Lasso Regression Assignment

Ignasi Mañé, Antoni Company & Chiara Barbi

24 de febrero de 2019

1. Introduction

The Boston House-price data-set concerns housing values in 506 suburbs of Boston corresponding to year 1978. The list of variables are:

1. CRIM: per capita crime rate by town.
2. ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS: proportion of non-retail business acres per town.
4. CHAS: Charles river dummy variable (= 1 if tract bounds river; 0 otherwise).
5. NOX: nitric oxides concentration (parts per 10 million).
6. RM: average number of rooms per dwelling.
7. AGE: proportion of owner-occupied units built prior to 1940.
8. DIS: weighted distances to five Boston employment centers.
9. RAD: index of accessibility to radial highways.
10. TAX: full-value property-tax rate per $10,000.
11. PTRATIO: pupil-teacher ratio by town.
12. B: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.
13. LSTAT: % lower status of the population.
14. MEDV: Median value of owner-occupied homes in $1000's.

The Boston House-price corrected data-set (available in boston.Rdata) contains the same data (with some corrections) and it also includes the UTM coordinates of the geographical centers of each neighborhood.

The following Chunk imports the Boston data and the libraries required to fulfill the assignment.

```r
library(glmnet)
library(caret)
set.seed(3456)
load("boston.Rdata")
boston <- boston.c
remove(boston.c, boston.soi, boston.utm)
#Import de boston corrected dataset to the boston data frame
```

2. Assignment questions

2.1. For the Boston House-price corrected dataset use Lasso estimation (in glmnet) to fit the regression model where the response is CMEDV (the corrected version of MEDV) and the explanatory variables are the remaining 13 variables in the previous list. Try to provide an interpretation to the estimated model.

We will split the data with the required variables into a matrix of covariates and a response vector and then it will be scaled and centered:

```r
#Non scaled data
Y <- boston$CMEDV
X <- apply(as.matrix(boston[,8:20]),2,as.numeric)
```

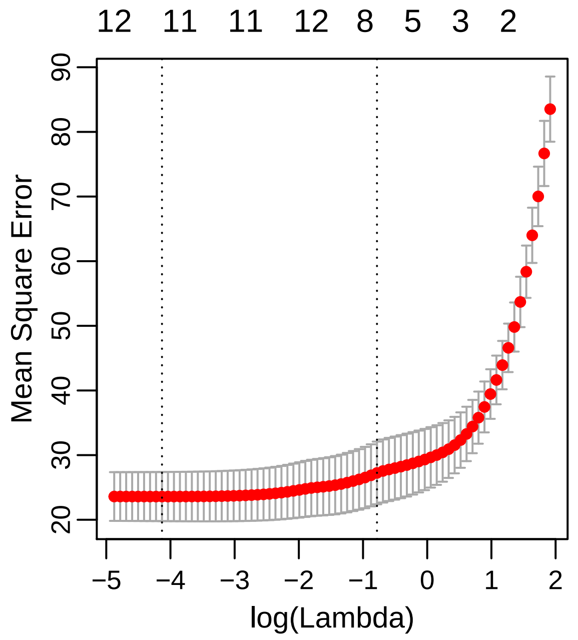
```r
#Scaled data
Y.scale <- scale(Y,center=TRUE,scale = FALSE)
X.scale <- scale(X,center=TRUE,scale = TRUE)
```

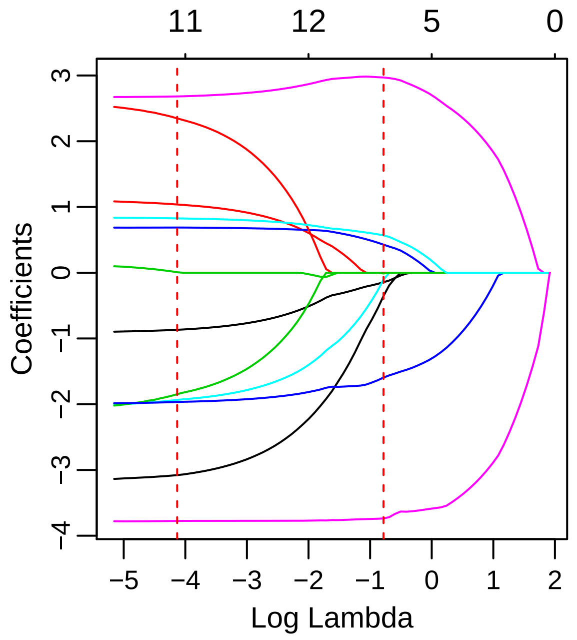
We also scaled the dummy variable CHAS according to Tibshirani (THE LASSO METHOD FOR VARIABLE SELECTION IN THE COX MODEL, Statistics in Medicine, VOL. 16, 385-395 (1997)) which on page 394 says "The lasso method requires initial standardization of the regressors, so that the penalization scheme is fair to all regressors. For categorical regressors, one codes the regressor with dummy variables and then standardizes the dummy variables. As pointed out by a referee, however, the relative scaling between continuous and categorical variables in this scheme can be somewhat arbitrary."

Once the data is centered and scaled we can proceed with the Lasso regression using the glmnet function, but, first, we will apply the 10-fold cross-validation method to define the optimum lambda for the Lasso Regression.

Since we already centered the data, we will set the parameter standardize=FALSE, and for simplicity will not consider an intercept parameter.

```r
#10-fold cross validation test in Lasso Regression
k <- 10
CV.CMEDV.lasso <- cv.glmnet(x = X.scale, y = Y.scale,
                            standardize=FALSE, intercept=FALSE, nfolds=k)
lambda.opt.lasso <- CV.CMEDV.lasso$lambda.min

#Lasso regression of the variable CMEDV
CMEDV.lasso <- glmnet(X.scale,Y.scale, standardize = FALSE , intercept = FALSE)

#plot results Lasso regression
par(mfrow=c(1,2), mgp=c(1.5,0.5,0), oma = c(3,0,1,0), mar = c(4,3,2,1))
plot(CV.CMEDV.lasso, ylab="Mean Square Error",
     cex.axis=0.8, cex.lab=0.9)
plot(CMEDV.lasso, xvar="lambda",
     cex.axis=0.8, cex.lab=0.9)
abline(v=log(CV.CMEDV.lasso$lambda.min),col=2,lty=2)
abline(v=log(CV.CMEDV.lasso$lambda.1se),col=2,lty=2)
```





In the left hand side of the picture, the chart shows a clear trend: the lesser the value of lambda is, the lesser the value of the the expected Mse. The minimum is reached for $\lambda^* = 0.016$ (vertical line). The second vertical line corresponds to the value of the "lambda one standard error" (λ^{**}).

In the right hand side of the picture there is a chart representing the value of the beta coefficients for each of the possible values of lambda considered by the default parameters of the glmnet function (100 values). In the optimum ($\lambda*$), only the variables $INDUS$ and AGE are associated to zero β coefficients, whereas for λ^{**} variables RAD, TAX and ZN also have zero β coefficients, thus reducing the degrees of freedom of the model from 11 to 8.

```
#coefficient of the optimal lambda
coef.lambda.min <- coef(CMEDV.lasso, s = CV.CMEDV.lasso$lambda.min)
#coefficient of the optimal lambda 1se
coef.lambda.1se <- coef(CMEDV.lasso, s = CV.CMEDV.lasso$lambda.1se)
#Total
Fin.lamb <- as.matrix(cbind(coef.lambda.min,coef.lambda.1se))
colnames(Fin.lamb) <- c("beta.lambda.min","beta.lambda.1se")
```

```
#Print Results
#optimal lambda for lasso regression with 10-fold cros-validation test
print("Optimal lambda for the Lasso Regression")
```

```
## [1] "Optimal lambda for the Lasso Regression"
```

```
lambda.opt.lasso
```

```
## [1] 0.01605262
```

```
print("Optimal lambda coefficients for the Lasso Regression")
```

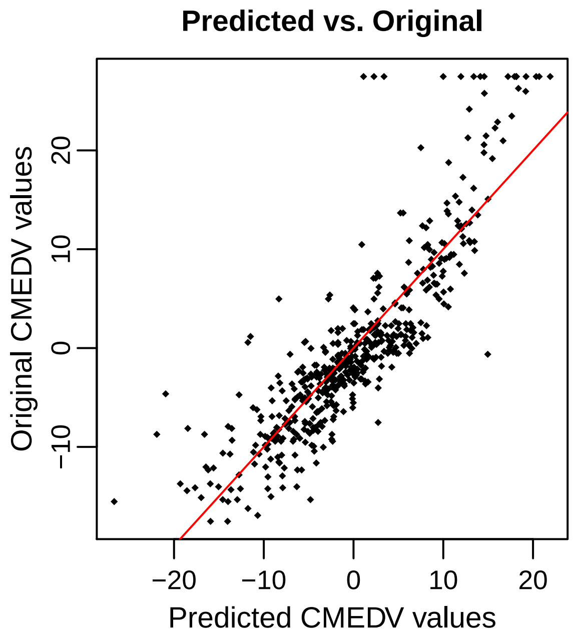
```
## [1] "Optimal lambda coefficients for the Lasso Regression"
```

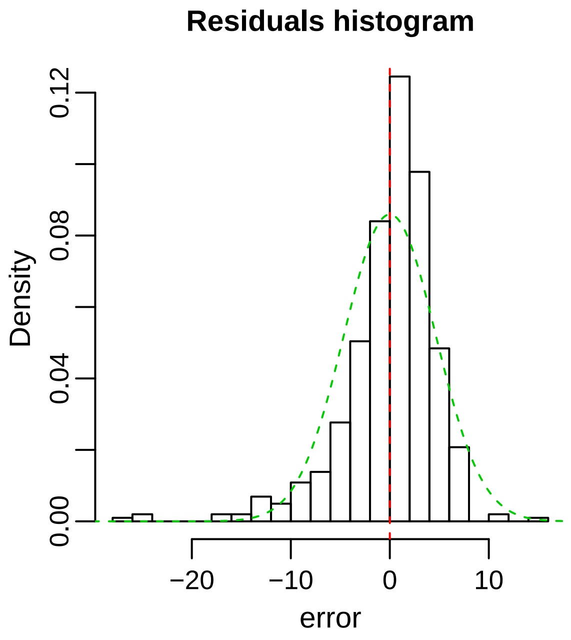
```
round(Fin.lamb,3)
```

```
##             beta.lambda.min beta.lambda.1se
## (Intercept)           0.000           0.000
## CRIM                 -0.868          -0.144
## ZN                      1.038           0.000
## INDUS                   0.008           0.000
## CHAS                    0.687           0.427
## NOX                    -1.934          -0.104
## RM                      2.682           2.970
## AGE                     0.000           0.000
## DIS                    -3.078          -0.354
## RAD                     2.347           0.000
## TAX                    -1.847          -0.012
## PTRATIO                -1.966          -1.595
## B                       0.827           0.568
## LSTAT                  -3.775          -3.737
```

Now, we will generate the predicted values using the lasso regression model for λ^* and compare them with the original values to visualize how accurate the prediction is.

```
CMEDV.predic <- predict(CMEDV.lasso, s = lambda.opt.lasso, newx = X.scale)

par(mfrow=c(1,2), mgp=c(1.5,0.5,0), oma = c(3,0,1,0), mar = c(4,3,2,1))

plot(x=CMEDV.predic, y=Y.scale, pch=18, cex=.5,
     cex.lab=0.9, cex.axis=0.8, cex.main=0.9,
     xlab = "Predicted CMEDV values",
     ylab =  "Original CMEDV values",
     main = "Predicted vs. Original" )
abline(a=0,b=1,col=2,lty=1)

error <- CMEDV.predic - Y.scale
hist(x = error, breaks = 16,freq = FALSE,
     main = "Residuals histogram",
     cex.lab=0.9, cex.axis=0.8, cex.main=0.9)
x.norm <- seq(-30,30,0.1)
y.norm <- dnorm(x = x.norm , mean = mean(error), sd = sd(error))
lines(x = x.norm,y=y.norm,col=3,lty=2)
abline(v=mean(error),col=2,lty=2)
```





It is clear that the correlation between the values predicted by the model and the real ones is very high. In addition, it “seems”" that the residuals follow a normal distribution.

2.2. Use glmnet to fit the previous model using ridge regression. Compare the 10-fold cross validation results from function cv.glmnet with those you obtained in the previous practice with your own functions.

2.2.1. Ridge Regression Glmnet

First, we will start doing 10-fold cross validation using the cv.glmnet function provided by R in the glmnet package. To apply the ridge regression, we must define $\alpha = 0$ in the cv.glmnet function and also a sequence of λ_i.

However, since we want to compare the glmnet algorithm with the one that we developed for the last assignment, we have to take into account that the scale of the lambda used in the glm package differs from that we used in theory because the definition of the optimization problem is slightly different.

According to the documentation of the version glmnet_2.0-13 the Ridge problem is defined as follows:

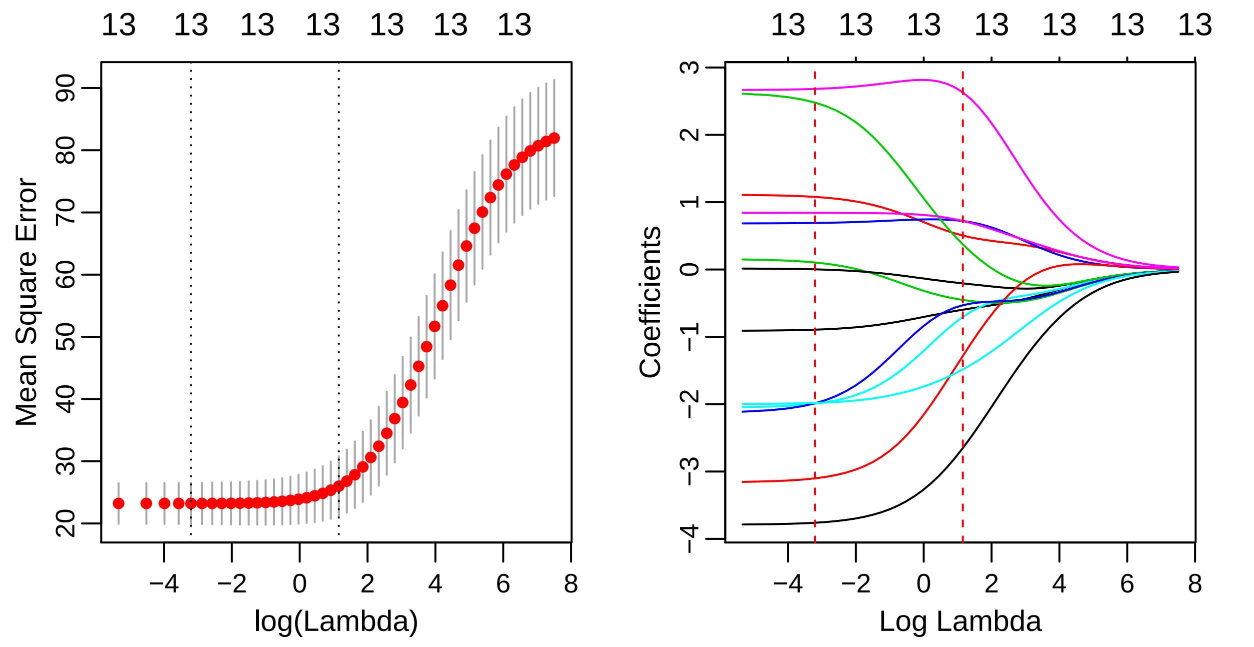
$$Min: \frac{1}{2N} \left\| \frac{y}{S_y} - X\beta \right\|_2^2 + \frac{\lambda}{2S_y} \|\beta\|_2^2$$

If we solve this problem, then we obtain the following expression:

$$\beta = (X^T X - \frac{N\lambda}{S_y} I_p) \frac{X^T y}{S_y}$$

Glmnet returns βS_y as the the vector of coefficients. However, notice that the value of lambda is scaled by a factor $\frac{N}{S_y}$. Taking into account this factor of scaling we now have to transform the vector of lambdas to penalize the model equivalently. Otherwise both algorithms would be incomparable because we would have used different values of lambdas to train the model.

```r
lambda.max <- 1e5
n.lambdas <- 50
lambda.v <- exp(seq(0,log(lambda.max+1),length=n.lambdas))-1

n <- dim(Y.scale)[1]
sd_y <- sqrt(var(Y.scale)*(n-1)/n)[1,1]
t.lambda <- lambda.v*sd_y/n

CV.CMEDV.ridge <- cv.glmnet(x = X.scale, y = Y.scale, alpha = 0,
                            lambda = t.lambda, nfolds= 10,standardize = FALSE,
                            intercept = FALSE,thresh = 1e-20, type.measure = "mse")

lambda.opt.ridge <- CV.CMEDV.ridge$lambda.min
lambda.opt.ridge.1se <- CV.CMEDV.ridge$lambda.1se

#Lasso regression of the variable CMEDV
CMEDV.ridge <- glmnet(X.scale,Y.scale, standardize = FALSE ,
                      intercept = FALSE, alpha = 0,lambda = t.lambda)

#plot results Lasso regression
par(mfrow=c(1,2), mgp=c(1.5,0.5,0), oma = c(3,0,1,0), mar = c(4,3,2,1))
plot(CV.CMEDV.ridge, ylab="Mean Square Error",
     cex.axis=0.8, cex.lab=0.9)
plot(CMEDV.ridge, xvar="lambda",
     cex.axis=0.8, cex.lab=0.9)
abline(v=log(CV.CMEDV.ridge$lambda.min),col=2,lty=2)
abline(v=log(CV.CMEDV.ridge$lambda.1se),col=2,lty=2)
```

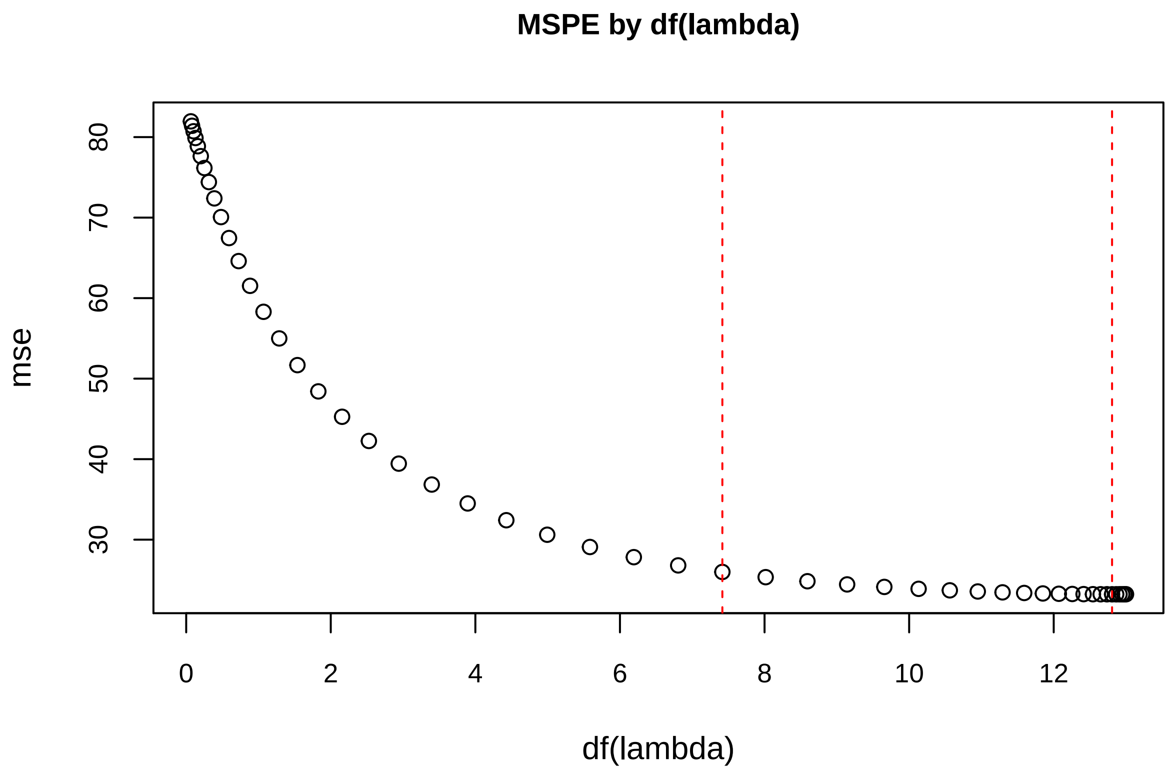


The figure is analogous to that presented in exercise 1. In the left hand side of the picture, the chart again shows same trend: the lesser the value of lambda is, the lesser the value of the the expected Mse. The minimum is reached for $\lambda^* = 0.04$. The vertical line corresponds to the value of the "lambda one standard error" (λ^{**}).

In the right hand side of the picture there chart represents the value of the beta coefficients for each of the possible 50 values of lambda considered. Notice that for big values of lambda, the value of the beta coefficients tend to 0 as expected. As in the previous chart, the vertical line corresponds to the value of the "lambda one standard error" (λ^{**}).

For ridge regression there is a simple close expression to compute the value of the effective number of degrees of freedom. So it makes sense to express the x axes in terms of this quantity for the left hand side chart. .

```r
df.lambda <- cal.edf(x = X.scale, lambda = lambda.v)
df.lambda.min <- cal.edf(x = X.scale, lambda = lambda.opt.ridge*n/sd_y)
df.lambda.1se <- cal.edf(x = X.scale, lambda = lambda.opt.ridge.1se*n/sd_y)
mse <- CV.CMEDV.ridge$cvm

plot(df.lambda,rev(mse),
     xlab="df(lambda)",
     ylab = "mse",
     main="MSPE by df(lambda)",
     cex.main=0.9, cex.axis=0.8)
abline(v=df.lambda.min,col=2,lty=2)
abline(v=df.lambda.1se,col=2,lty=2)
```



The lambdas (in the glmnet scale) and the value of the coefficients of beta for the ridge regression using the glmnet function that we obtain are:

```r
#coefficient of the optimal lambda
coef.lambda.min <- coef(CMEDV.ridge, s = CV.CMEDV.ridge$lambda.min)
#coefficient of the optimal lambda 1se
coef.lambda.1se <- coef(CMEDV.ridge, s = CV.CMEDV.ridge$lambda.1se)
#Total
Fin.bet.lamb <- as.matrix(cbind(coef.lambda.min,coef.lambda.1se))
colnames(Fin.bet.lamb) <- c("lambda.min","lambda.1se")
```

```r
#Print Results
print("Optimal lambdas glmnet scale")
```

```
## [1] "Optimal lambdas glmnet scale"
```

```r
c(CV.CMEDV.ridge$lambda.min, CV.CMEDV.ridge$lambda.1se)
```

```
## [1] 0.04056215 3.16801557
```

```r
print("Optimal Beta coefficients for the Ridge Regression")
```

```
## [1] "Optimal Beta coefficients for the Ridge Regression"
```

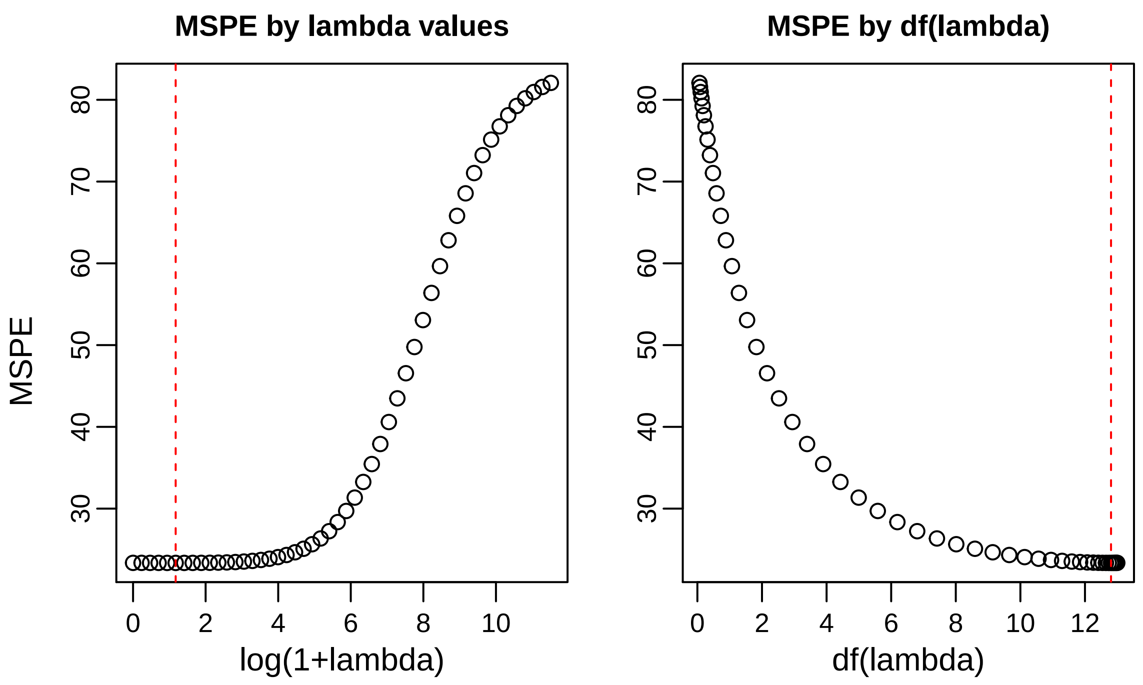
```r
print(round(Fin.bet.lamb,3))
```

```
##             lambda.min lambda.1se
## (Intercept)      0.000      0.000
## CRIM            -0.893     -0.597
## ZN               1.078      0.505
## INDUS            0.105     -0.452
## CHAS             0.691      0.720
## NOX             -1.991     -0.704
## RM               2.682      2.630
## AGE              0.003     -0.205
## DIS             -3.101     -1.284
## RAD              2.479      0.374
## TAX             -1.987     -0.530
## PTRATIO         -1.981     -1.484
## B                0.841      0.721
## LSTAT           -3.761     -2.652
```

2.2.1. Manual Ridge Regression

Now we repeat the same process with the functions developed in the last assignment to study whether there exist any differences.

```r
CV.CMEDV.ridge.function <- ridgereg.cv(x=X.scale,y=Y.scale,k=10,lambda=lambda.v,plot=TRUE)
```



```
lambda.opt.ridge.func <- CV.CMEDV.ridge.function[which.min(CV.CMEDV.ridge.function[,2]),1]
df.opt.ridge.func <- CV.CMEDV.ridge.function[which.min(CV.CMEDV.ridge.function[,2]),3]
CMEDV.ridge.function.coef <- fit.ridge(x=X.scale,y=Y.scale,xval=X.scale,yval=Y.scale,lambda=lambda.opt.r


print("Optimal lambda")
```

```
## [1] "Optimal lambda"
```

```
lambda.opt.ridge.func
```

```
##   lambda
## 2.237461
```

```
print("Coefficients of the optimal lambda")
```

```
## [1] "Coefficients of the optimal lambda"
```

```
CMEDV.ridge.function.coef$beta
```

```
##                   [,1]
## CRIM    -0.895599267
## ZN       1.078980200
## INDUS    0.108124227
## CHAS     0.691054263
## NOX     -1.995144347
## RM       2.681405843
## AGE      0.002520772
## DIS     -3.101994075
## RAD      2.490718852
## TAX     -1.996108720
## PTRATIO -1.982875244
## B        0.841722961
## LSTAT   -3.760074073
```

2.2.1. Compairason

To compare both algorithms we have to use the same scale for lambda. Otherwise the comparison would be useless. To transform the lambdas form the glmnet scale to the scale that we used in the manual algorithm we have to multiply the values by a factor of $\frac{N}{S_y}$, just as we did to compute the equivalent degrees of freedom for the values of lambda computed by the glmnet algorithm. Taking into account only the optimal lambda, the final results are:

Optimal Lambdas

```
lambdas.fin <- c(CV.CMEDV.ridge$lambda.min*n/sd_y,lambda.opt.ridge.func)
names(lambdas.fin) <- c("Glmnet lambda","Theory lambda")

print(lambdas.fin)
```

```
## Glmnet lambda Theory lambda
##      2.237461      2.237461
```

Optimal beta coeficients

```
Final.betas <- cbind(matrix(Fin.bet.lamb[-1,1], ncol = 1),CMEDV.ridge.function.coef$beta)
colnames(Final.betas) <- c("beta Glmnet lambda","beta Theory lambda")

print(Final.betas)
```

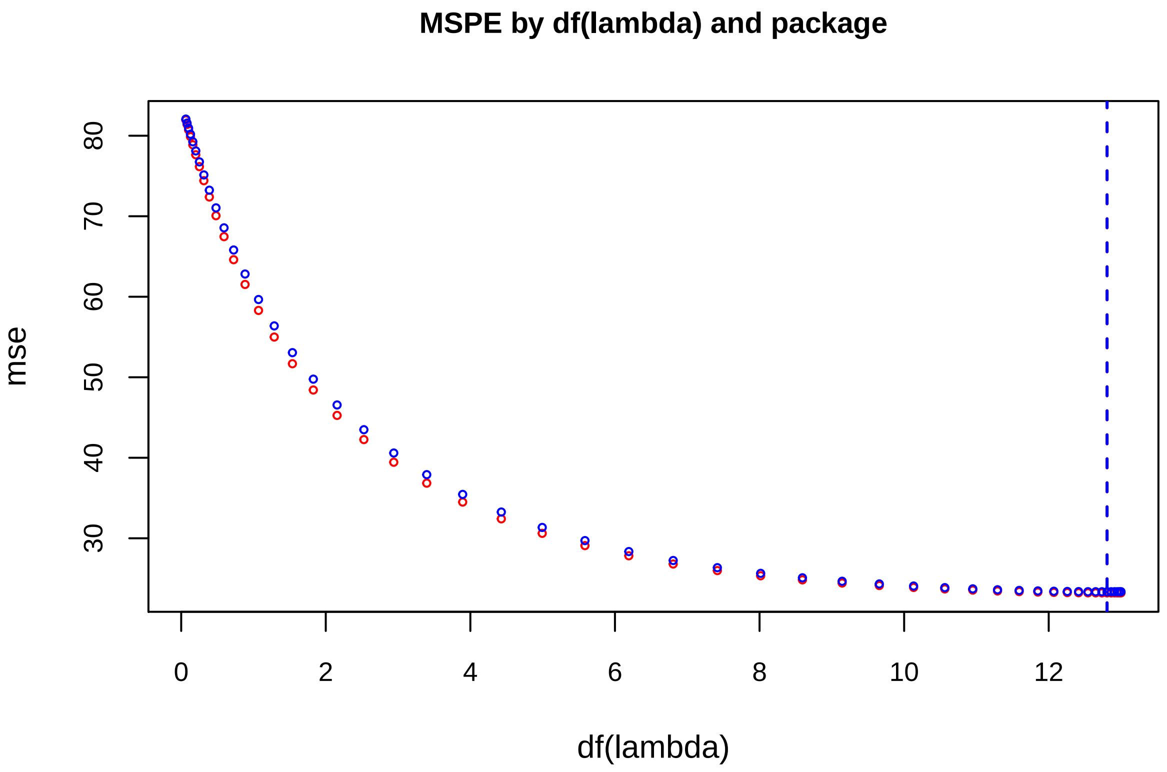
```
##         beta Glmnet lambda beta Theory lambda
## CRIM           -0.89328267       -0.895599267
## ZN              1.07828132        1.078980200
## INDUS           0.10453877        0.108124227
## CHAS            0.69146761        0.691054263
## NOX            -1.99143647       -1.995144347
## RM              2.68210858        2.681405843
## AGE             0.00251577        0.002520772
## DIS            -3.10149659       -3.101994075
## RAD             2.47870989        2.490718852
## TAX            -1.98686431       -1.996108720
## PTRATIO        -1.98122241       -1.982875244
## B               0.84140209        0.841722961
## LSTAT          -3.76063393       -3.760074073
```

As expected, both the value of the lambdas and the value of the beta coefficients are similar. The difference in the beta coefficients might be caused by the tolerance of the algorithm to solve the least squares algorithm.

Finally, we compare the results in the following chart.

```
plot(df.lambda,rev(mse),
     xlab="df(lambda)",
     ylab = "mse",
     main="MSPE by df(lambda) and package",
     cex.main=0.9, cex.axis=0.8, col=2, pch=1, cex=0.5)
abline(v=df.lambda.min,col=2,lty=2,lwd=1.5)
```

```
points(CV.CMEDV.ridge.function[,3],CV.CMEDV.ridge.function[,2],
       pch=1,col=4, cex=0.5)
abline(v=df.opt.ridge.func,col=4,lty=2, lwd=1.5)
```



Clearly there is little difference between the curves.The difference in the values probably is caused by the different procedure used to split the data into 10 folds in each of the algorithms Since the k-folds used in each algorithm were different, so are the values of the expected mean square error. **Thus we have shown that both algorithms are more or less equivalent**