Assigment 1

Joram Keijser

Monday, November 17, 2014

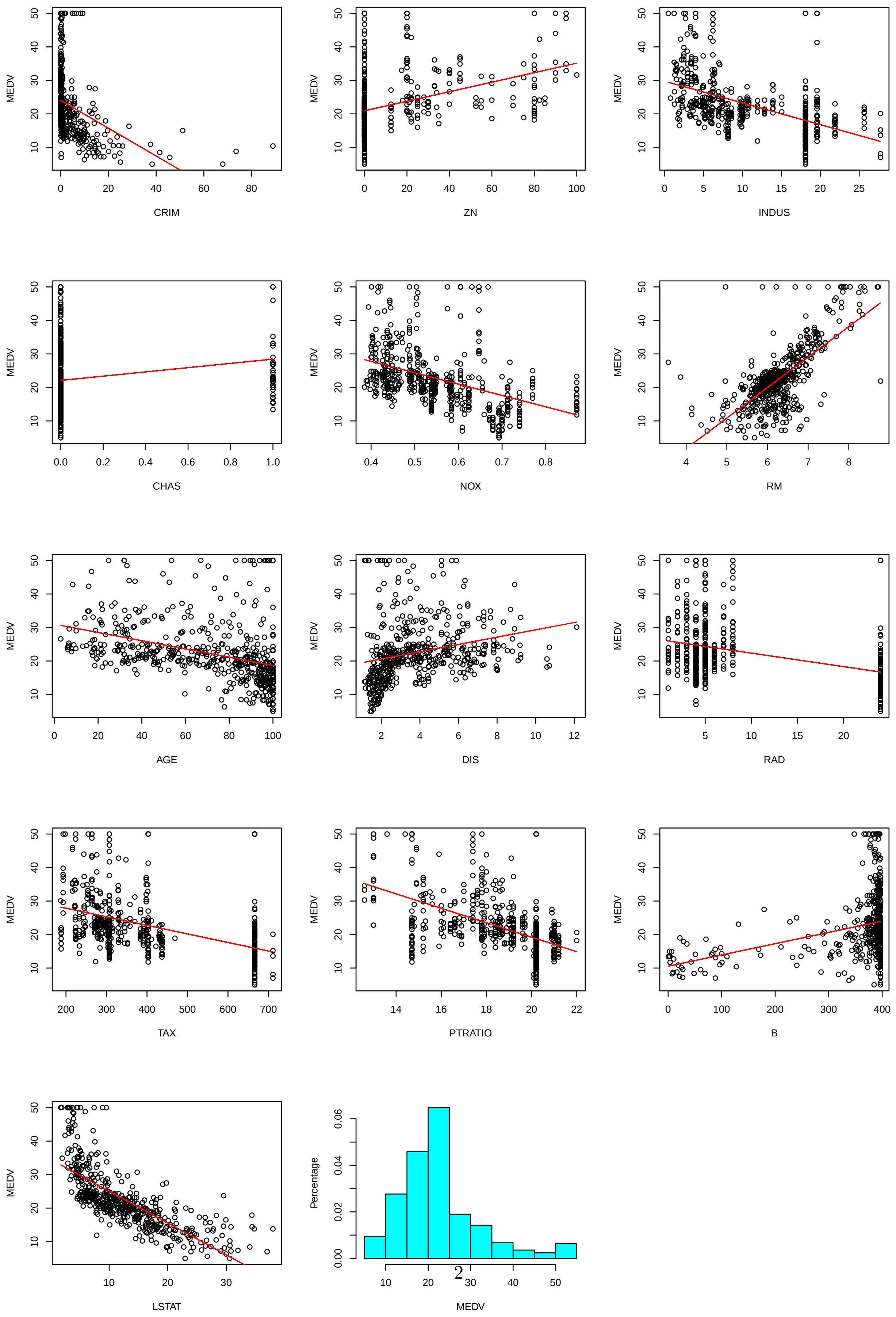
1, 2 Data

Get the data, divide into test and training set.

```
Data <- read.table("http://www.timvanerven.nl/wp-content/uploads/teaching/statlearn2014/housing-p.data"
Train <- Data[1:350, ]
Test <- Data[351:506, ]
FOLDS = 5
```

There are 13 explanatory variables, from which we try to predict 1 response variable: `MEDV`, the median value of owner-occupied homes in $1000's.We plot the each of the 13 explanatory variables against the number of `MEDV`, the median value of owner-occupied homes in $1000's. We added the single variable least squares fit to each plot. The independent variables are, on average, influencing `MEDV` in the way we expect them to. For example: higher criminality leads to a lower price, whereas a higher number of rooms leads to an higher price. Some plots show a nice 'continuous' relationship between prediction and response. See e.g. `LSTAT`, the percentage lower status of the population. Data on other variables is rather discrete of nature. Examples are `RAD`, an index of accessibility to radial highways and the binary `CHAS`. Finally, we notice some variables which do seem to correlate with `MEDV` in a particular way, but whose values are lumped together around a certain value. See for example `CRIM` and `B`. Also included is an histogram showing the distribution of `MEDV`.

```
par(mfrow = c(5, 3))
for(i in 1:13) {
  x <- Data[, i]
  y <- Data[, 14]
  plot(x, y, xlab = names(Data)[i], ylab = names(Data)[14])
  lm.model <- lm(y ~ x)
  lines(x, lm.model$coefficients[[1]] + lm.model$coefficients[[2]]*x, col = 'red')
}
truehist(Data$MEDV, xlab = 'MEDV', ylab = 'Percentage')
```


MEDV
CRIM
ZN
INDUS
CHAS
NOX
RM
AGE
DIS
RAD
TAX
PTRATIO
B
LSTAT
Percentage
MEDV

3, 4 Regression

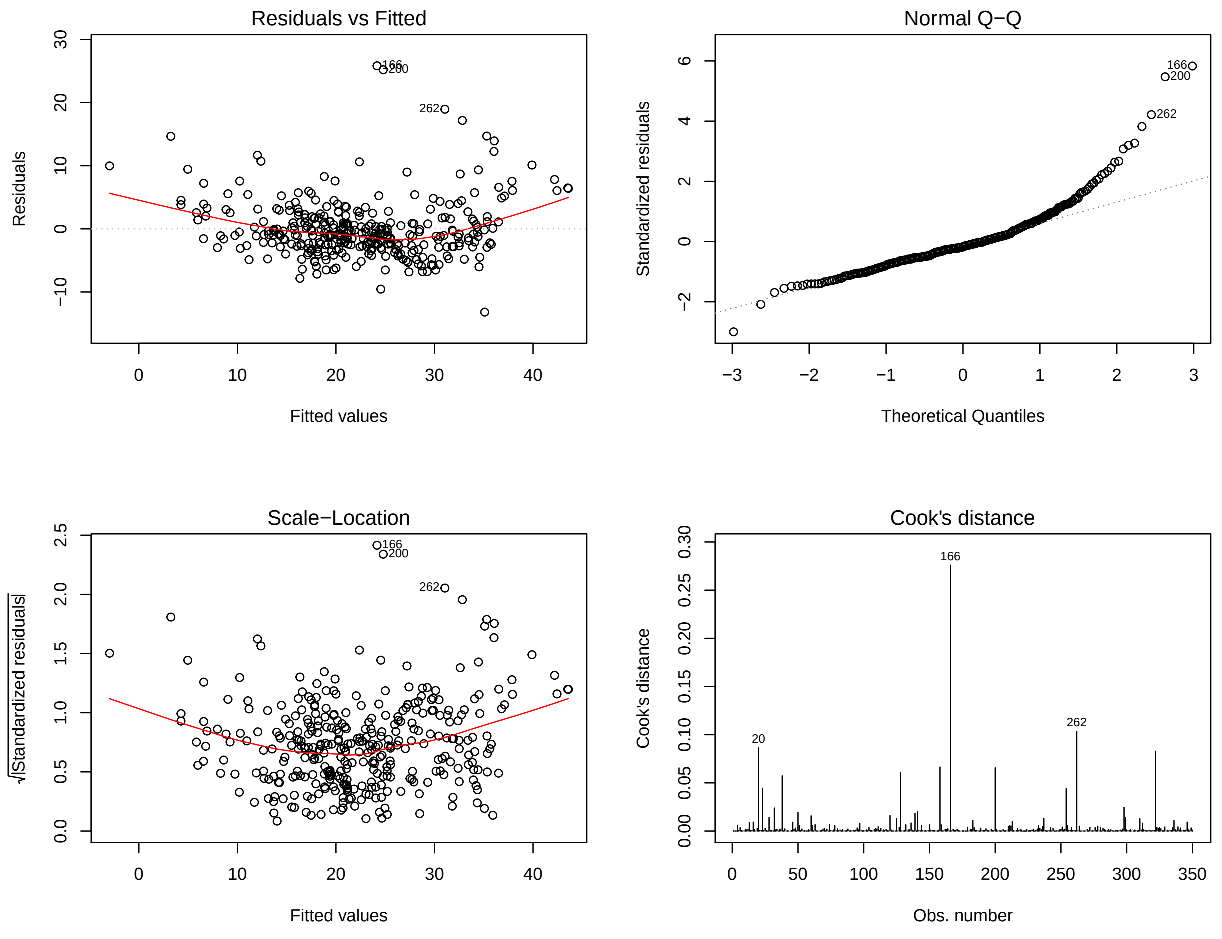
Of the 13 explanatory variables, 3 have not-statistically significant regression coefficients in our multivariate linear model. These are `INDUS` (proportion of non-retail business acres per town), `CHAS` (boolean indicating whether the tract bounds on the Charles River) and `AGE` (proportion of owner-occupied units built prior to 1940). The obtained coefficients are not very surprising. The independent variables are, on average, influencing `MEDV` in the way we expect them to. For example: higher criminality leads to a lower price, whereas a higher number of rooms leads to an higher price. In the single variable model based on all data, the variables `INDUS` and `DIS` are correlated negatively and positively, respectively. In the multivariate model build on the training data, these relationships are switched.

```r
lm.model <- lm(MEDV ~ ., data = Train)
summary(lm.model)
```

```
##
## Call:
## lm(formula = MEDV ~ ., data = Train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.191  -2.659  -0.788   1.732  25.831
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  39.40033    6.17442    6.38  5.8e-10 ***
## CRIM         -0.11352    0.04178   -2.72  0.00692 **
## ZN            0.06372    0.01638    3.89  0.00012 ***
## INDUS         0.02483    0.07374    0.34  0.73647
## CHAS          1.48609    0.99799    1.49  0.13740
## NOX         -17.03198    4.70072   -3.62  0.00034 ***
## RM            3.35805    0.52239    6.43  4.4e-10 ***
## AGE          -0.00537    0.01562   -0.34  0.73101
## DIS          -1.64248    0.23896   -6.87  3.1e-11 ***
## RAD           0.29926    0.07541    3.97  8.8e-05 ***
## TAX          -0.01378    0.00432   -3.19  0.00155 **
## PTRATIO      -0.84220    0.15372   -5.48  8.4e-08 ***
## B             0.00631    0.00318    1.99  0.04770 *
## LSTAT        -0.52157    0.06067   -8.60  3.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.67 on 336 degrees of freedom
## Multiple R-squared:  0.745,  Adjusted R-squared:  0.735
## F-statistic: 75.4 on 13 and 336 DF,  p-value: <2e-16
```

The plots Residuals vs fitted and Scale-location plot show that the residuals are to some extent, but not entirely randomly distributed when plotted against the fitted values. The Q-Q-plot shows that the residuals are neatly normaly distributed except in the very high (and very low) range. A plot of Cook's distance shows that there are 3 clear outliers, but no single observation influences the regression too much.

```r
#http://stats.stackexchange.com/questions/58141/interpreting-plot-lm
par(mfrow = c(2, 2))
for(i in 1:4) {
  plot.lm(lm.model, which = i)
}
```



5. Cross-validation

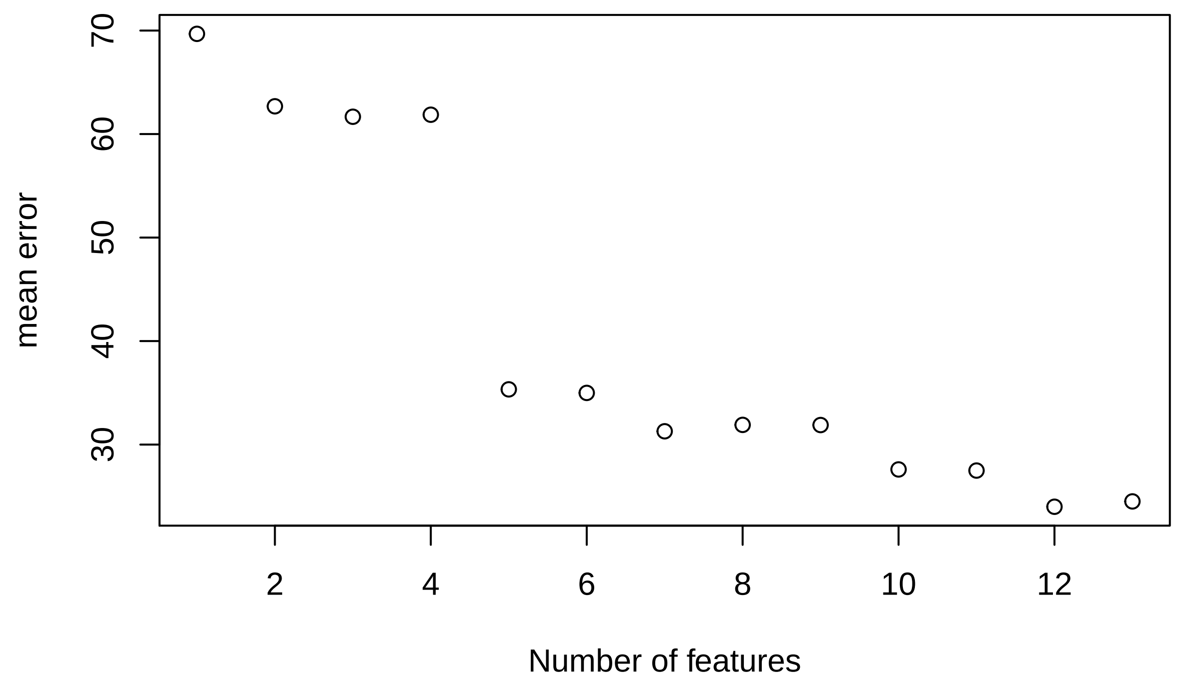
The training error equals 7341.966. This corresponds to an average squared difference between predicted and actual `MEDV` of 20.977. Since the variable is measured in $1000's, this corresponds to 20977 Dollars. We will see that the cross validation error is higher.

```
performCV <- function(Data, K = FOLDS, fitFUN = lm, ...){
  set.seed(123)
  # Divide data in K folds
  folds <- sample.int(K, size = dim(Data)[[1]], replace = TRUE)
  # Vector will contain mean error of every fold
  errors <- rep(0, 5)
  aantal <- rep(0, 5)
  for(k in 1:K) {
    # test on fold k, train on other folds:
    aantal[k] <- sum(folds == k)
    test = Data[folds == k, ]
    train = Data[folds != k, ]
    model <- fitFUN(MEDV ~ ., data = train)
    predict <- data.matrix(cbind(1, test[, -dim(test)[[2]]]))%*%data.matrix(model$coef)
    errors[k] <- sum((predict - test[, dim(Data)[[2]]])^2)
  }
  #return(mean(errors)) #averaged over all folds
  return(data.frame(errors = errors, samples = aantal))
}
result <- performCV(Train)
total <- sum(result[,1])
```

The mean cross validation error equals 8578.361. Averaged: 24.51, indeed higher than the training error. This does not come as a surprise, since the cross-validation error is supposed to be a more accurate estimate of the model's performance.

6. Full subset search

```
set.seed(123)
subModels <- regsubsets(as.matrix(Train[, -14]), Train[, 14], nbest = 1, nvmax = 20)
LMerrors <- rep(0, dim(summary(subModels)$which)[1])
for( k in 1:length(LMerrors)) {
  select <- as.logical(summary(subModels)$which[k,])
  select[14] <- TRUE
  subset <- Train[, select]
  result <- performCV(subset)
  LMerrors[k] <- sum(result[,1])/sum(result[,2])
}
plot(LMerrors, xlab = 'Number of features', ylab = 'mean error')
```



```
summary(subModels)$which[which(LMerrors == min(LMerrors)), ]
```

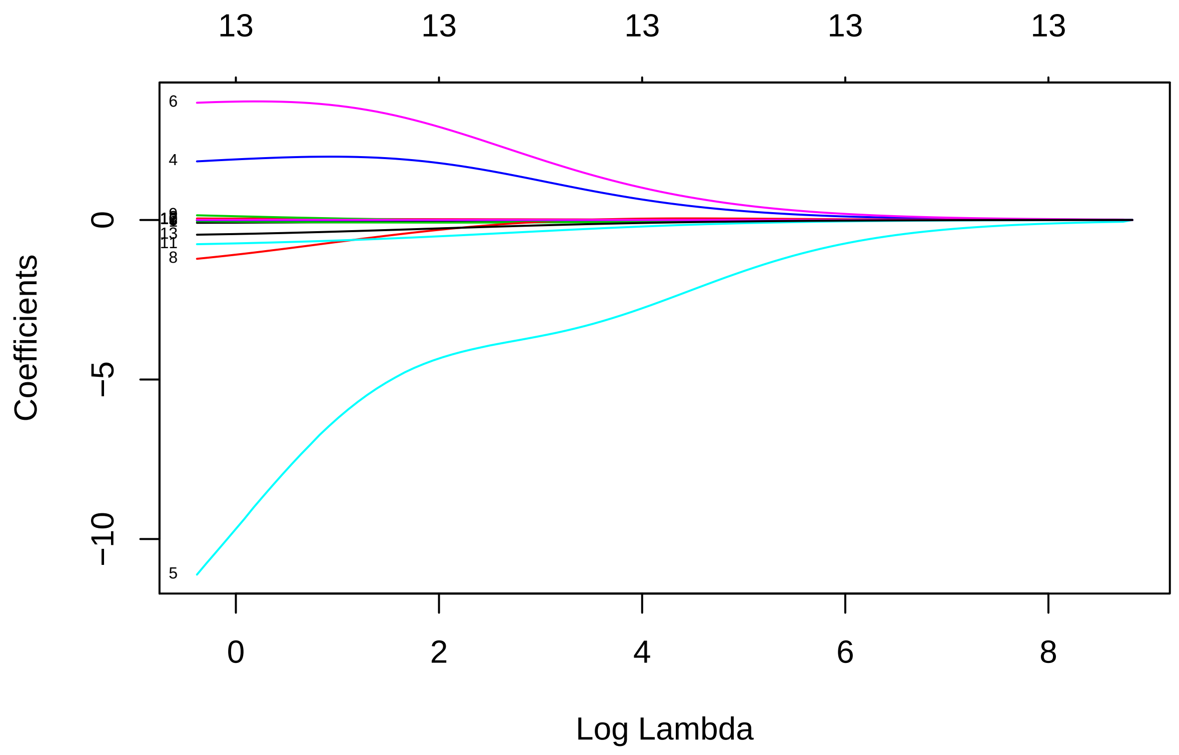
```
## (Intercept)        CRIM          ZN       INDUS        CHAS         NOX 
##        TRUE        TRUE        TRUE       FALSE        TRUE        TRUE 
##          RM         AGE         DIS         RAD         TAX     PTRATIO 
##        TRUE        TRUE        TRUE        TRUE        TRUE        TRUE 
##           B       LSTAT 
##        TRUE        TRUE
```

Minimum of mean error: 23.995, achieved with all 12 features `INDUS`. This error is slightly higher than that achieved when using all 13 features (see figure). This is not a very big surprise, since coefficient of the excluded feature has a very high p-value in the multivariate linear model.

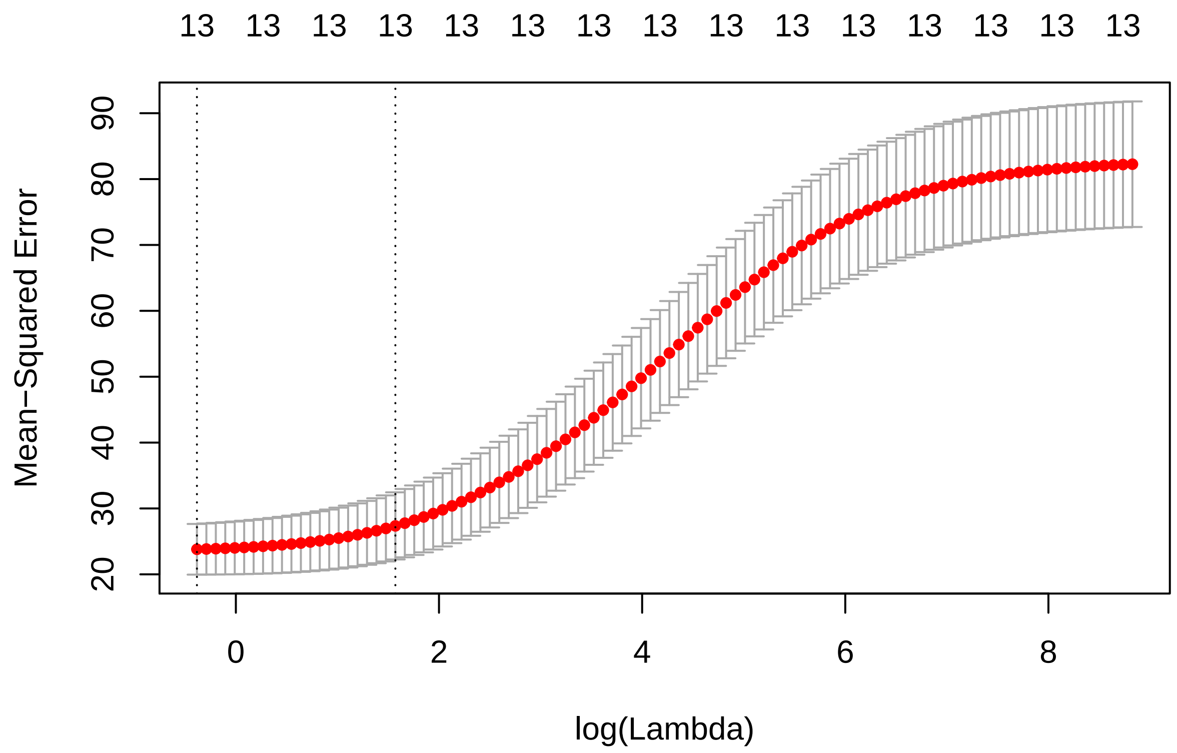
7. Ridge

We use the package `glmnet` which provides higher level functions for performing ridge and lasso regression. The first plot shows that for small values of (log) lambda, the coefficients of variables `CHAS`, `NOX` and `RM` (4, 5, and 6, respectively) are very large. Just like we saw in the least squares fit, of course.

```
set.seed(123)
x <- as.matrix(Train[, c(1:13)])
y <- as.matrix(Train[, 14])
fit.ridge <- glmnet(x, y, alpha = 0)
plot(fit.ridge, xvar="lambda", label = TRUE)
```



```
cv.ridge = cv.glmnet(x, y, alpha = 0, nfold = FOLDS)
plot(cv.ridge)
```



```r
cv.ridge = cv.glmnet(x, y, alpha = 0, nfold = FOLDS, lambda = seq(0.001, 2, by = 0.001))
predict <- predict(cv.ridge, newx = x, s = "lambda.min")
minIndex <- which(cv.ridge$lambda == cv.ridge$lambda.min)
Ridge = as.matrix(coef(cv.ridge, s = "lambda.min"))
LM = summary(lm.model)$coef[,1]
```

From the second plot, we notice that the cross-validation error is smallest for values of lambda smaller than 1. By subsequentely letting lambda range between 0 and 1, we find that the minimal mean-squared error equals 23.602. The corresponding lambda equals 0.162. This cv-error is less than that of 0.369, achieved by ordinary least squares. The coefficients are indeed smaller than in the LM model. The difference is only -0.071, but the different scales of the prediction variables make this number not directly interpretable.

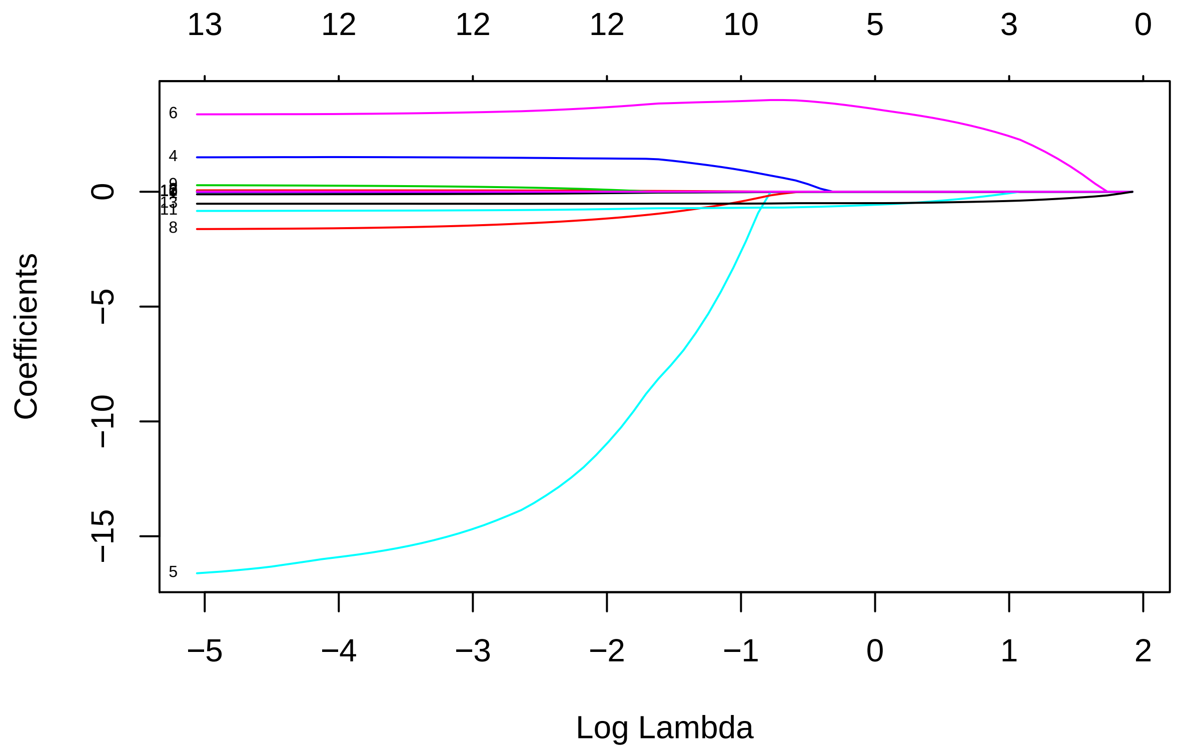
```r
data.frame(Ridge = Ridge, LM = LM)
```

```
##                      X1        LM
## (Intercept)  36.03344  39.40033
## CRIM         -0.10405  -0.11352
## ZN            0.05756   0.06372
## INDUS        -0.00258   0.02483
## CHAS          1.62462   1.48609
## NOX         -15.02044 -17.03198
## RM            3.48789   3.35805
## AGE          -0.00612  -0.00537
## DIS          -1.51390  -1.64248
## RAD           0.23958   0.29926
## TAX          -0.01095  -0.01378
## PTRATIO      -0.81602  -0.84220
## B             0.00641   0.00631
## LSTAT        -0.50413  -0.52157
```

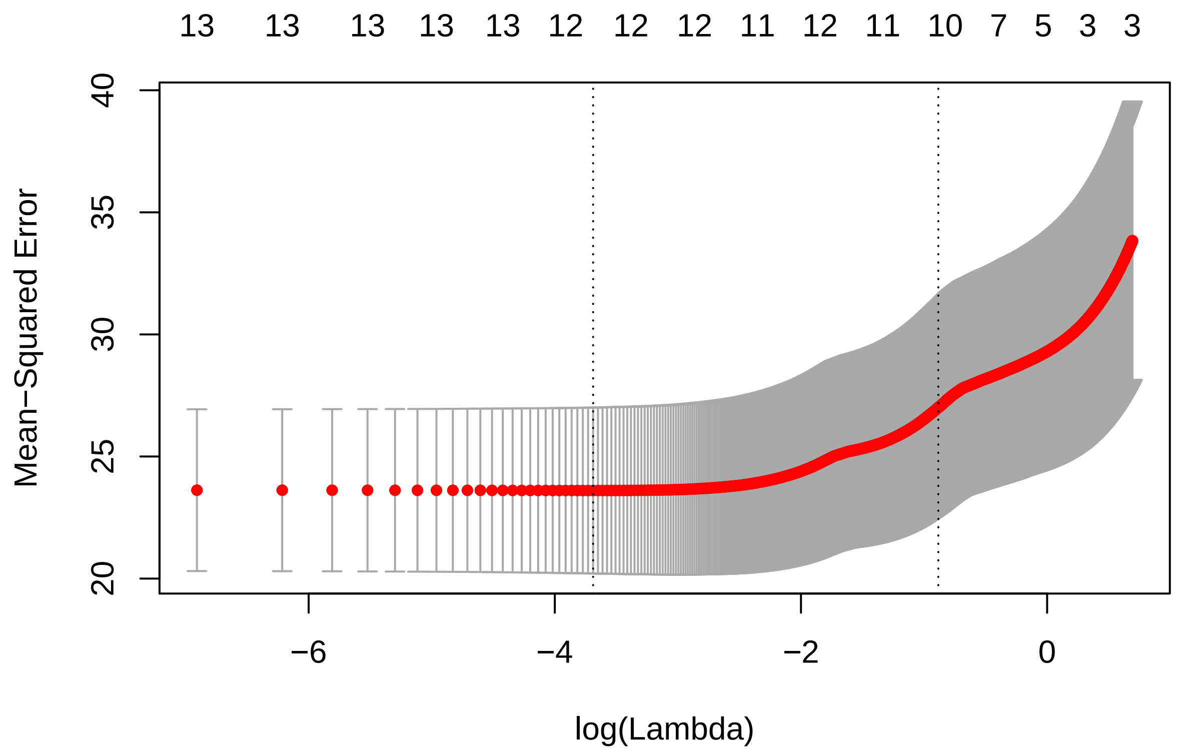
8 Lasso

Also using `glmnet` package.

```r
set.seed(123)
fit.lasso<- glmnet(x, y, alpha = 1)
plot(fit.lasso, xvar="lambda", label = TRUE)
```



```r
cv.lasso = cv.glmnet(x, y, alpha = 1, nfold = FOLDS, lambda = seq(0.001, 2, by = 0.001))
plot(cv.lasso)
```



```r
predict <- predict(cv.lasso, newx = x, s = "lambda.min")
minIndex <- which(cv.lasso$lambda == cv.lasso$lambda.min)
Lasso = as.matrix(coef(cv.lasso, s = "lambda.min"))
```

Best: 23.606. This is better than both LS and Ridge. The corresponding lambda equals 0.025. Since this optimal lambda is small, only one variable has coefficient 0. This is `INDUS`, exactly the variable kicked out by model selection.

```r
coefs <- data.frame(Lasso, Ridge, LM)
colnames(coefs) <- c("Lasso", "Ridge", "LM")
coefs
```

```
##                 Lasso     Ridge        LM
## (Intercept)  37.44529  36.03344  39.40033
## CRIM         -0.10427  -0.10405  -0.11352
## ZN            0.05961   0.05756   0.06372
## INDUS         0.00000  -0.00258   0.02483
## CHAS          1.50770   1.62462   1.48609
## NOX         -15.67319 -15.02044 -17.03198
## RM            3.40370   3.48789   3.35805
## AGE          -0.00364  -0.00612  -0.00537
## DIS          -1.56345  -1.51390  -1.64248
## RAD           0.25312   0.23958   0.29926
## TAX          -0.01160  -0.01095  -0.01378
## PTRATIO      -0.82062  -0.81602  -0.84220
## B              0.00609   0.00641   0.00631
## LSTAT         -0.52102  -0.50413  -0.52157
```

9 All three methods

In the end, the difference in performance of the methods is immaterial.

```
set.seed(123)
loss <- function(predict){
  mean((predict- Test[,14])^2)

}
lm.model <- lm(MEDV ~., data = Train[, -which(names(Test) == "INDUS")])
predict.lm <- data.matrix(cbind(1, Test[, -c(which(names(Test) == "INDUS"),14)]))%*%
  data.matrix(lm.model$coef)
loss(predict.lm)
```

```
## [1] 25.2
```

```
x_test = as.matrix(Test[, c(1:13)])
predict.Ridge <- predict(cv.ridge, newx = x_test, s="lambda.min")
loss(as.matrix(predict.Ridge))
```

```
## [1] 25.1
```

```
predict.Lasso <- predict(cv.lasso, newx = x_test, s="lambda.min")
loss(as.matrix(predict.Lasso))
```

```
## [1] 25.2
```