Task 3

David Cardoner & Arnau Mercader

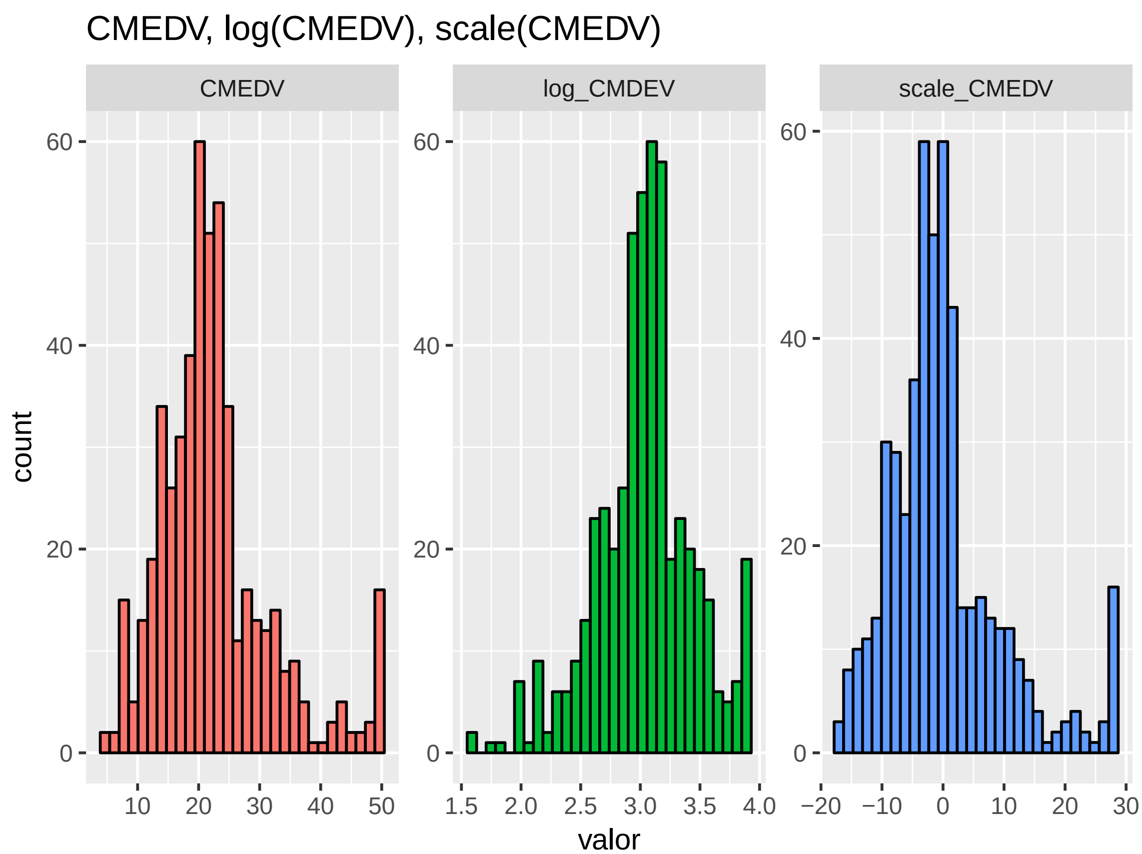
Ap 1.

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. Note: We have some extra data related with coordinates (boston.utm dataset), we merge it with Boston data for gain two extra features. Look first rows of data:

```
     CRIM ZN INDUS CHAS   NOX    RM  AGE    DIS RAD TAX PTRATIO      B
1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90
2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90
3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83
4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63
5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90
6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12
  LSTAT CMEDV coordx  coordy
1  4.98  24.0 338.73 4679.73
2  9.14  21.6 339.23 4683.33
3  4.03  34.7 340.37 4682.80
4  2.94  33.4 341.05 4683.89
5  5.33  36.2 341.56 4684.44
6  5.21  28.7 342.03 4685.09
```

First we are going to explore the response **CMDEV** (corrected variable) and apply some transformation. We are not going to scale the response because we will use intercept in Lasso regression model **(glmnet)** (the best normalization is using log transformation). We consider hot encoding in **RAD** feature and to illustrate we compare results with **caret**, another great package for machine learning problems with R.

```
Boston %>% mutate(log_CMDEV=log(CMEDV),
scale_CMEDV = CMEDV-mean(CMEDV)) %>%
gather(respuesta,valor,c(log_CMDEV,CMEDV,scale_CMEDV)) %>%
ggplot(.,aes(valor,fill=as.factor(respuesta))) +
geom_histogram(bins=30,colour='black') + guides(fill=FALSE) +
ggtitle("CMEDV, log(CMEDV), scale(CMEDV)") + facet_wrap(~respuesta,scales = "free")
```




```
# Define of caret control
set.seed(123)
CARET.TRAIN.CTRL <- trainControl(method="cv",
                                 number = 10,
                                 verboseIter=FALSE)
```

We take a look at the data variables. Find NA's and categorical encoding for variable rad.

```
# Boston$CMEDV <- log(Boston$medv+1)

#str(Boston)
# categorical hot encoding for rad
Boston$RAD <- as.factor(as.character(Boston$RAD))
dummies<-dummyVars(~RAD,data = Boston)


BostonSc <- Boston %>% select_if(is.numeric) %>% mutate_all(funs(scale)) %>% as.data.frame()

# add binary variable
BostonSc$CHAS <- as.numeric(Boston$CHAS)

##number of NA's
sapply(Boston,function(x){sum(which(is.na(x)))})
```

```
   CRIM      ZN   INDUS    CHAS     NOX      RM     AGE     DIS     RAD
      0       0       0       0       0       0       0       0       0
    TAX PTRATIO       B   LSTAT   CMEDV  coordx  coordy
```

```
      0        0        0        0        0        0        0
```

```
# no NA's in data

categorical_1_hot <- predict(dummies,Boston)

# merge data with dummy on RAD
BostonSc <- cbind(BostonSc,categorical_1_hot)
```

We fit a Lasso regression model fixing $\alpha = 1$ in glmnet function.

```
sn<-sample.split(BostonSc,0.75)
BostonSc <- BostonSc %>% select(-c(CMEDV))
train <- BostonSc[sn,]
test <- BostonSc[!sn,]
y <- log(Boston$CMEDV)[sn]
testy <- Boston$CMEDV[!sn]

X_train <- train
X_test <- test

set.seed(123)  # for reproduce output
model_lasso <- train(x=X_train,y=y,
                  method="glmnet",
                  metric="RMSE",
                  maximize=FALSE,
                  trControl=CARET.TRAIN.CTRL,
                  tuneGrid=expand.grid(alpha=1,  # Lasso regression
                                       lambda=c(1,0.1,0.05,0.01,seq(0.009,0.001,-0.001),
                                            0.00075,0.0005,0.0001)))
```

```r
# grid search and metrics
model_lasso
```

```
glmnet

380 samples
 23 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 342, 343, 342, 341, 341, 342, ...
Resampling results across tuning parameters:

  lambda   RMSE       Rsquared   MAE
  0.00010  0.1890880  0.8094378  0.1361911
  0.00050  0.1890580  0.8096037  0.1361356
  0.00075  0.1890570  0.8097450  0.1360847
  0.00100  0.1890286  0.8098990  0.1360226
  0.00200  0.1889174  0.8105757  0.1354777
  0.00300  0.1893160  0.8102295  0.1353301
  0.00400  0.1899913  0.8093770  0.1353514
  0.00500  0.1909329  0.8077847  0.1356070
  0.00600  0.1917016  0.8063279  0.1359860
  0.00700  0.1920756  0.8057361  0.1360848
  0.00800  0.1923399  0.8053020  0.1362607
  0.00900  0.1928314  0.8044579  0.1366613
  0.01000  0.1934024  0.8034854  0.1371471
  0.05000  0.2168481  0.7766437  0.1572053
  0.10000  0.2499482  0.7393119  0.1823589
  1.00000  0.4154508        NaN  0.3066650

Tuning parameter 'alpha' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 1 and lambda = 0.002.
```

cv.glmnet function with 10 fold CV and default lambdas (100 values)

```r
X_train <- data.matrix(X_train)
X_test <- data.matrix(X_test)
set.seed(123)
# by default uses 10 fold cross validation. Uses 100 lambdas
modelglmnet <- cv.glmnet(X_train,y,family = "gaussian",
                         alpha = 1,standardize = FALSE,
                         intercept = TRUE,type.measure = "mse")

# plot(modelglmnet,main = "Lasso")
# coef(modelglmnet, s = "lambda.1se")

cat("The best lambda is",modelglmnet$lambda.min,"\n")
```

```
The best lambda is 0.0003164707
```

```r
cat("Metric MSE value is:",min(modelglmnet$cvm))
```

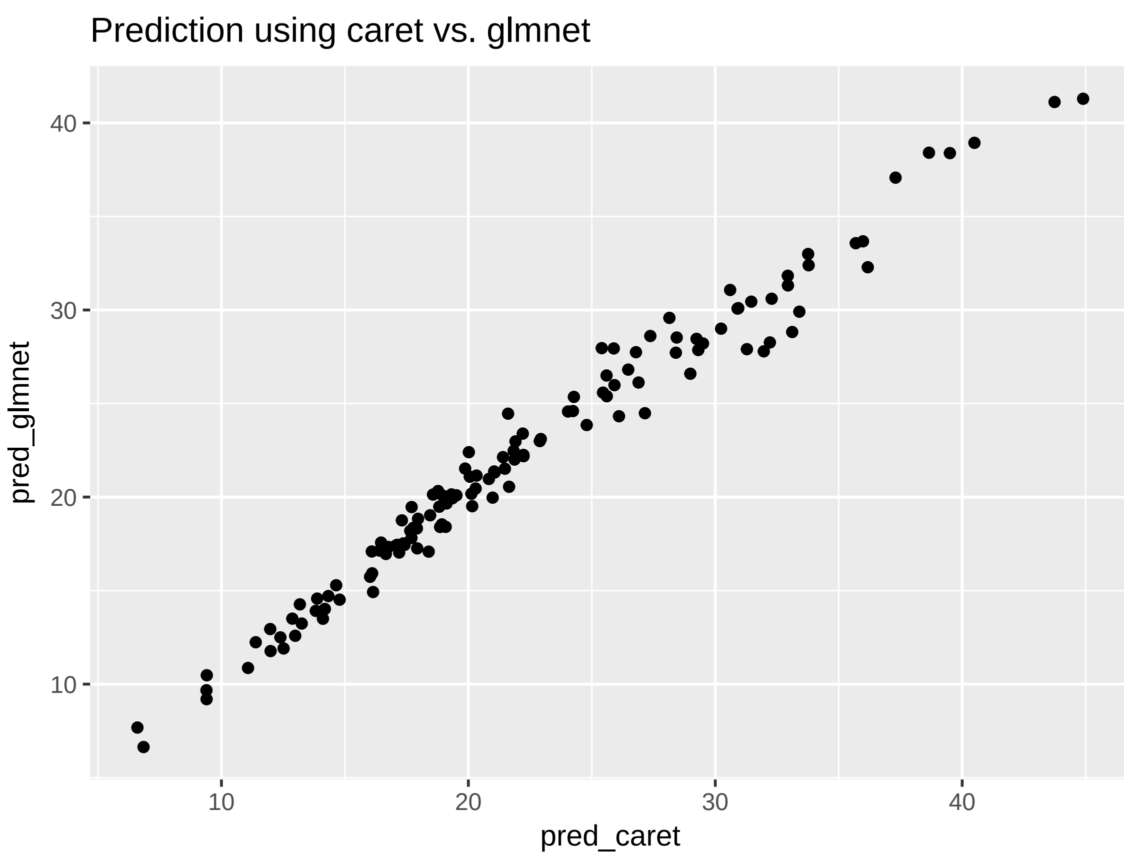
```
Metric MSE value is: 0.03732293
```

Glmnet with 10 fold cross-validation and same lambdas as caret

```
modelglmnet2 <- cv.glmnet(X_train,y,family = "gaussian",
                          alpha = 1,standardize = FALSE,intercept = TRUE,lambda = c(1,0.1,0.05,0.01,seq
                          type.measure = "mse")
```

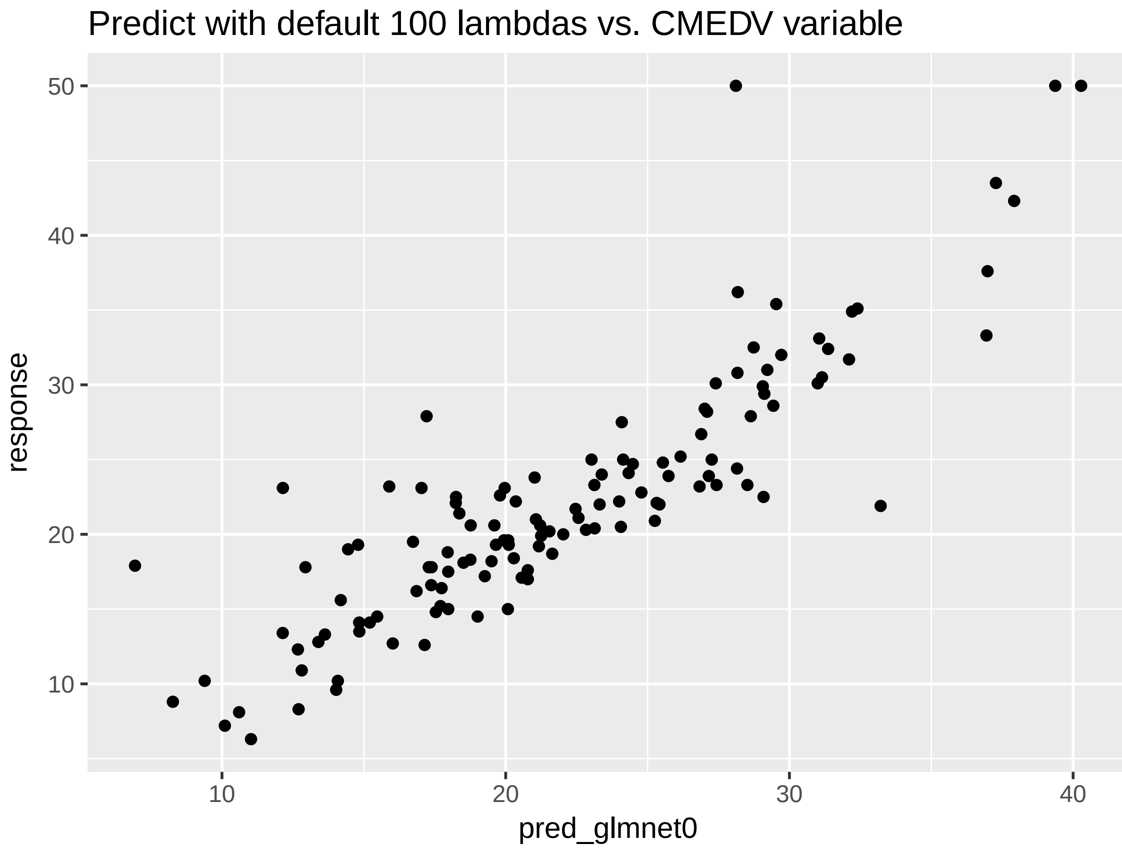
Now we make predictions with glmnet. 10 fold CV is performed with $\alpha = 1$ for Lasso regression. Intercept is set to True because **response** is not centered (we apply log transformation).

```
pred_glmnet = exp(predict(modelglmnet2,newx=X_test))
pred_caret = exp(predict(model_lasso,newdata = X_test))
# cbind(pred,exp(testy)-1) %>% View()

data_pred <- data.frame(pred_glmnet=pred_glmnet,pred_caret=pred_caret)
data_pred %>%
ggplot(.,aes(pred_caret,pred_glmnet)) + geom_point() +
ggtitle("Prediction using caret vs. glmnet")
```



We observe that our predictions are very correlated. Now we examine prediction of glmnet model (using default lambdas) vs. real values. We have problems with high values, we predict 30 to 40 and real value are 40 to 50, but in interval 10-30 we have good performance in general terms.

```
# prepare predict object
pred_glmnet0 = exp(predict(modelglmnet,newx=X_test))
data_pred0 <- data.frame(pred_glmnet0=pred_glmnet0,response=(testy))

data_pred0 %>%   ggplot(.,aes(pred_glmnet0,response)) + geom_point() +
ggtitle("Predict with default 100 lambdas vs. CMEDV variable")
```


Predict with default 100 lambdas vs. CMEDV variable
response
50
40
30
20
10
10
20
30
40
pred_glmnet0

MSE metric obtained in 3 methods

```r
# nice package for calculate a lot of metrics:
require(Metrics)
cat("With tune lambdas")
```

```
With tune lambdas
```

```r
mse(pred_glmnet,testy)
```

```
[1] 16.27008
```

```r
cat("With default lambdas")
```

```
With default lambdas
```

```r
mse(pred_glmnet0,testy)
```

```
[1] 17.05281
```

```r
cat("Using caret package and tuning lambdas:")
```

```
Using caret package and tuning lambdas:
```

```r
mse(pred_caret,testy)
```

```
[1] 14.19313
```

```r
cat("The best metric is obtained with caret and in second
    place with default lambdas using glmnet package. \n")
```

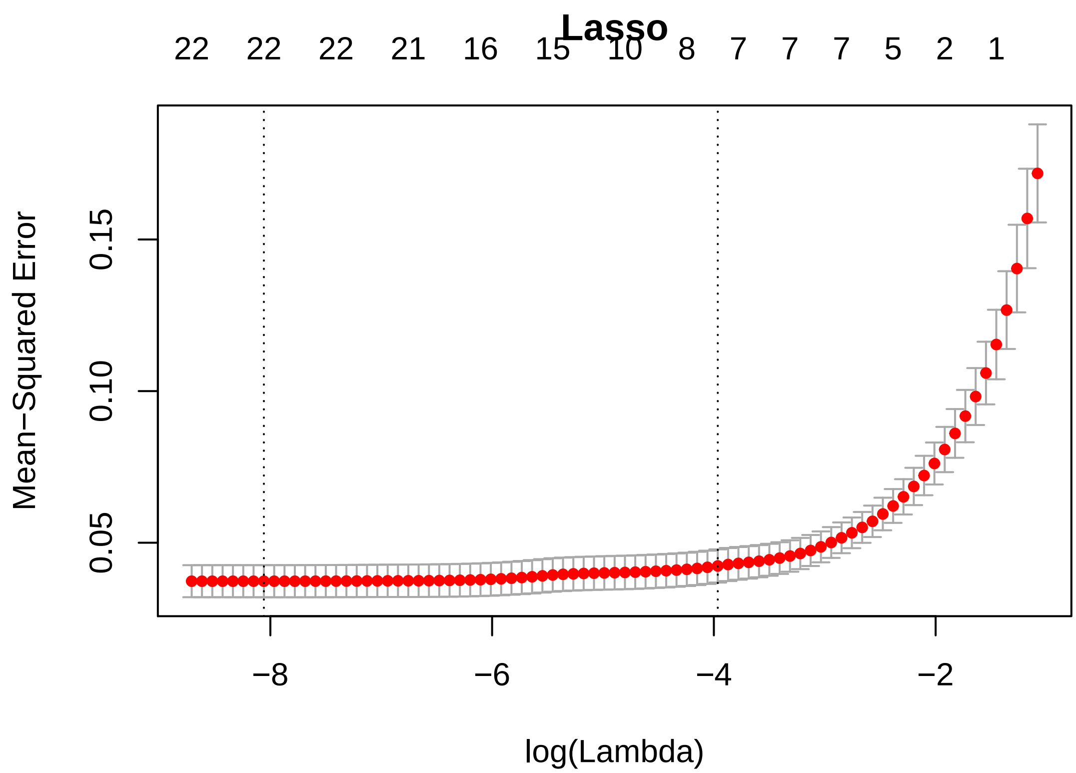
```
The best metric is obtained with caret and in second
    place with default lambdas using glmnet package.
```

Interpretation of estimated best model (object modelglmnet)

```r
cat("Plot lambdas vs. metric")
```

```
Plot lambdas vs. metric
```

```r
plot(modelglmnet,main = "Lasso")
```




```r
cat("Coef. with best lambda")
```

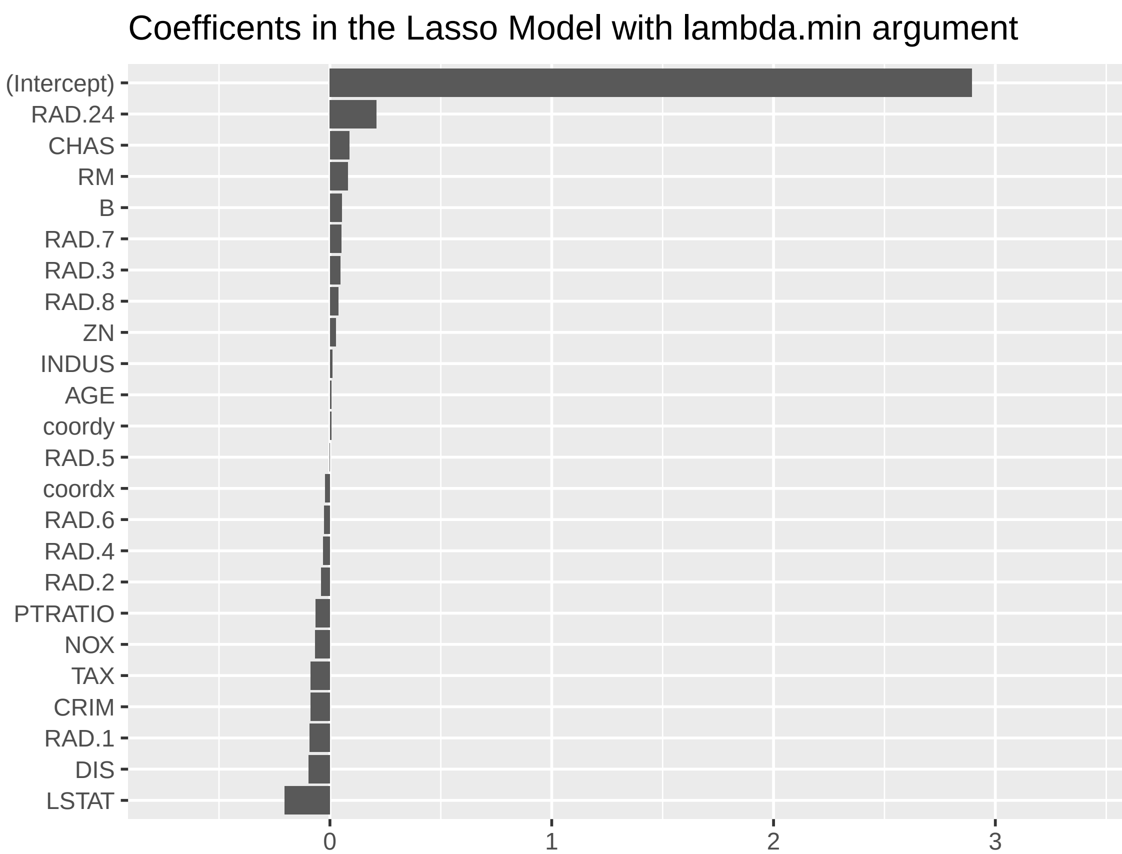
```
Coef. with best lambda
```

```r
coef(modelglmnet, s = "lambda.min")
```

```
24 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)  2.894907305
CRIM        -0.087763136
ZN           0.025974442
INDUS        0.010436016
NOX         -0.066208646
RM           0.079929443
AGE          0.005804724
DIS         -0.095755444
TAX         -0.086770824
PTRATIO     -0.064018413
B            0.052841977
LSTAT       -0.204805934
coordx      -0.021318660
coordy       0.005272290
CHAS         0.086445578
RAD.1       -0.090280867
RAD.2       -0.039367820
RAD.24       0.210303448
RAD.3        0.046679742
```

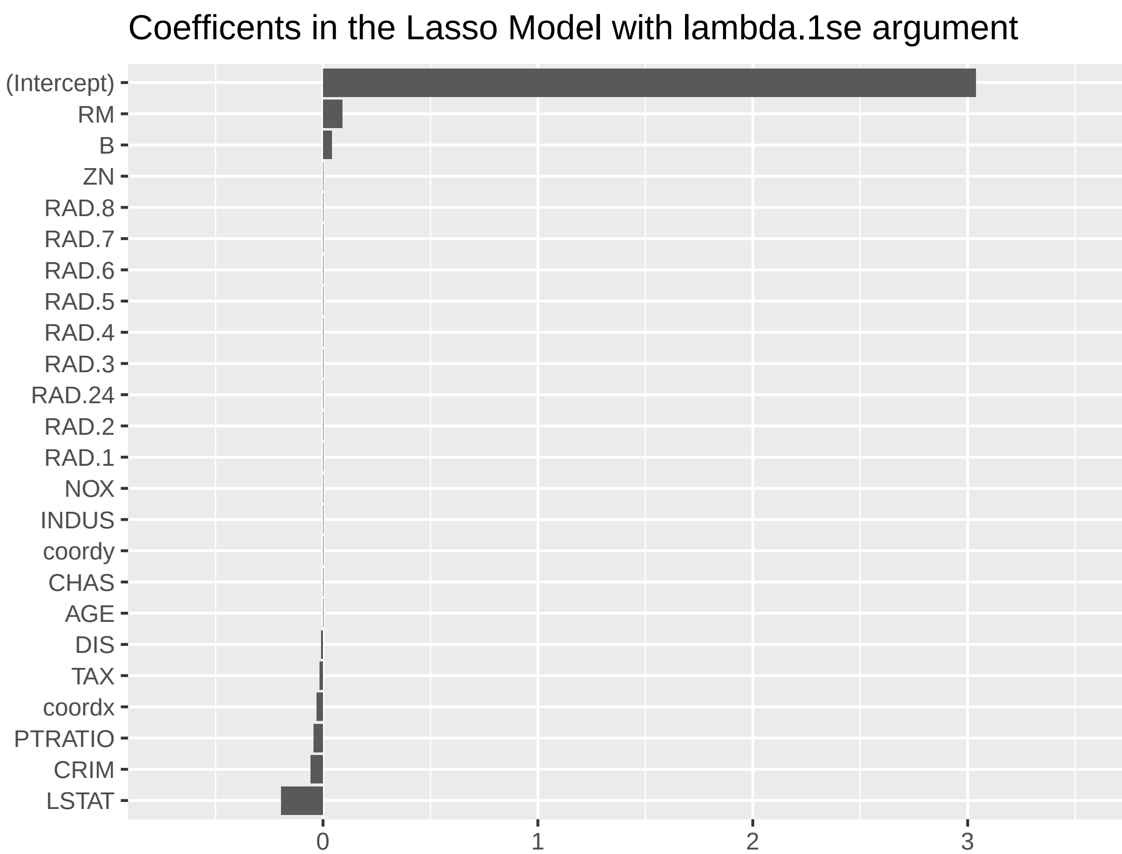
```
RAD.4       -0.031011868
RAD.5        .
RAD.6       -0.026420467
RAD.7        0.051742954
RAD.8        0.037723081
```

```
imp_coef <- as.data.frame(as.matrix(coef(modelglmnet, s = "lambda.min")))
imp_coef$coef.name <- rownames(imp_coef)
imp_coef$coef.value <- imp_coef$`1`

ggplot(imp_coef) +
    geom_bar(aes(x=reorder(coef.name,coef.value),y=coef.value),
             stat="identity") +
    ylim(min(imp_coef$coef.value)-.5,max(imp_coef$coef.value)+.5) +
    coord_flip() +
    ggtitle("Coefficents in the Lasso Model with lambda.min argument") +
    theme(axis.title=element_blank())
```




```
imp_coef <- as.data.frame(as.matrix(coef(modelglmnet, s = "lambda.1se")))
imp_coef$coef.name <- rownames(imp_coef)
imp_coef$coef.value <- imp_coef$`1`

ggplot(imp_coef) +
    geom_bar(aes(x=reorder(coef.name,coef.value),y=coef.value),
             stat="identity") +
    ylim(min(imp_coef$coef.value)-.5,max(imp_coef$coef.value)+.5) +
    coord_flip() +
```

```r
    ggtitle("Coefficents in the Lasso Model with lambda.1se argument") +
    theme(axis.title=element_blank())
```




```r
cat("Coef. with lambda in 1 square error")
```

```
Coef. with lambda in 1 square error
```

```r
coef(modelglmnet, s = "lambda.1se")
```

```
24 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept)  3.036509951
CRIM        -0.057173387
ZN           .
INDUS        .
NOX          .
RM           0.088655959
AGE          .
DIS         -0.007803401
TAX         -0.015484432
PTRATIO     -0.042484451
B            0.039729247
LSTAT       -0.195373670
coordx      -0.028789905
coordy       .
CHAS         .
RAD.1        .
```

```
RAD.2         .
RAD.24        .
RAD.3         .
RAD.4         .
RAD.5         .
RAD.6         .
RAD.7         .
RAD.8         .
```

```
cat("Notice that loosing few value of metric we obtain a model
with less parameters, \n indicating that lasso obtain sparsity solutions.")
```

```
Notice that loosing few value of metric we obtain a model
with less parameters,
 indicating that lasso obtain sparsity solutions.
```

Ap 2.

Perform ridge regression and compare with the ridge realised in previous practice. To perform ridge regression we fix the value of $\alpha = 0$.

```
# in this exercise we go to use log response.
y <- log(Boston$CMEDV)
X_features <- Boston[,-14]
X_features <- sapply(X_features,as.numeric)
X_features <- as.matrix(X_features)

set.seed(123)
modelglmnet_ridge <- cv.glmnet(x=X_features,y = y ,
family = "gaussian",alpha = 0,
standardize = TRUE,intercept = TRUE,type.measure = "mse",nfolds = 10)
```

```
CV_kfolds_ridge <- function(X,y,folds,lambda.v,seed_value=NULL) {

  # as.numeric all covariables X
  X <- sapply(X,as.numeric)

  # scale data & input data to -1
  X[is.na(X)] <- -1
  mean_X <- apply(X,2,mean)
  sd_X <- apply(X,2,sd)
  y_mean <- mean(y)

  X <- as.matrix(scale(X, center=mean_X, scale=sd_X))
  y <- as.matrix(scale(y, center=y_mean, scale=FALSE))

  data <- cbind(y,X)

  if(is.null(seed_value)){1234}
  set.seed(seed_value)
  # permut data and make k folds
  data<-data[sample(nrow(data)),]


  # Create k size folds
```

```r
  kfolds <- cut(seq(1,nrow(data)),breaks=folds,labels=FALSE)


  # result for lambda l
  cv_kfolds_mpse_lambda <- numeric(length(lambda.v))


  for (l in 1:length(lambda.v)){

    lambda <- lambda.v[l]
    cv_kfolds_mpse <- numeric(folds)

    for (i in 1:folds){

    idx <- which(kfolds==i)
    train <- data[-idx, ]
    test <- data[idx, ]
    n_train <- dim(train)[1]
    p_train <- dim(train)[2]-1
    n_test <- dim(test)[1]
    p_test <- dim(test)[2]-1


    beta.path <- matrix(0,nrow=1, ncol=p_train)


    XtX <- t(train[,-1]) %*% train[,-1]
    H.lambda.aux <- t(solve(XtX + lambda*diag(1,p_train))) %*% t(train[,-1])
    beta.path <-  H.lambda.aux %*% as.matrix(train[,1])
    hat.Y_val <- test[,-1] %*% beta.path
    cv_kfolds_mpse[i] <- sum((test[,1]-hat.Y_val)^2)/n_test
    }

    cv_kfolds_mpse_lambda[l] <- mean(cv_kfolds_mpse)
    cv_kfolds_mpse <- NULL
  }


  min_lambda <- lambda.v[which.min(cv_kfolds_mpse_lambda)]
  cat("Resultados considerando k=",folds,"folds: \n")
  cat("\n")
  cat("El lambda que hace mínimo el PMSE es: ",min_lambda,"\n")
  cat("\n")
  cat("El PMSE es: ",min(cv_kfolds_mpse_lambda) )

  plot(lambda.v,cv_kfolds_mpse_lambda,main = 'Valores lambda vs. k-fold MSPE',
       ylab='MPSE error',xlab="values of lambda")

}

lambda.search = modelglmnet_ridge$lambda
```

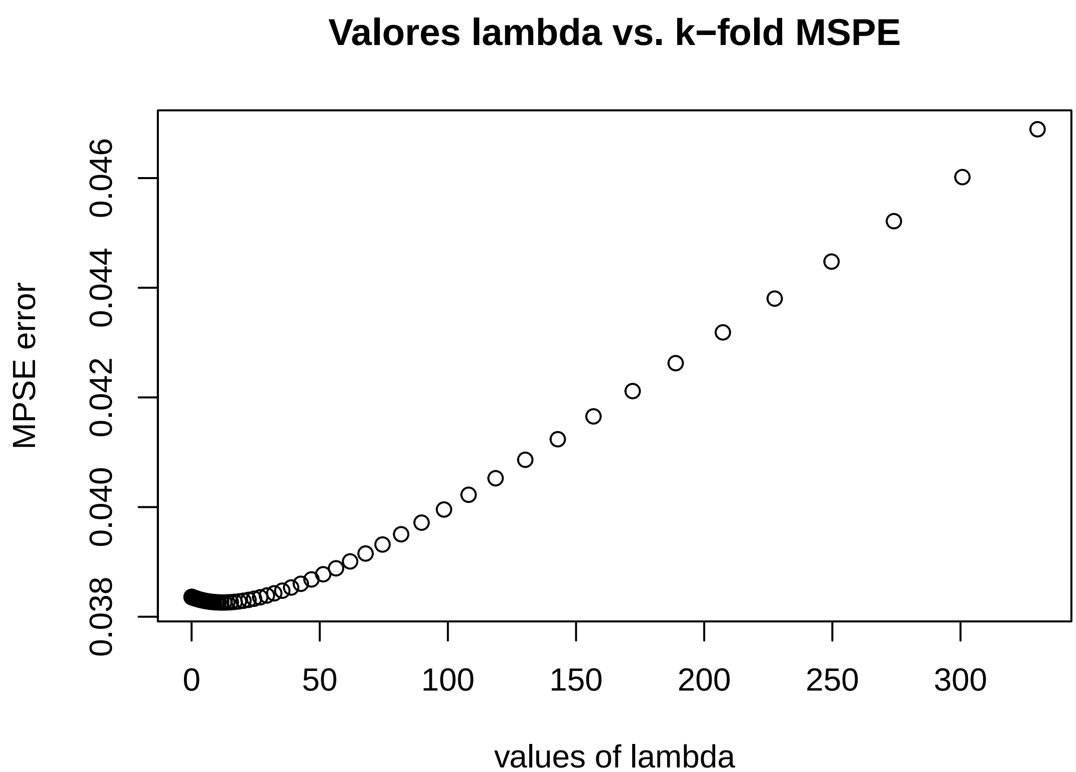
```r
# CV_kfolds_ridge(X=Boston[,-14],y=log(Boston$CMEDV),
#                  folds=10,lambda.v=lambda.search,seed_value = 123)
```

Compare methods

Using 10 fold cv (with seed 123), with our function we obtain:

```
lambda.search = modelglmnet_ridge$lambda

CV_kfolds_ridge(X=Boston[,-14],y=log(Boston$CMEDV),folds=10,lambda.v=lambda.search,seed_value = 123)
```

```
Resultados considerando k= 10 folds:

El lambda que hace mínimo el PMSE es:  12.71908

El PMSE es:  0.03826021
```

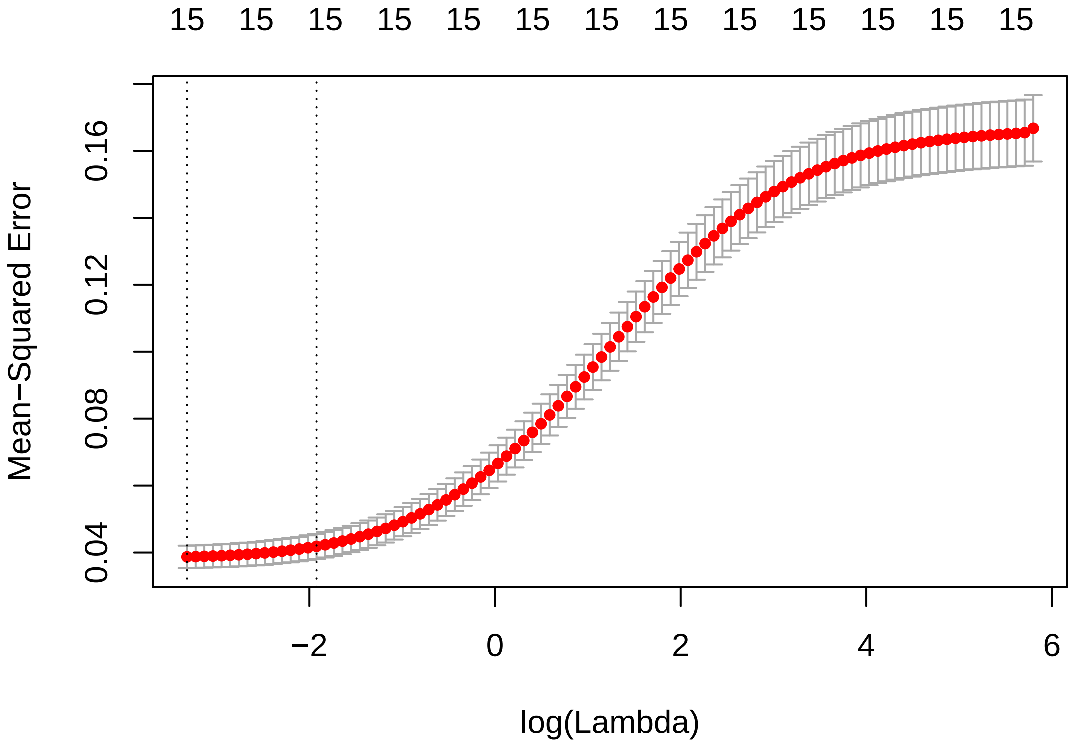


With glmnet function we obtain:

```
cat("PMSE with glmnet is:",min(modelglmnet_ridge$cvm),"\n")
```

```
PMSE with glmnet is: 0.03871147
```

```
plot(modelglmnet_ridge)
```



With two methods, obtain closest PMSE.