Advanced Statistical Modelling: Ridge Regression

Ricard Meyerhofer & Joel Cantero

29/10/2019

Choosing the penalization parameter λ

The objective of this exercise is to implement Ridge Regression with two different approaches: $MSPE_{val}(\lambda)$ and $MSPE_{k-CV}(\lambda)$. In both cases we are going to take as input data the following:

- Matrix x and vector y corresponding to the training sample.
- Matrix x_{val} and vector y_{val} corresponding to the validation set.
- Vector *lambda.v* of candidate values for λ.

We are going to output for each element λ in *lambda.v* and the value of the $MSPE_{val}(\lambda)$ / $MSPE_{k-CV}(\lambda)$. Furthermore, we are going to plot these values against $log(1+\lambda)-1$.

Once we have build these two functions, we are going to use the prostate data used in class. We are going to choose a λ according to:

- Behaviour in the validation set (30 validations not included in the training sample)
- 5-fold, 10-fold cross-validation.
- Compare our results with those obtained with leave-one-out and generalized cross-validation.

Ridge regression based on $MSPE_{val}(\lambda)$

In order to choose the penalization parameter λ, we are going to write a function implementing the ridge regression penalization parameter λ choice based on the minimization of the $MSPE_{val}(\lambda)$.

```
ridge_lambda_search <- function(x, y, x.val, y.val, lambda.v) {
  n <- length(lambda.v)
  out <- data.frame(lambda=lambda.v, mspe=rep(0,n), df=rep(0,n))
  p <- ncol(x)
  # Get x Singular Value Decomposition
  x.svd <- svd(x)
  d <- x.svd$d
  v <- x.svd$v

  # Per lambda candidate compute coefficients, MSPE and df
  for(i in 1:n) {
    lambda <- lambda.v[i]
    # Compute (D^2 - lambda*Id)^-1
    d_inv <- diag(1/(d*d - lambda))
    # Compute (X^TX + lambda*Id)^-1
    xx_inv <- t( solve( t(x) %*% x + lambda*diag(1,p) ))
    # Compute beta
    beta <- xx_inv %*% t(x) %*% y
    # Compute y prediciton y.hat
    y.hat <- x.val %*% beta
    # Compute MSPE
    mspe <- sum((y.val - y.hat)^2) / length(y.val)
    # Compute df depending on the singular values
```

```
    df <- sum(d^2 / (d^2 +lambda))
    # Add results to output
    out$mspe[i] <- mspe
    out$df[i] <- df
  }

  # Plot mspe-log(lamba+1)
  plot(mspe~log(1+lambda), out, col=2)
  lambda.min <- out$lambda[which.min(out$mspe)]
  abline(v=log(1+lambda.min),col=2,lty=2)
  # Plot mspe-df
  plot(mspe~df, out, col=3)
  df.min <- out$df[which.min(out$mspe)]
  abline(v=df.min,col=3,lty=2)
  return(out)
}
```

Ridge regression based on $MSPE_{k-CV}(\lambda)$

Now, we will write an R function implementing the ridge regression penalization parameterλ choice based on k-fold cross-validation $MSPE_{kCV}(\lambda)$

```
ridge_lambda_search_cv <- function(x, y, lambda.v, cv=10) {
  n <- length(lambda.v)
  out <- data.frame(lambda=lambda.v, mspe=rep(0,n), df=rep(0,n))
  p <- ncol(x)

  # Per lambda candidate compute coefficients, MSPE and df
  for(i in 1:n) {
    lambda <- lambda.v[i]
    # Create folds for CV
    folds <- createFolds(1:nrow(x), k = cv)
    # Create fold out data.frame
    out.cv <- data.frame(mspe=rep(0,cv), df=rep(0,cv))
    for(j in 1:cv) {
      fold <- folds[[j]]
      # Get training and validation data for fold
      x.train <- x[-fold,]
      y.train <- y[-fold]
      x.val <- x[fold,]
      y.val <- y[fold]
      # Get x Singular Value Decomposition
      x.svd <- svd(x.train)
      d <- x.svd$d
      v <- x.svd$v
      # Compute (D^2 - lambda*Id)^-1
      d_inv <- diag(1/(d*d - lambda))
      # Compute (X^TX + lambda*Id)^-1
      xx_inv <- t( solve( t(x.train) %*% x.train + lambda*diag(1,p) ))
      # Compute beta
      beta <- xx_inv %*% t(x.train) %*% y.train
      # Compute y prediciton y.hat
      y.hat <- x.val %*% beta
      # Compute MSPE
```

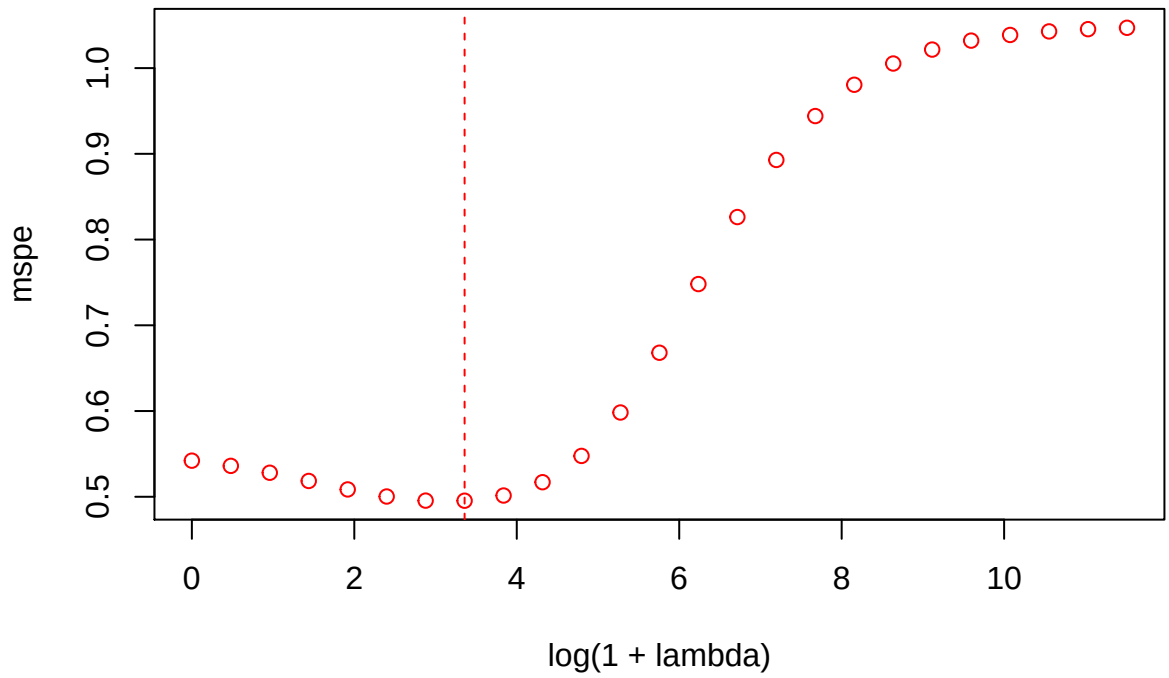
```
      mspe <- sum((y.val - y.hat)^2) / length(y.val)
      # Compute df depending on the singular values
      df <- sum(d^2 / (d^2 +lambda))
      # Add results to output
      out.cv$mspe[j] <- mspe
      out.cv$df[j] <- df
    }
    # Add mean of mspe/df to out data.frame
    out$mspe[i] <- mean(out.cv$mspe)
    out$df[i] <- mean(out.cv$df)

  }

  # Plot mspe-log(lamba+1)
  plot(mspe~log(1+lambda), out, col=2)
  lambda.min <- out$lambda[which.min(out$mspe)]
  abline(v=log(1+lambda.min),col=2,lty=2)
  # Plot mspe-df
  plot(mspe~df, out, col=3)
  df.min <- out$df[which.min(out$mspe)]
  abline(v=df.min,col=3,lty=2)

  return(out)

}
```

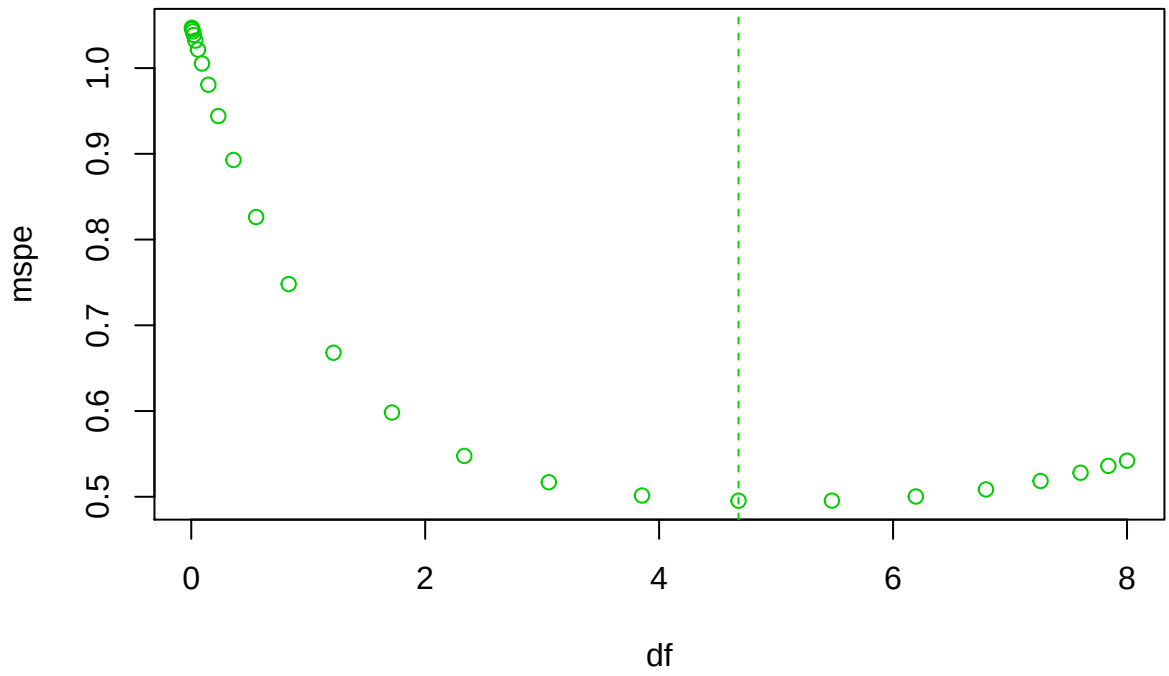
Comparation between penalization parameters of $MSPE_{val}(\lambda)$ and $MSPE_{k-CV}(\lambda)$

```
ridge_lambda_search_loocv_gcv <- function(x, y, lambda.v) {
  l <- length(lambda.v)
  out <- data.frame(lambda=lambda.v, loocv=rep(0,l), gcv=rep(0,l),df=rep(0,l))
  n <- nrow(x)
  p <- ncol(x)
  # Get x Singular Value Decomposition
  x.svd <- svd(x)
  d <- x.svd$d
  v <- x.svd$v
  u <- x.svd$u
  # Per lambda candidate compute coefficients, MSPE and df
  for(i in 1:l) {
    lambda <- lambda.v[i]
    # Compute (D^2 - lambda*Id)^-1
    d_inv <- diag(1/(d^2 - lambda))
    # Compute (X^TX + lambda*Id)^-1
    xx_inv <- t( solve( t(x) %*% x + lambda*diag(1,p) ))
    # Compute beta
    beta <- (xx_inv %*% t(x)) %*% y
    # Compute y prediciton y.hat
    y.hat <- x %*% beta
    # Compute df depending on the singular values
    df <- sum(d^2 / (d^2 +lambda))
    # Compute h such that y.hat = h * y
    h <- x %*% xx_inv %*% t(x)
```

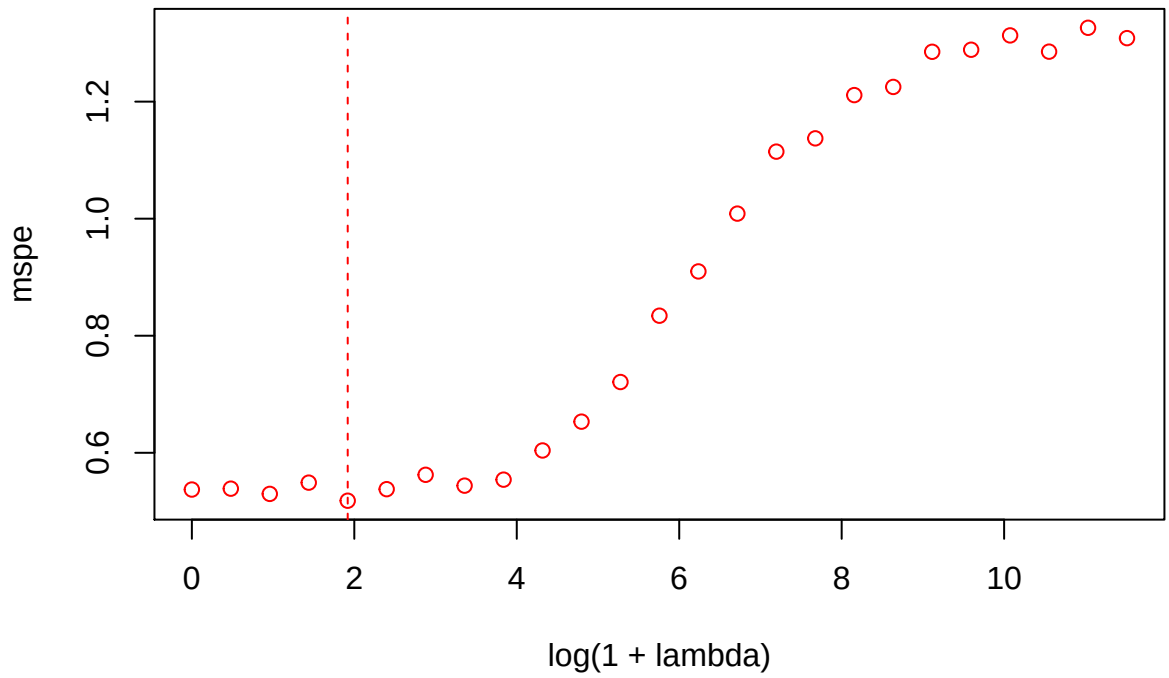
```
    # Add results to output
    out$loocv[i] <- sum( ( (y - y.hat)/(1 - diag(h)) )^2 ) / n
    out$gcv[i] <- sum( ( (y - y.hat)/(1 - df/n) )^2 ) / n
    out$df[i] <- df
  }

  # Plot loocv-log(lamba+1)
  plot(loocv~log(1+lambda), out, col=2)
  lambda.min <- out$lambda[which.min(out$loocv)]
  abline(v=log(1+lambda.min),col=2,lty=2)
  # Plot loocv-df
  plot(loocv~df, out, col=3)
  df.min <- out$df[which.min(out$loocv)]
  abline(v=df.min,col=3,lty=2)

  # Plot loocv-log(lamba+1)
  plot(gcv~log(1+lambda), out, col=2)
  lambda.min <- out$lambda[which.min(out$gcv)]
  abline(v=log(1+lambda.min),col=2,lty=2)
  # Plot loocv-df
  plot(gcv~df, out, col=3)
  df.min <- out$df[which.min(out$gcv)]
  abline(v=df.min,col=3,lty=2)
  return(out)
}
```

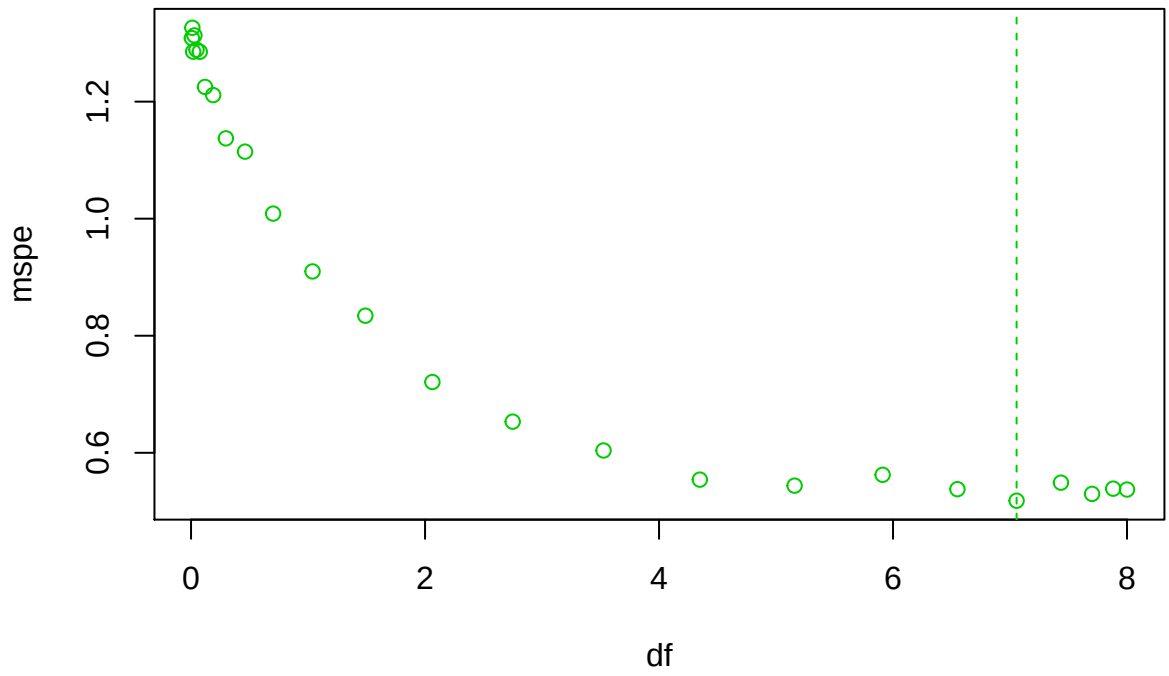
```
# Data load and pre-process
prostate <- read.table("prostate_data.txt", header=TRUE, row.names = 1)
data <- prostate %>%
  dplyr::select(-train)
# No gold rule for choosing this
lambda.max <- 1e5
n.lambdas <- 25
# Evenly spaced in scale of logarithm
lambda.v <- exp(seq(0,log(lambda.max+1),length=n.lambdas))-1
# With Validation data
validation.ind <- prostate$train
validation <- data[!validation.ind,]
training <- data[validation.ind,]
x <- training %>% select(-lpsa) %>% scale(center = T, scale = T)
y <- training %>% select(lpsa) %>% scale(center = T, scale = F)
x.val <- validation %>% select(-lpsa) %>% scale(center = T, scale = T)
y.val <- validation %>% select(lpsa) %>% scale(center = T, scale = F)
out.valid <- ridge_lambda_search(x, y, x.val, y.val, lambda.v)
```


mspe
0.5
0.6
0.7
0.8
0.9
1.0
0
2
4
6
8
10
log(1 + lambda)

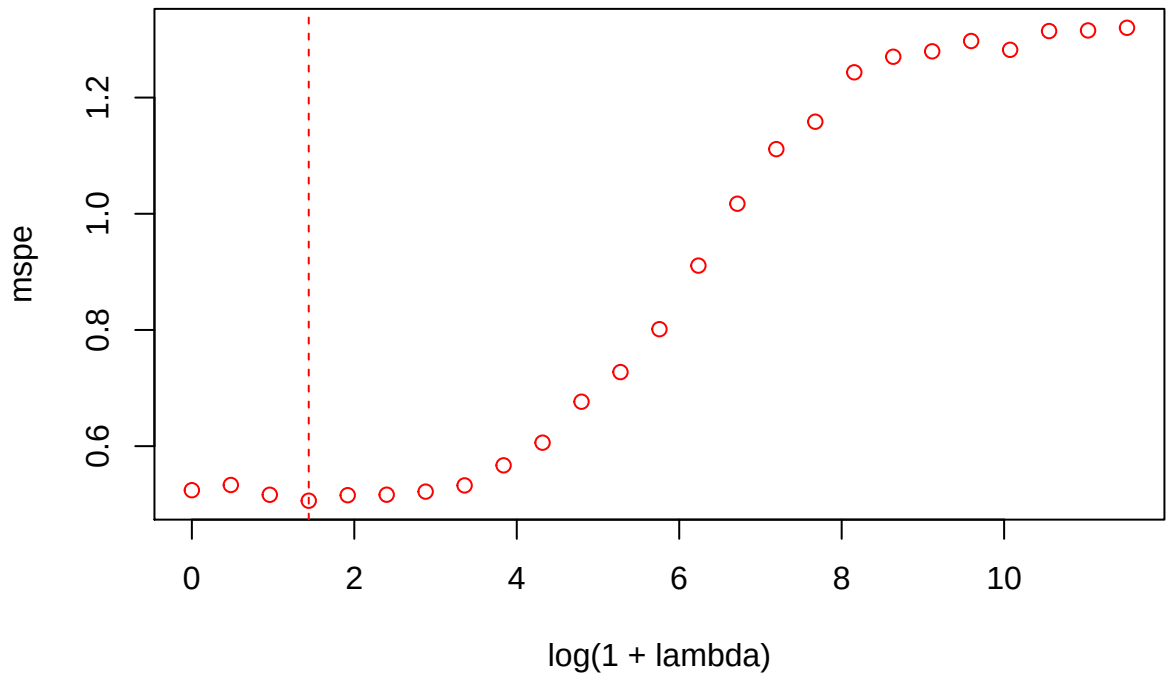


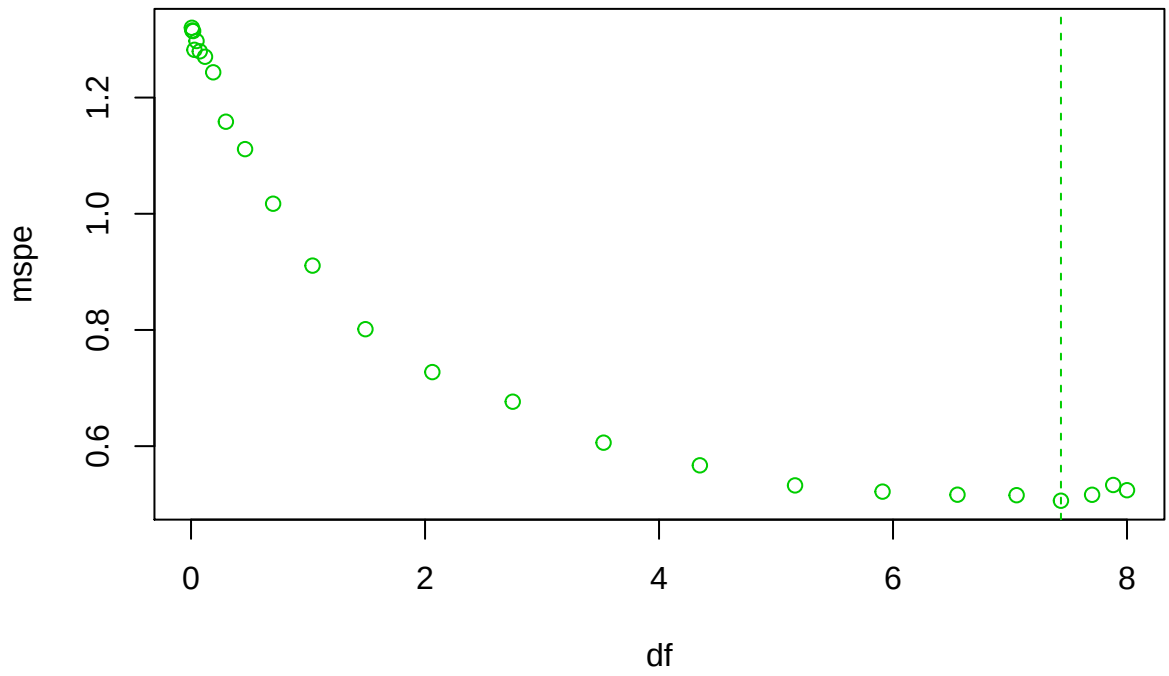

```
# 5-fold and 10-fold CV
# With CV
x <- data %>% select(-lpsa) %>% scale(center = T, scale = T)
y <- data %>% select(lpsa) %>% scale(center = T, scale = F)
out.5.cv <- ridge_lambda_search_cv(x, y, cv = 10, lambda.v)
```


mspe
0.6
0.8
1.0
1.2
0
2
4
6
8
10
log(1 + lambda)

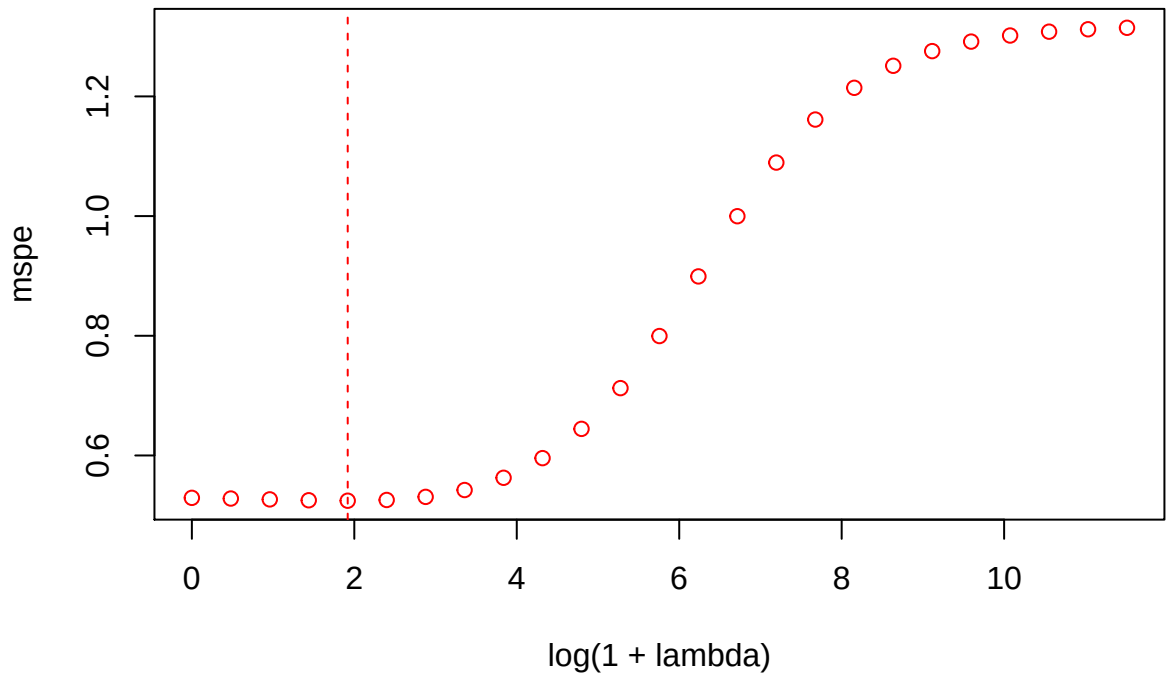


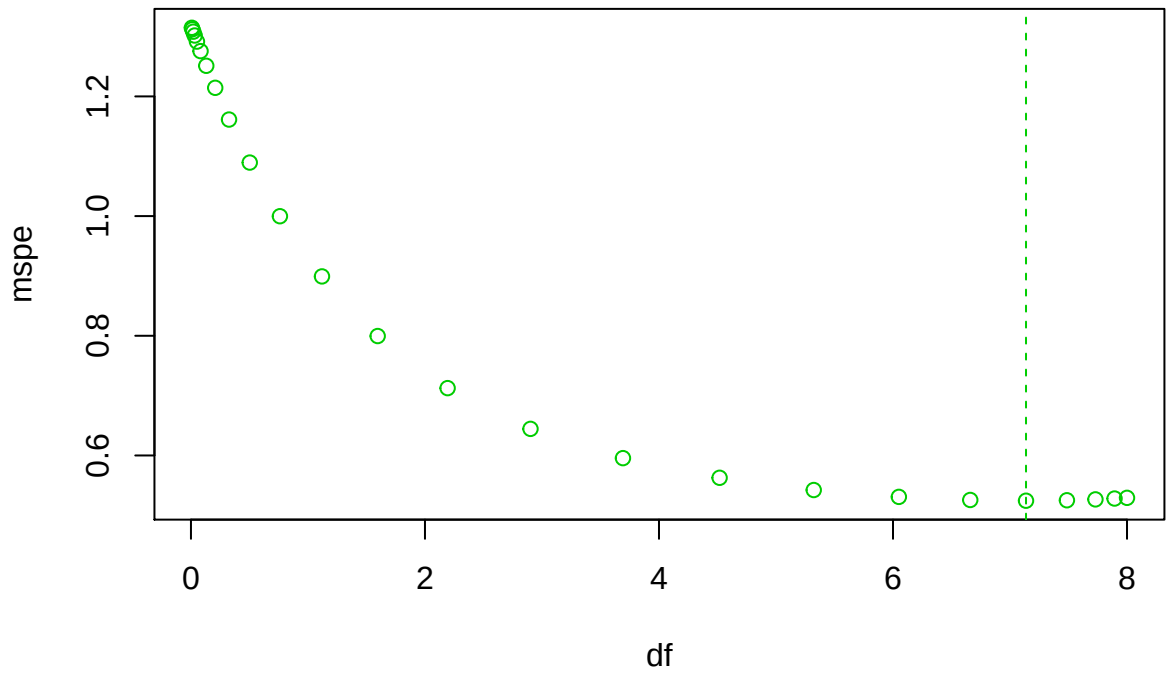

```
out.10.cv <- ridge_lambda_search_cv(x, y, cv = 10, lambda.v)
```


mspe
0.6
0.8
1.0
1.2
0
2
4
6
8
10
log(1 + lambda)

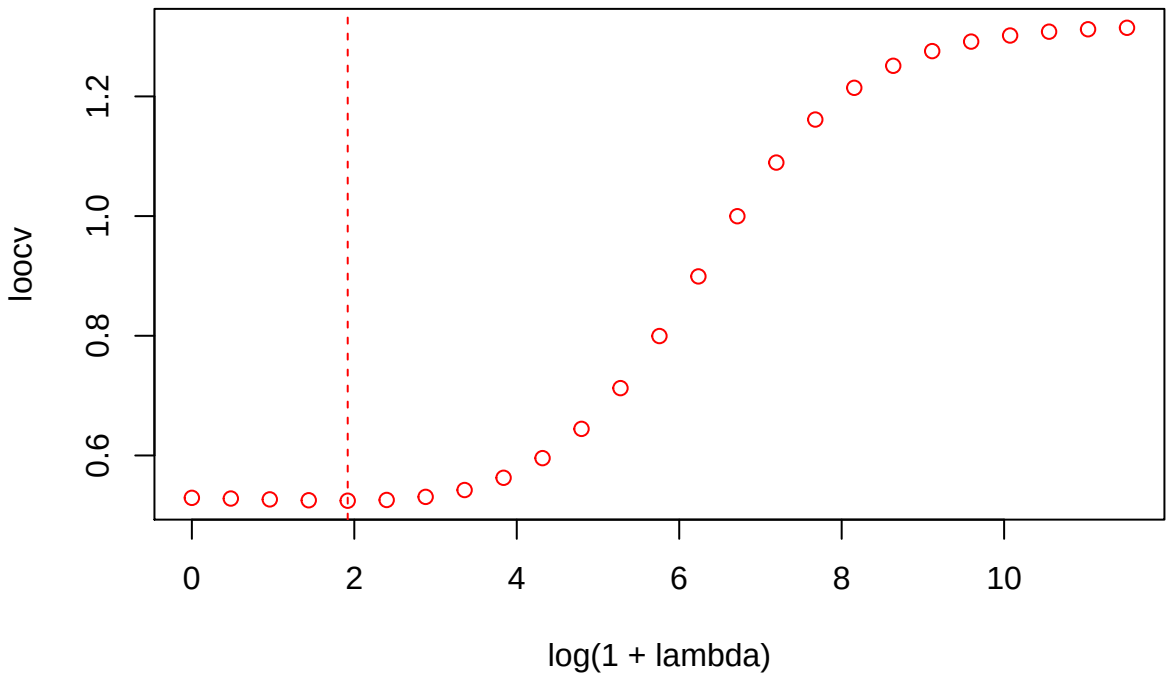


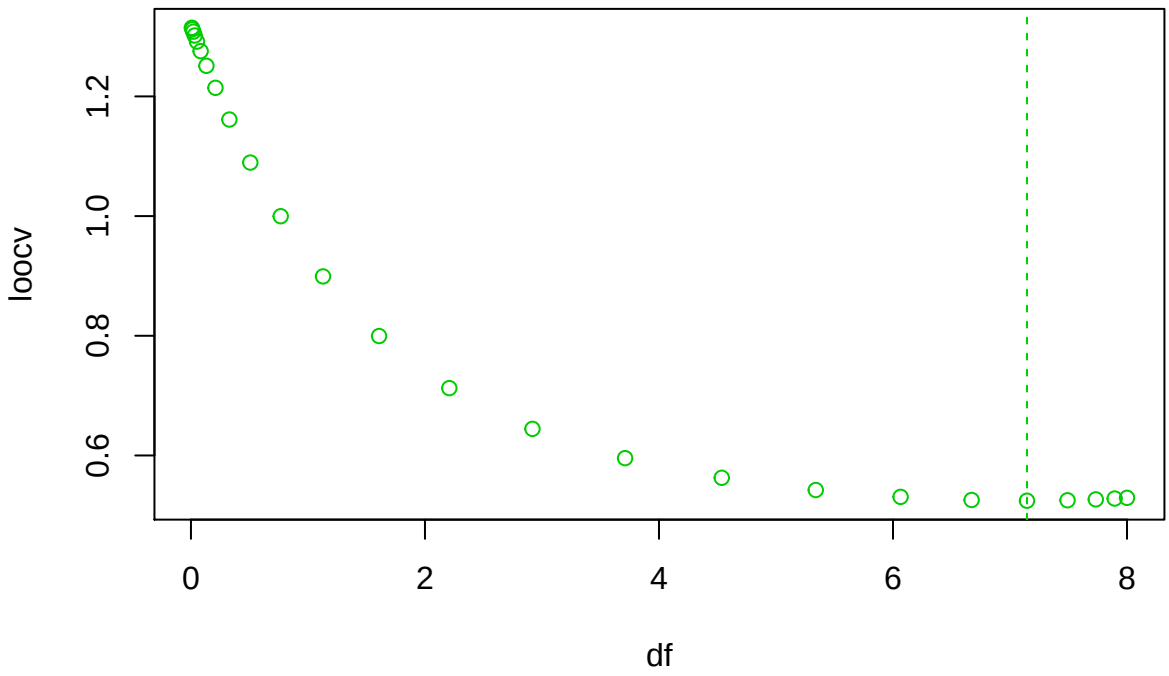

```
# LOOCV and GCV estimates
out.loocv <- ridge_lambda_search_cv(x, y, cv = nrow(x), lambda.v)
```

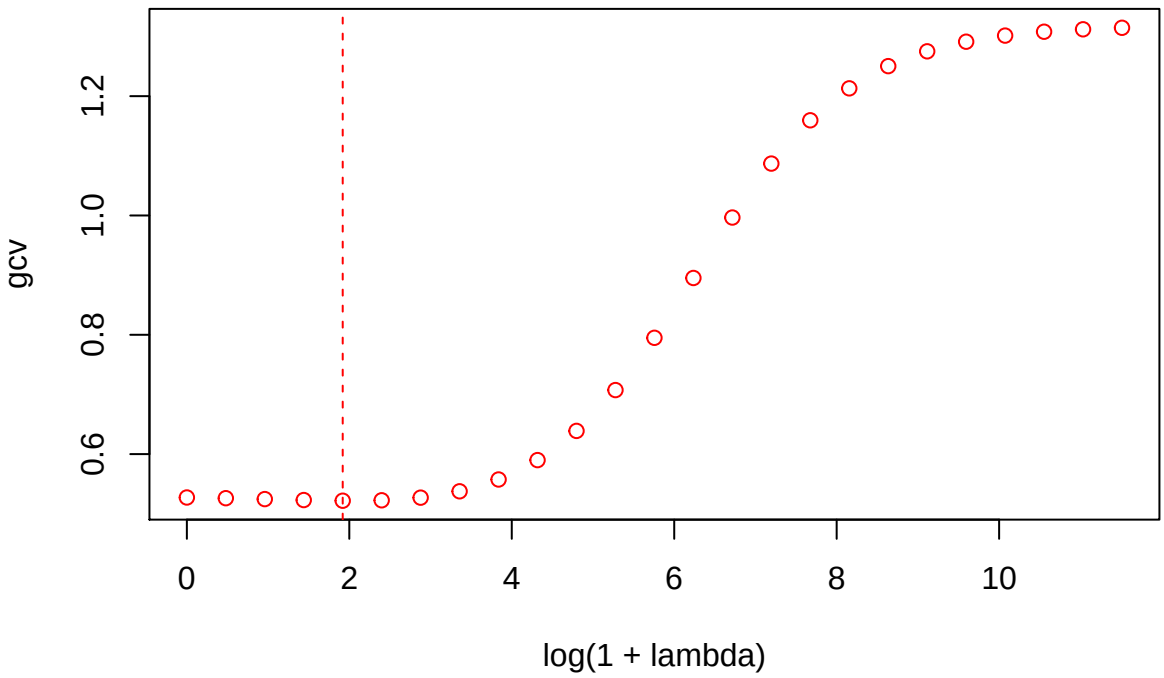

mspe
0.6
0.8
1.0
1.2
0
2
4
6
8
10
log(1 + lambda)

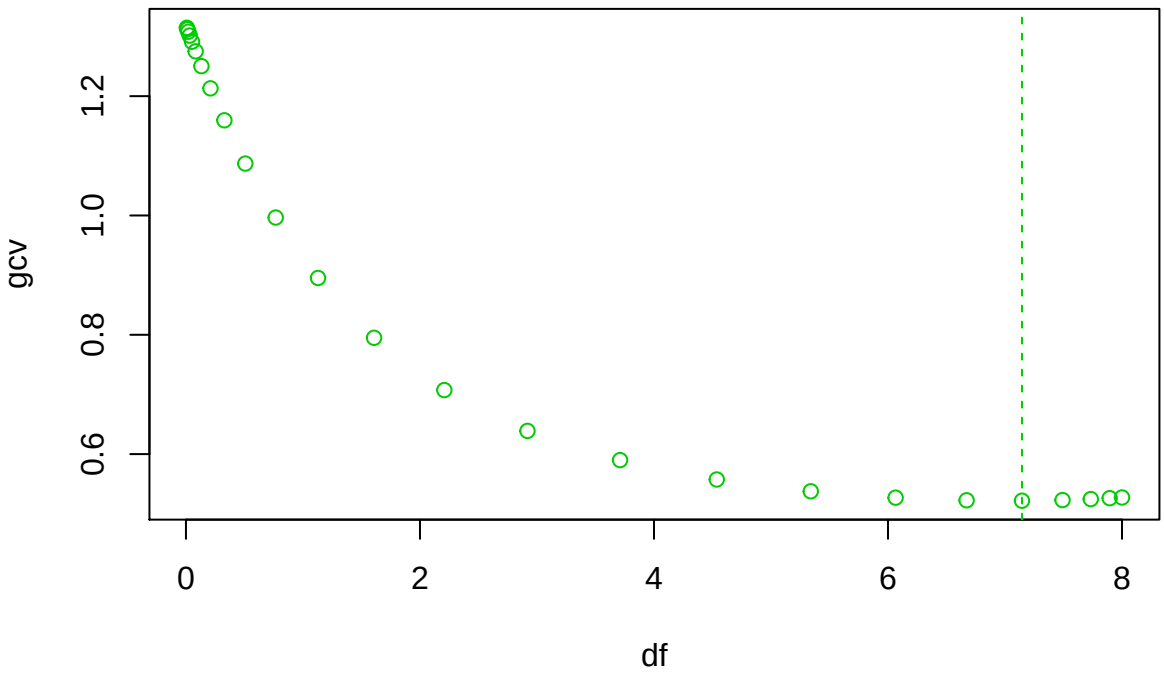



```
out.loocv.gcv <- ridge_lambda_search_loocv_gcv(x, y, lambda.v)
```


loocv
0.6
0.8
1.0
1.2
0
2
4
6
8
10
log(1 + lambda)


loocv
0.6
0.8
1.0
1.2
0
2
4
6
8
df


gcv
0.6
0.8
1.0
1.2
0
2
4
6
8
10
log(1 + lambda)

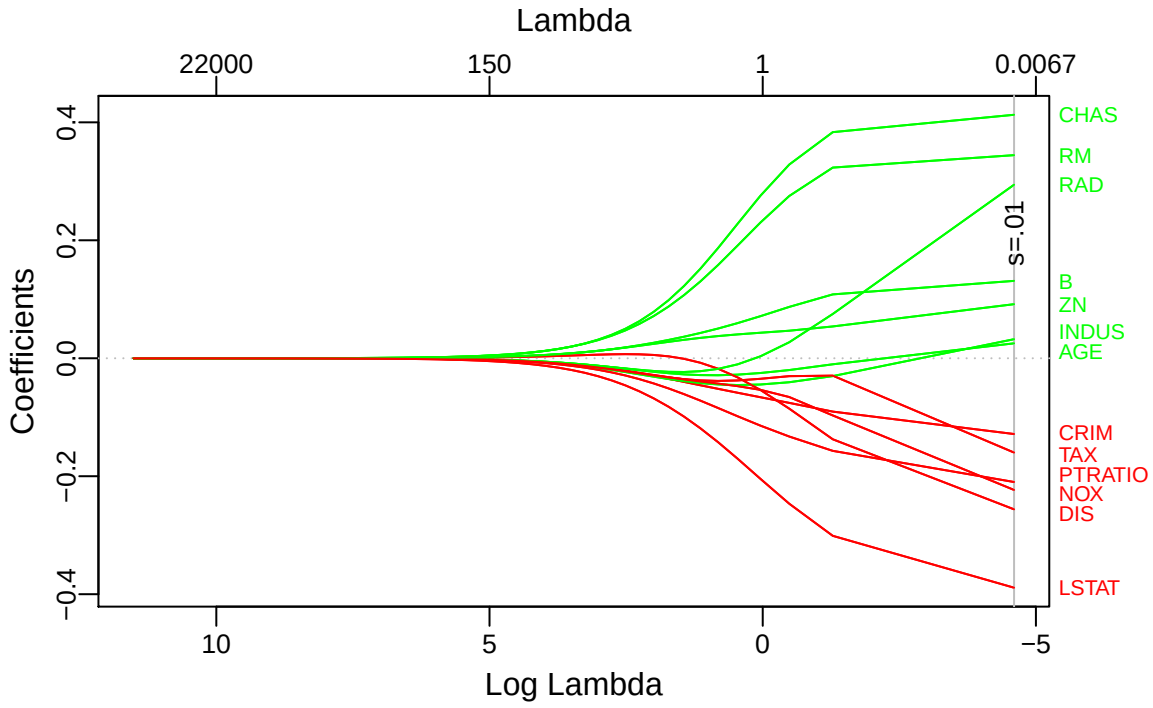


We observe that the results of MSPE seem more stable in the LOOCV and GCV compared to 5-fold and 10-fold CV, as one would expect given the small size of the dataset. Furthermore, the validation set results are also relatively stable, although with a different curve than for the LOOCV/GCV.

Ridge regression for the Boston Housing data

We start by scaling and splitting the Boston dataset to training and test using a 2/3 ratio. Since *CHAR* is a factor variable we do not include it in the *scale* function. First we need to tune the parameter λ. To do this we use 10 fold cross validation performed by *cv.glmnet*.

```
## Warning: package 'plotmo' was built under R version 3.5.3

## Loading required package: Formula

## Loading required package: plotrix

## Warning: package 'plotrix' was built under R version 3.5.3

## Loading required package: TeachingDemos

## Warning: package 'TeachingDemos' was built under R version 3.5.3
```



To select the best model, we now use 10x10-CV using the lambda that best minimised the error in cross-validation, which is 0.0100502.

So our final model has $Df=$ 13 which is the number of non-zero coefficients and $\%Dev=$ 0.7565166 is the percent deviance explained, which is quite good.

In terms of interpreting the coefficients, we observe that each additional room (RM) is associated with an increase in the house price, on average. This is quite straightforward, in principle, since it is to be expected that the larger the house, loosely speaking, the more expensive it will be. In addition, we see that an increase in RAD (index of accessibility to radial highways) is associated with an increse in $MEDV$. So basically, if we were to think of the town as a graph we would be capturing the connectivity degree of a specific suburb; so a remote node would have a lower value. Moreover, an increase in $CHAS$ would mean that it will take the value of 1 is associate with an increase in $MEDV$, so ultimately if the Charles River passes through this suburb then this signals a higher house price, on average.

On the other hand, an increase in $LSTAT$ (% lower status of the population) is associated with a decrease in the house price, on average. Most interestinglty though is that the increase in $PTRATIO$ (pupil-teacher ratio by town) is associated with a decrease in the house price, on average. So in other words, the education offering of a town increases its value. Also, an increase in DIS (weighted distances to five Boston employment centres) is associated with a decrease in $MEDV$, on average. So, having to do a larger commute to work signals a lower house price. Another reasonable result is the fact that an increase in NOX (nitric oxides concentration) is associated with a decrease in $MEDV$, so air pollution is a detractor to house price.

Furthermore, we see that neither AGE (proportion of owner-occupied units built prior to 1940) nor $INDUS$ (roportion of non-retail business acres per town) seem to have a considerable effect on $MEDV$.

Finally, we obtain the train and test error.

Table 1: Model Errors Summary

regression_method	train_MSE	test_MSE
ridge regression	0.249	0.305

The difference between train and test errors is not that large, even if the test set is relatively small, and thus subject to a great variance.