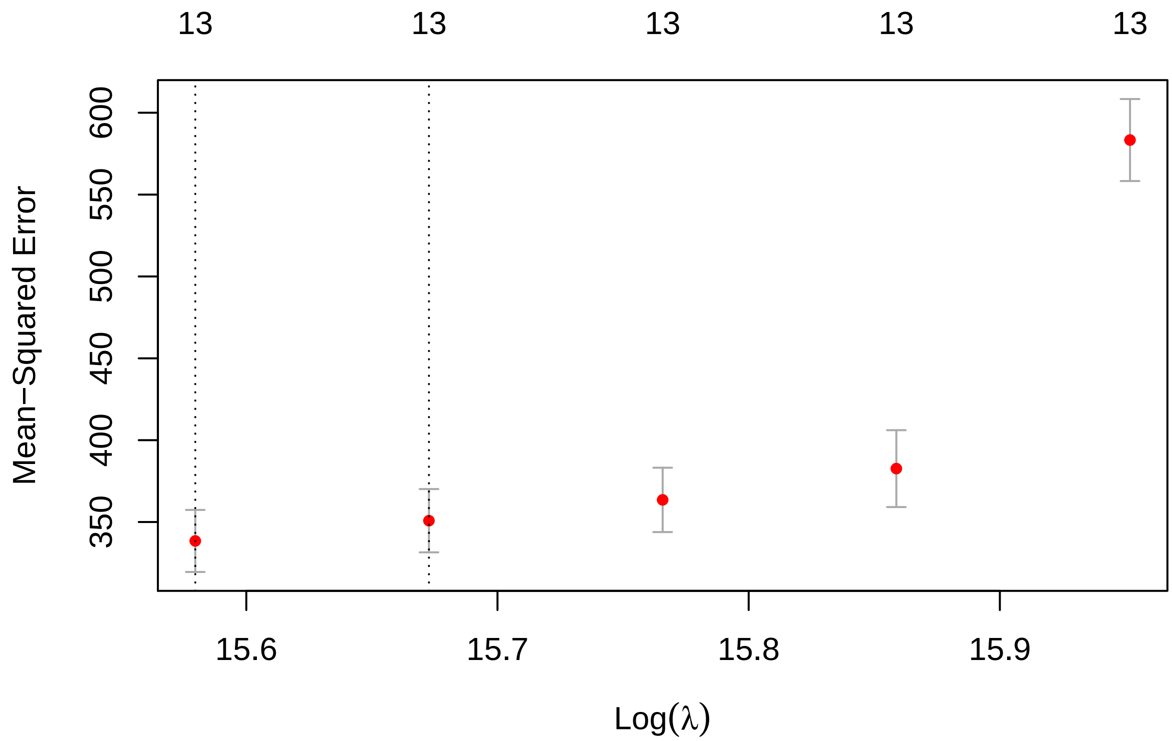
The 10-fold-function from previous assignment.

```
PMSE_k_fold <- function(X_t, Y_t, lambda.v, k=10){

  n_X_t <- dim(X_t)[1]
  p_X_t <- dim(X_t)[2] #length of columns
  n_subset <- as.integer((n_X_t)/k)+ 1
  group <- rep(seq(1,k), times = n_subset)
  group <- group[1:n_X_t]
  group_random <- sample(group)
  X_t_group <- cbind(X_t, group_random)
  Y_t_group <- cbind(Y_t, group_random)

#now we can start:

  PMSE <- list()

  for (la in 1:n_lambdas){

    lambda <- lambda.v[la]

    y_hat <- list()
    beta <- list()
    h <- list()
    y <- list()

    for (l in 1:k){

      new_X_t_val <- subset(X_t_group, group_random==l)[ ,1:p_X_t]
      new_X_t_test <- subset(X_t_group, group_random!=l)[ ,1:p_X_t]

      new_Y_t_val <- subset.matrix(Y_t_group, group_random==l)[,1]
      new_Y_t_test <- subset.matrix(Y_t_group, group_random!=l)[,1]

      p_new <- dim(new_X_t_test)[2]
      p_new_v <- dim(new_X_t_val)[2]

      beta[[l]] <- solve(t(new_X_t_test)%*%new_X_t_test + lambda*diag(1,p_new))%*% t(new_X_t_test)%*%(n

      H_val <- new_X_t_val%*%solve(t(new_X_t_val)%*%new_X_t_val + (lambda+1e-13)*diag(1,p_new_v))%*% t(

      # singular matrix for lambda = 0 -> trick: add a very small number

      y_hat[[l]] <-  (new_X_t_val)%*%beta[[l]]
      h[[l]] <- diag(H_val)
      y[[l]] <- new_Y_t_val


  }

  y_hat <- c(do.call(rbind, y_hat))
  beta <- c(do.call(cbind, beta))
  h <- c(do.call(rbind, h))
  y <- c(do.call(rbind, y))
```

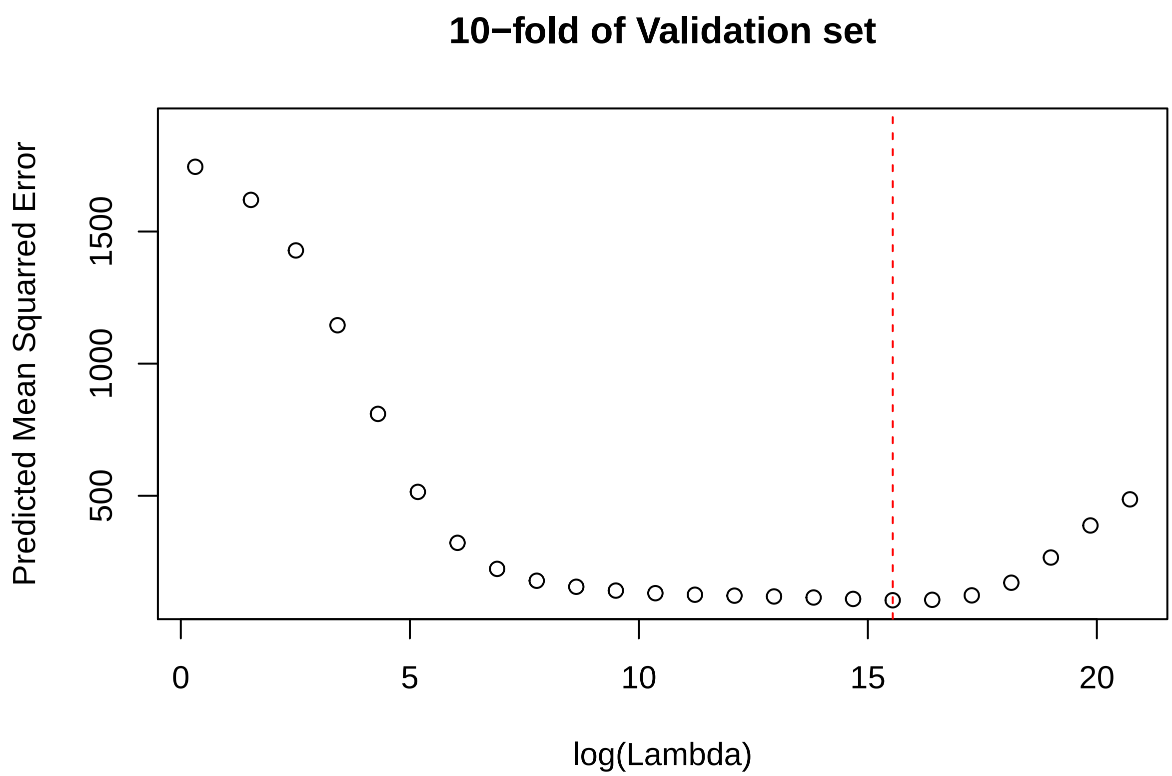
```r
  PMSE[[la]] <- 1/n_X_t * sum(((y-y_hat)/(1-h))^2)

  }

return(PMSE)

}


lambda.max = 1e9
n_lambdas <- 25
lambda.v <- exp(seq(0,log(lambda.max+1),length=n_lambdas))-1
```

Comparing the ridge regression using R cv_glmnet and our own function.

```r
plot(cv_ridge_boston)
```




```r
PMSE_val_10 <- PMSE_k_fold(X_t = as.matrix(boston.c[, explanatory]), Y_t = boston.c$CMEDV, lambda.v = la

PMSE_min <- min(unlist(PMSE_val_10))
min <- which.min(PMSE_val_10)

plot(log(lambda.v), PMSE_val_10, ylab = "Predicted Mean Squarred Error", xlab = "log(Lambda)", main = "
abline(v=log(lambda.v[min]),col=2,lty=2)
```



As shown in the plots the two approchas give a similar $log(lambda)$ value.

2. A regression model with $p >> n$

Reading in the data.

```
express <- read.csv("journal.pbio.0020108.sd012.CSV",header=FALSE)
surv <- read.csv("journal.pbio.0020108.sd013.CSV",header=FALSE)
death <- (surv[,2]==1)
log.surv <- log(surv[death,1]+.05)
expr <- as.matrix(t(express[,death]))
colnames(expr) <- paste("V", 1:nrow(express), sep = "")
```

2.1 Lasso estimation using glmnet for regressing 'log.surv' against 'expr'

Glmnet and cv.glmnet are used to obtain the lasso regressino for $log(surv)$ against $express$.

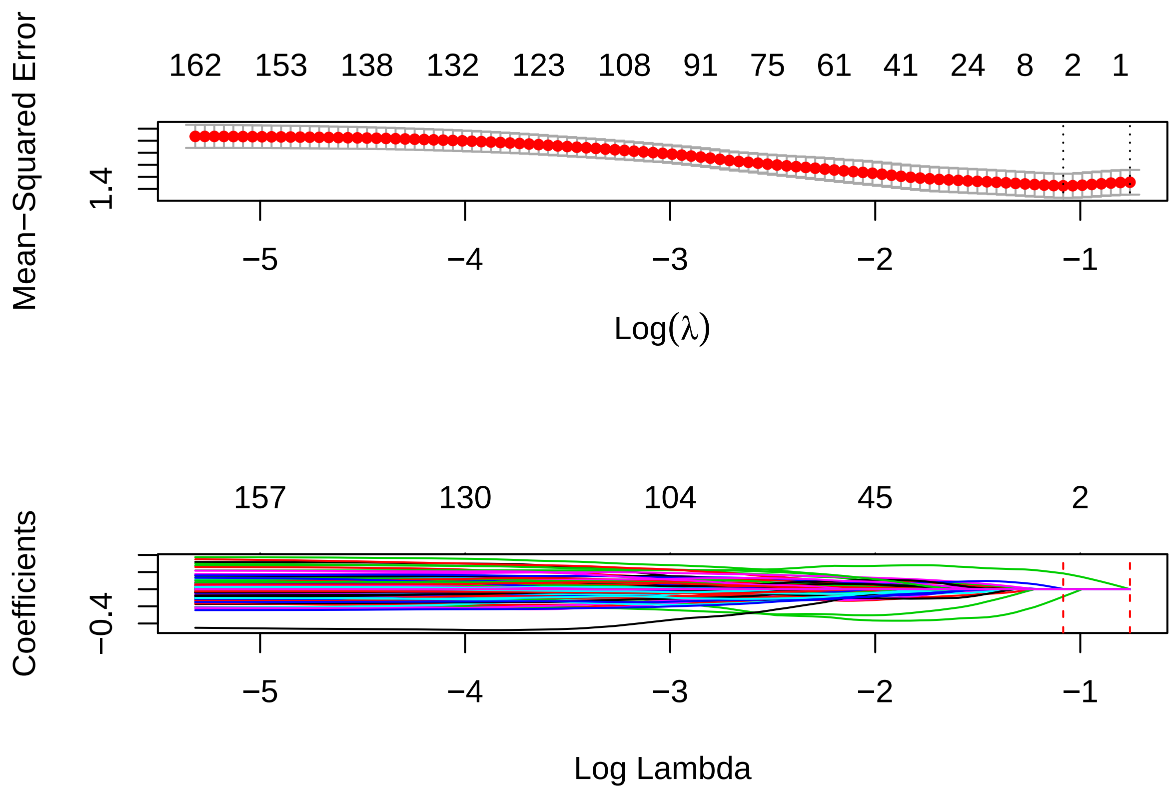
```
set.seed(1234)
lasso_surv <- glmnet(x = expr, y = log.surv,
                     alpha = 1)

cv_lasso_surv <- cv.glmnet(x = expr, y = log.surv,
                     alpha = 1)

# Number of non-zero coefficients
print("Number of non-zero coeff")
```

```
## [1] "Number of non-zero coeff"
```

```r
length(rownames(coef(cv_lasso_surv, s = "lambda.min"))[coef(cv_lasso_surv, s = "lambda.min")[,1] != 0])
```

```
## [1] 4
```

```r
# Plot two graphics
par(mfrow=c(2,1))
plot(cv_lasso_surv)
plot(lasso_surv,xvar="lambda")
abline(v=log(cv_lasso_surv$lambda.min),col=2,lty=2)
abline(v=log(cv_lasso_surv$lambda.1se),col=2,lty=2)
```




```r
par(mfrow=c(1,1))
```

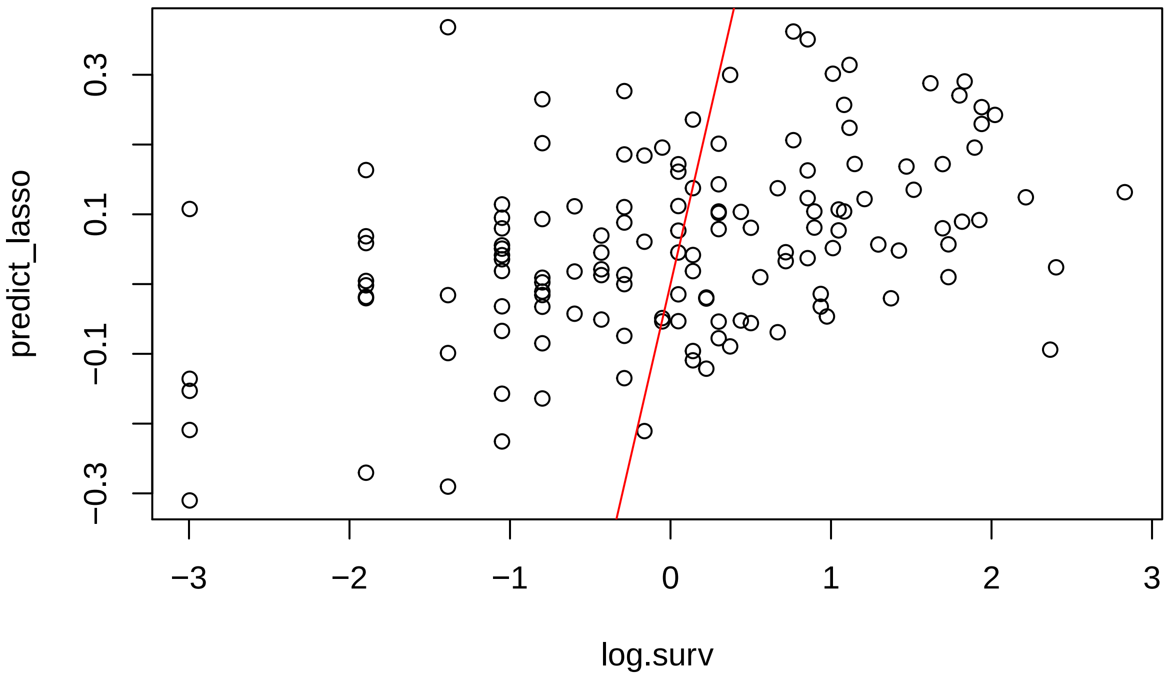
There are 3 non zero coefficiants using Lasso regreassion. As one can see from the MSE plot this corresponds well with what looks like the point with the lowest MSE. From the coefficient plot one can also see that min lambda results in a vertical line crossing 3 betapaths which corresponds to 3 beta values unequal to zero.

2.2 Computation of the responding fitted values

Task description:

Compute the fitted values with the Lasso estimated model (you can use predict). Plot the observed values for the response variable against the Lasso fitted values.

```r
predict_lasso <- predict(lasso_surv,
                         newx = expr,
                         s = cv_lasso_surv$lambda.min)
plot( log.surv, predict_lasso)
abline(a=0, b=1, col = 2)
```



As one can see from the plot the real values are on a larger scale than the predicted values. This could mean that this is not the best way to modle this type of data.

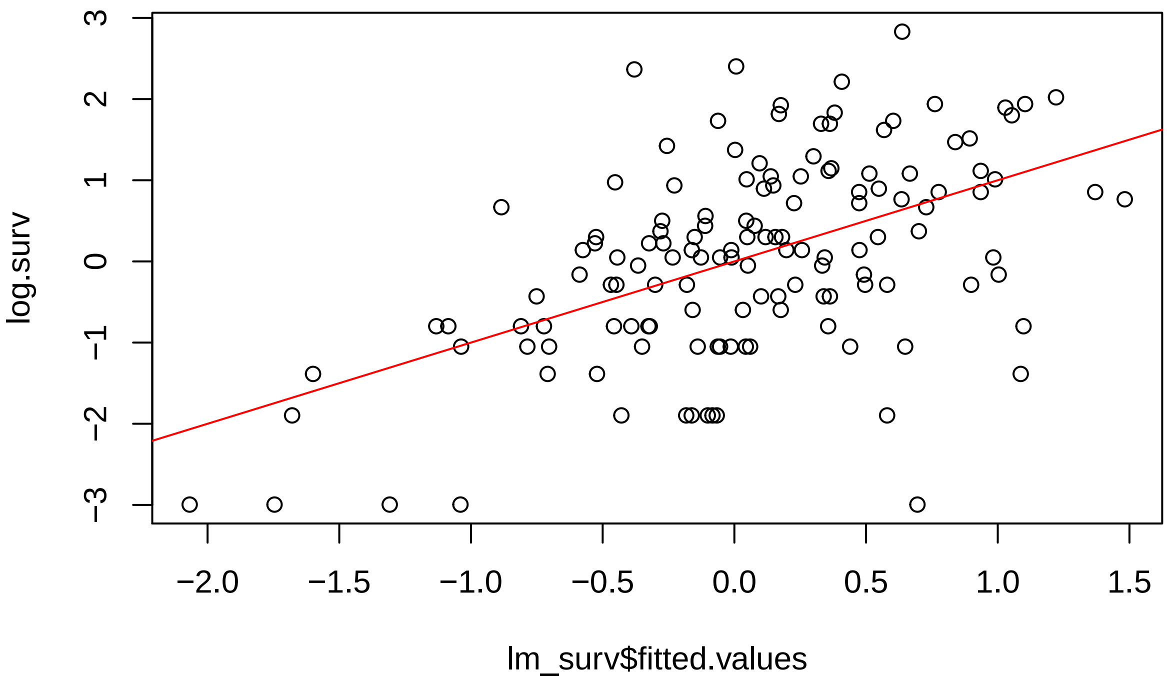
2.3 OLS regression model for 'log.surv' against 'expr'

Now we will fitt and OLS model with the responsvariables given by the non-zero coefficiants in the Lasso regreassion.

```
coeff_lasso <- rownames(coef(cv_lasso_surv, s = "lambda.min"))[coef(cv_lasso_surv, s = "lambda.min")[,1]

coeff_lasso
```

```
## [1] "(Intercept)" "V2252"       "V3787"       "V5352"
```

```
lm_surv <- lm(log.surv ~ expr[, coeff_lasso[-1]])


plot(lm_surv$fitted.values, log.surv)
abline(a = 0, b = 1, col = 2)
```



As one can clearly see this gives a much better prediction for our data. The scales are more similar and the data is quite evenly distributet around the line $x = y$.

2.4 Comparison of Lasso and OLS Regression

Task description:

Compare the OLS and Lasso fitted values. Do a plot for that.

```
print("Coefficiants Lasso regression")
coef(cv_lasso_surv, s = "lambda.min")[coef(cv_lasso_surv, s = "lambda.min")[,1] != 0]
print("Coefficiants OLS")
lm_surv$coefficients

plot(lm_surv$fitted.values, predict_lasso)
abline(a = 0, b = 1, col = 2)
```

The OLS coefficiants are a lot larger than the Lasso regression coefficiants, this can also be seen in the plot where one can observe that the OLS fittes values are on a much larger scale than the Lasso fitted values.