Assignment 2
Ridge Regression

Víctor Duque, Geraldo Gariza, Jan Leyva and Andreu Meca
Universitat Politècnica de Catalunya

28th of February, 2021

Ridge Regression

1. Choosing the penalization parameter lambda

Write an R function implementing the ridge regression penalization parameter λ choice based on the minimization of the mean squared prediction error in a validation set ($MSPE_{val}(\lambda)$).

```
# Ridge Regression function
ridge_regression <- function(x.train, y.train, x.val, y.val, lambda.v){

  Y <- scale(y.train, center=TRUE, scale=FALSE)
  X <- scale(as.matrix(x.train), center=TRUE, scale=TRUE)
  y.val.m <- scale(y.val, center=TRUE, scale=FALSE)
  x.val.m <- scale(as.matrix(x.val), center=TRUE, scale=TRUE)
  n <- dim(X)[1]
  p <- dim(X)[2]
  XtX <- t(X)%*%X
  d2 <- eigen(XtX,symmetric = TRUE, only.values = TRUE)$values
  n.lambdas <- length(lambda.v)

  MSPE <- rep(0, n.lambdas)
  beta.path <- matrix(0,nrow=n.lambdas, ncol=p)
  diag.H.lambda <- matrix(0,nrow=n.lambdas, ncol=n)
  for (l in 1:n.lambdas){
   lambda <- lambda.v[l]
    H.lambda.aux <- t(solve(XtX + lambda*diag(1,p))) %*% t(X)
    beta.path[l,] <-  H.lambda.aux %*% Y
    MSPE[l] <- sum((y.val.m - x.val.m%*%beta.path[l,])^2)/length(y.val.m)
  }
```

```
  df <- cbind(as.data.frame(MSPE), as.data.frame(lambda.v))
  plot_mspe <- ggplot(data = df,
                      mapping = aes(x = log(1+lambda.v)-1, y = MSPE)) +
    geom_point() +
    labs(x = "Lambda (log(1+lambda.v)-1)",
         y = "MSPE",
         title = "Ridge Regression MSPE depending on lambda") +
    theme_bw()

  Min_MSPE <- min(MSPE)

  list_func <- list(plot_mspe,
                    MSPE,
                    paste0("Min. MSPE: ",
                           Min_MSPE))

  return(list_func)
}
```

Write an R function implementing the ridge regression penalization parameter λ choice based on k-fold cross validation ($MSPE_{k-CV}(\lambda)$).

```
#Ridge Regression k-fold cross validation function
ridge_regression_k_fold_CV <- function(x.train, y.train, lambda.v, k){

  Y <- scale(y.train, center=TRUE, scale=FALSE)
  X <- scale(as.matrix(x.train), center=TRUE, scale=TRUE)
  n <- dim(X)[1]
  p <- dim(X)[2]
  XtX <- t(X)%*%X
```

```
d2 <- eigen(XtX,symmetric = TRUE, only.values = TRUE)$values
n.lambdas <- length(lambda.v)

set.seed(123)
fold <- sample(1:k, nrow(x.train), replace = T)
PMSE.CV.H.lambda <- rep(0, n.lambdas)
for (l in 1:n.lambdas){
  lambda <- lambda.v[l]
  for (i in 1:k){
    X.train <- X[fold!=i,]
    Y.train <- Y[fold!=i,]

    n <- dim(X.train)[1]
    p <- dim(X.train)[2]

    beta.path <- matrix(0,nrow=n.lambdas, ncol=p)
    diag.H.lambda <- matrix(0,nrow=n.lambdas, ncol=n)

    XtX <- t(X.train)%*%X.train

    H.lambda.aux <- t(solve(XtX + lambda*diag(1,p))) %*% t(X.train)
    beta.path[l,] <-  H.lambda.aux %*% Y.train
    H.lambda <- X.train %*% H.lambda.aux
    diag.H.lambda[l,] <- diag(H.lambda)
    hat.Y <- X.train %*% beta.path[l,]
    PMSE_temp <- sum(((Y.train-hat.Y)/(1-diag.H.lambda[l,]))^2)
    PMSE.CV.H.lambda[l] <- PMSE.CV.H.lambda[l] + PMSE_temp
  }
PMSE.CV.H.lambda[l] <- PMSE.CV.H.lambda[l]/n
```

```
  }

  df <- cbind(as.data.frame(PMSE.CV.H.lambda), as.data.frame(lambda.v))
  plot_mspe <- ggplot(data = df,
                      mapping = aes(x = log(1+lambda.v)-1,
                                    y = PMSE.CV.H.lambda)) +
    geom_point() +
    labs(x = "Lambda (log(1+lambda.v)-1)",
         y = "MSPE",
         title = "Ridge Regression k-fold CV MSPE depending on lambda") +
    theme_bw()

  Min_PMSE.CV.H.lambda <- round(min(PMSE.CV.H.lambda),digits = 5)
  lambda_pos <- which.min(PMSE.CV.H.lambda)
  lambda_optima <- round(lambda.v[lambda_pos],5)

  list_func <- list(plot_mspe,
                    PMSE.CV.H.lambda,
                    paste("Min. MSPE: ",
                          Min_PMSE.CV.H.lambda),
                    paste("Position of best lambda:",
                          lambda_pos),
                    paste("Value of best lambda:",
                          lambda_optima))

  return(list_func)
}
```

Consider the prostate date used in class. Use your routines to choose the penalization parameter λ by the following criteria: behavior in the validation set (the 30 observations

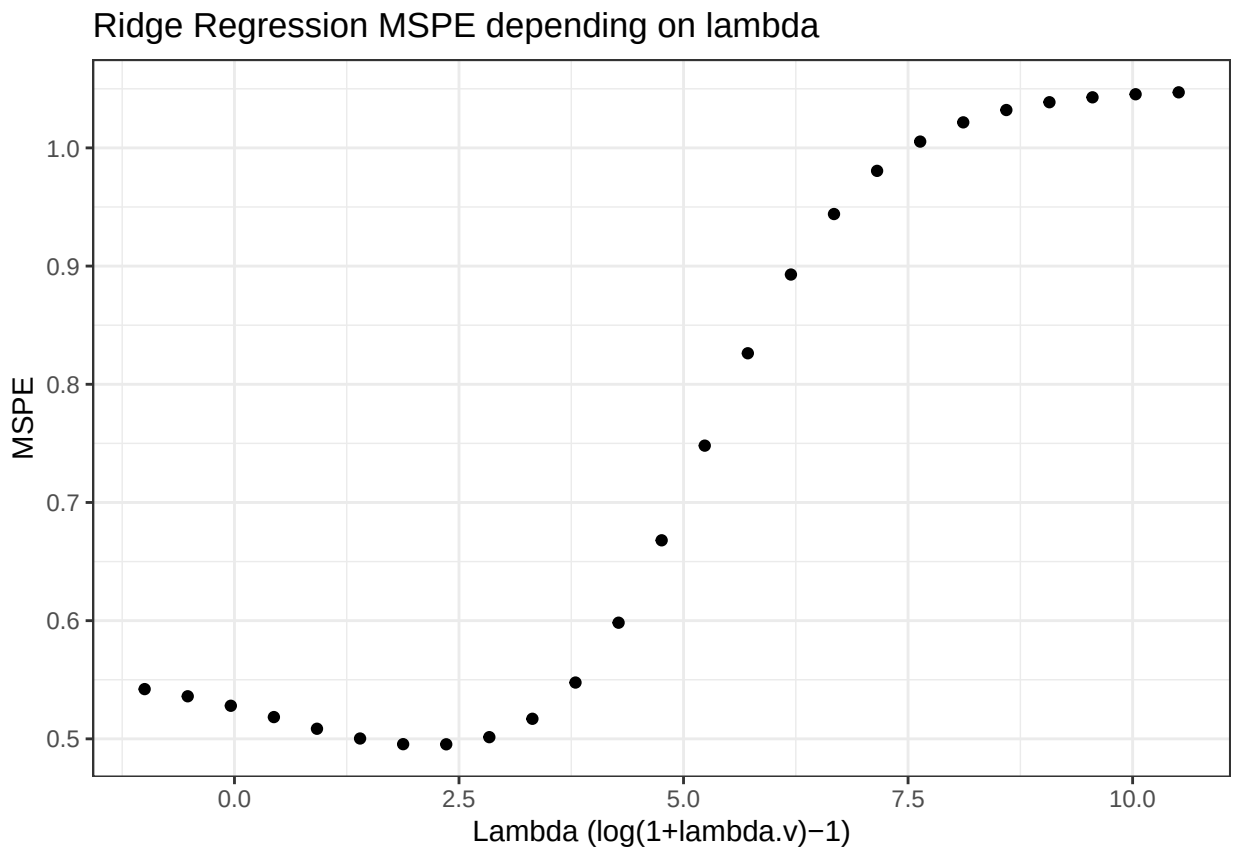
not being in the training sample); 5-fold and 10-fold cross-validation. Compare your results with those obtained when using leave-one-out and generalized cross-validation.

```
prostate <- read.table("prostate_data.txt", header=TRUE, row.names = 1)
y.train <- prostate$lpsa[which(prostate$train==TRUE)]
x.train <- prostate[which(prostate$train==TRUE),1:8]
y.val <- prostate$lpsa[which(prostate$train==FALSE)]
x.val <- prostate[which(prostate$train==FALSE),1:8]
```

```
lambda.max <- 1e5
n.lambdas <- 25
lambda.v <- exp(seq(0,log(lambda.max+1),length=n.lambdas))-1
```

```
ridge_regression(x.train, y.train, x.val, y.val, lambda.v)
```

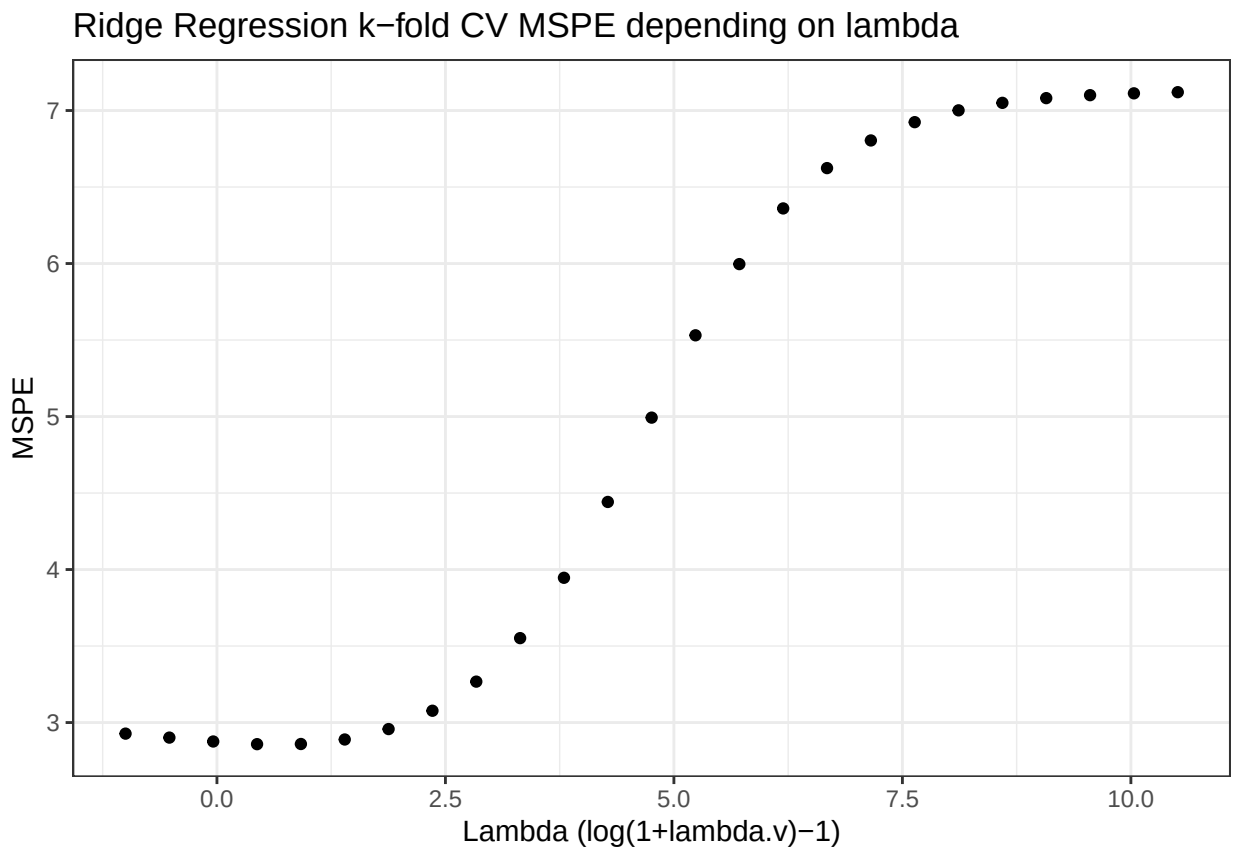
```
## [[1]]
```



```
##
## [[2]]
##  [1] 0.5421042 0.5360290 0.5279985 0.5184215 0.5085513 0.5003000 0.4954982
##  [8] 0.4953927 0.5014366 0.5169823 0.5476379 0.5982753 0.6679911 0.7480967
## [15] 0.8262517 0.8928568 0.9440439 0.9805716 1.0053507 1.0216070 1.0320442
## [22] 1.0386541 1.0428041 1.0453957 1.0470086
##
## [[3]]
## [1] "Min. MSPE: 0.495392666048034"
```

```
ridge_regression_k_fold_CV(x.train, y.train, lambda.v, 5)
```

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

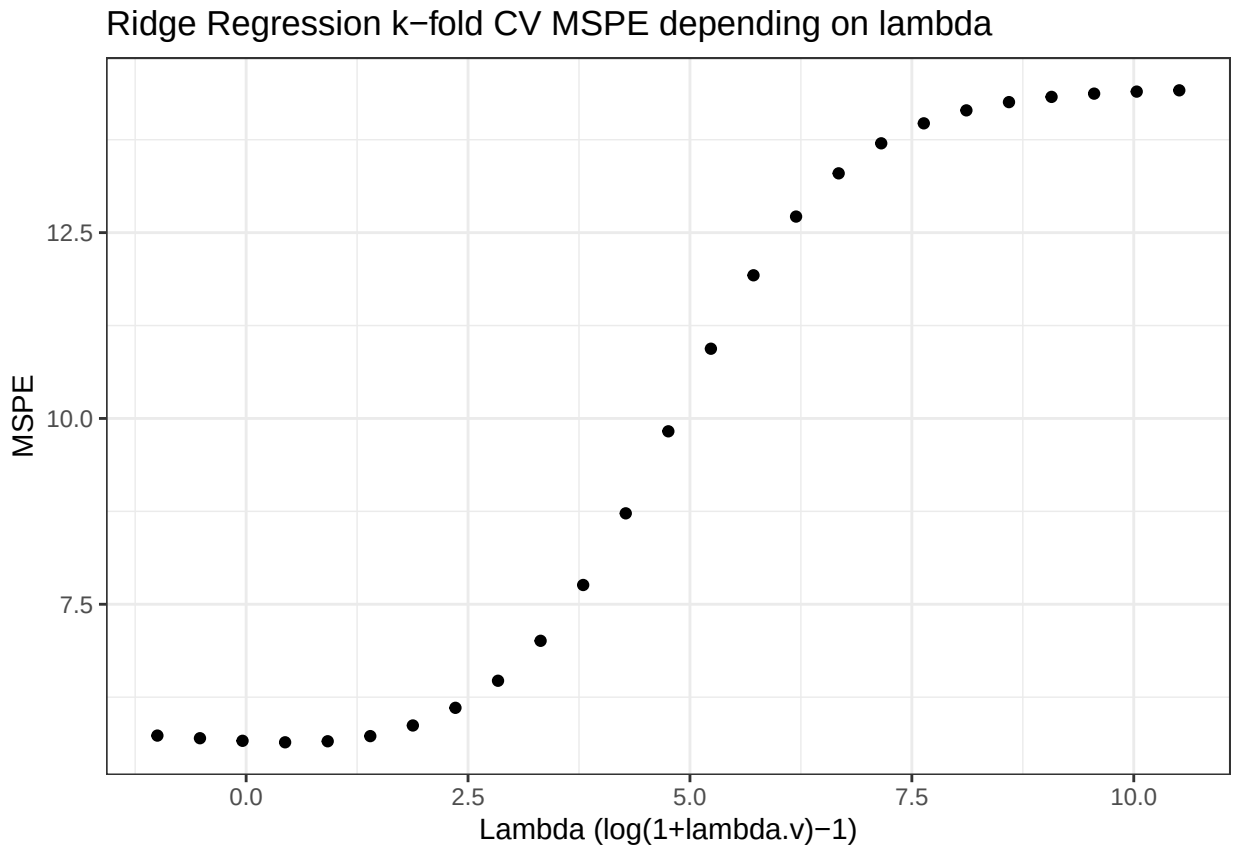



```
##
```

```
## [[2]]
##  [1] 2.927576 2.901732 2.876160 2.858793 2.859786 2.889342 2.957508 3.077212
##  [9] 3.267524 3.551745 3.946217 4.441845 4.992935 5.530923 5.996199 6.360044
## [17] 6.623608 6.804422 6.923975 7.001133 7.050161 7.081010 7.100300 7.112315
## [25] 7.119782
##
## [[3]]
## [1] "Min. MSPE:  2.85879"
##
## [[4]]
## [1] "Position of best lambda: 4"
##
## [[5]]
## [1] "Value of best lambda: 3.21697"
```

```
ridge_regression_k_fold_CV(x.train, y.train, lambda.v, 10)
```

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




```
##
## [[2]]
##  [1]  5.733332  5.697925  5.663178  5.642936  5.657017  5.726313  5.869334
##  [8]  6.106754  6.471108  7.008332  7.759375  8.723375  9.826949 10.938046
## [15] 11.926057 12.716094 13.297860 13.701544 13.970493 14.144925 14.256112
## [22] 14.326210 14.370097 14.397455 14.414464
##
## [[3]]
## [1] "Min. MSPE:  5.64294"
##
## [[4]]
## [1] "Position of best lambda: 4"
##
## [[5]]
```

```
## [1] "Value of best lambda: 3.21697"
```

We have the value of each of the models:

Leave one out CV:

lambda: 3.21697

Min PMSE: 0.5572292

position: 4

Generalized CV:

lambda: 5.812932

Min PMSE: 0.5572324

position: 5

Validation set:

lamda: 27,72993

Min PMSE: 0.4953927

position: 8

K-fold=5 CV:

lambda: 3.21697

Min PMSE: 2.85879

position: 4

K-fold=10 CV:

lambda: 3.21697

Min PMSE: 5.64294

position: 4

We can see that 3 out of the 5 models choose $\lambda = 3.21697$ as the one that optimizes the regression.

2. Ridge regression for the Boston Housing

The Boston House-price dataset concerns housing values in 506 suburbs of Boston corresponding to year 1978. They are available at the library MASS.

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

For the Boston House-price corrected dataset use ridge regression to fit the regression model where the response is MEDV and the explanatory variables are the remaining 13 variables in the previous list. Try to provide an interpretation to the estimated model.

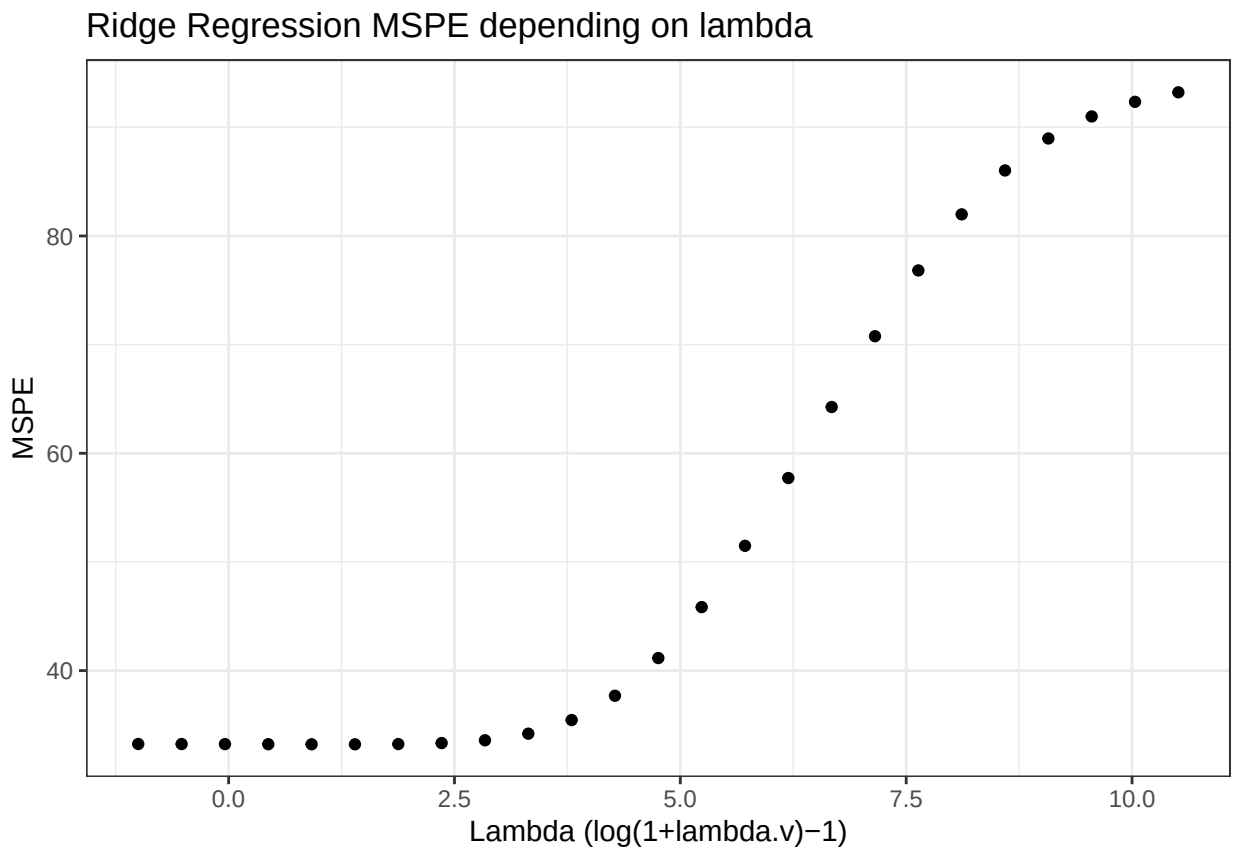
```
data(Boston)
d<-Boston

set.seed(100)

index = sample(1:nrow(d), 0.7*nrow(d))

train = d[index,] # Create the training data
test = d[-index,]

Y <- scale( train[,14], center=TRUE, scale=FALSE)
X <- scale(as.matrix(train[,c(-14,-4)]), center=TRUE, scale=TRUE)

y.val <- scale( test[,14], center=TRUE, scale=FALSE)
x.val <- scale(as.matrix(test[,c(-14,-4)]), center=TRUE, scale=TRUE)


X<-cbind(X,train[,4])
```

```
x.val<-cbind(x.val,test[,4])


n <- dim(X)[1]
p <- dim(X)[2]



lambda.max <- 1e5
n.lambdas <- 25
lambda.v <- exp(seq(0,log(lambda.max+1),length=n.lambdas))-1
```

Ridge regression function:

```
ridge_regression (x.train=X, y.train=Y, x.val=x.val, y.val=y.val, lambda.v=lambda.v)
```

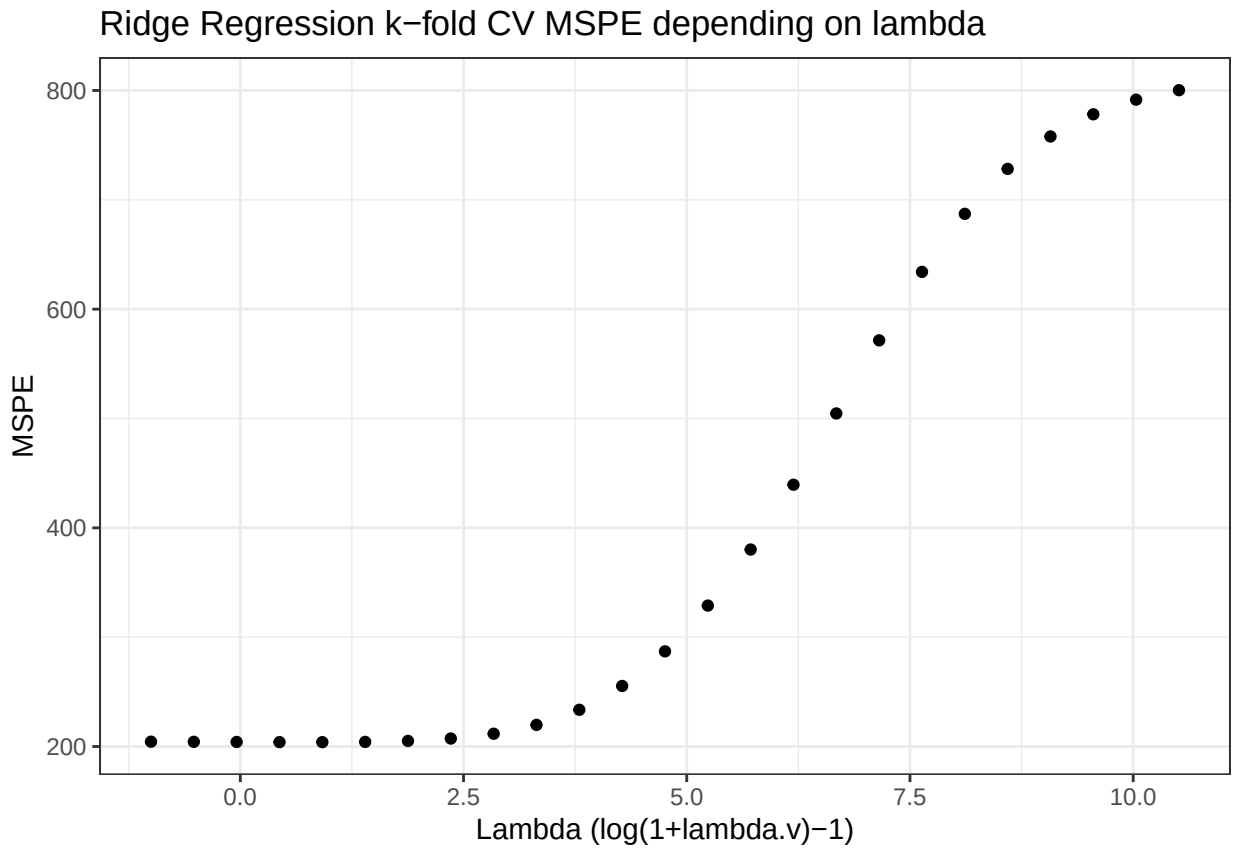
```
## [[1]]
```



```
##
## [[2]]
##  [1] 33.24812 33.24318 33.23636 33.22791 33.21983 33.21850 33.24125 33.33178
##  [9] 33.58846 34.19868 35.45192 37.68792 41.15409 45.84324 51.48334 57.72080
## [17] 64.25807 70.77281 76.82667 81.98744 86.02610 88.96715 90.99584 92.34299
## [25] 93.21508
##
## [[3]]
## [1] "Min. MSPE: 33.2184981802302"
```

```
ridge_regression_k_fold_CV(x.train=X, y.train=Y, lambda.v=lambda.v, k=10)
```

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




```
##
```

```
## [[2]]
##  [1] 204.4426 204.3275 204.1840 204.0439 203.9982 204.2444 205.1452 207.3094
##  [9] 211.7062 219.7985 233.5986 255.4097 287.0355 328.9137 380.1804 439.4544
## [17] 504.6072 571.4791 634.0997 687.1782 728.2913 757.9360 778.2261 791.6254
## [25] 800.2673
##
## [[3]]
## [1] "Min. MSPE:  203.99824"
##
## [[4]]
## [1] "Position of best lambda: 5"
##
## [[5]]
## [1] "Value of best lambda: 5.81293"
```

We find that the best lambda value is: 5.81293 and it is in the fifth iteration of lambda values. Also, the minimum value of MSPE in k-fold=10 is: 203.99824.