Deliverable 2. Choosing the tuning parameter in ridge regression

Daniel Fuentes, Marc Valenti

20 de febrero de 2018

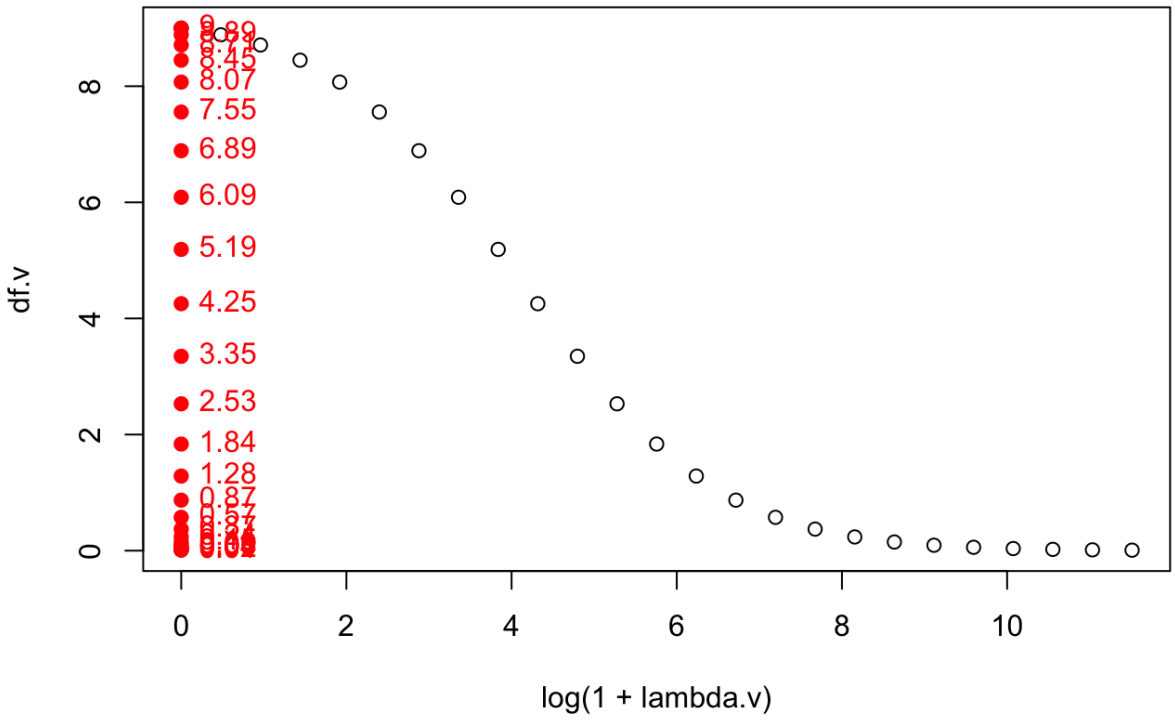
Write a report that contains the results of the computations that you are asked to carry out below, as well as the explanation of what you are doing. The main text (2 or 3 pages) should include pieces of source code and graphical and nu- merical output. Upload your answers in a .pdf document (use LaTeX or R Markdown, for ins- tance) to ATENEA, as well as the source code (*.R or* .Rmd, for instance). Your work must be reproducible.

```
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL, title="") {
  require(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

  if (nchar(title)>0){
    layout<-rbind(rep(0, ncol(layout)), layout)
  }

  if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout), heights =if(nchar(title)>0){

    # Make each plot, in the correct location
    if (nchar(title)>0){
      grid.text(title, vp = viewport(layout.pos.row = 1, layout.pos.col = 1:ncol(layout)))
    }

    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
```

```
                                        layout.pos.col = matchidx$col))
    }
  }
}

get_plot_cv<-function(PMSE,title){
  par(mfrow=c(1,2))
  aux<-cbind(PMSE,df.v,lambda.v) %>%
    as.data.frame()
  aux_min<-aux %>%
    slice(which.min(PMSE))

  p1<-ggplot() +
    geom_point(data=aux,
               aes(y=PMSE.mean,
               x=log(1+lambda.v))) +
    geom_errorbar(data=aux,
                  aes(x=log(1+lambda.v),
                      ymin=PMSE.mean - PMSE.sd,
                      ymax=PMSE.mean + PMSE.sd)) +
    geom_vline(xintercept = log(1+aux_min$lambda.v),
               linetype='dashed',
               color='red') +
    labs(x=expression(paste('log (1+',
                  lambda,
                  ')')))

  p2<-ggplot() +
    geom_point(data=aux,
               aes(y=PMSE.mean,
               x=df.v)) +
    geom_errorbar(data=aux,
                  aes(x=df.v,
                      ymin=PMSE.mean - PMSE.sd,
                      ymax=PMSE.mean + PMSE.sd)) +
    geom_vline(xintercept = aux_min$df.v,
                linetype='dashed',
                color='red') +
    labs(x=expression(paste('df(',
                  lambda,
                  ')')))
  multiplot(p1,p2,cols=2,title=title)

}
```

```
prostate <- read.table("prostate_data.txt", header=TRUE, row.names = 1)

################################Training set################################
train<- which(prostate$train==TRUE) #Seleccionem mostra d'entrenament
Ytn <- scale( prostate$lpsa[train], center=TRUE, scale=FALSE) #Centered matrix, mean removal of Y
Xtn <- scale( as.matrix(prostate[train,1:8]), center=TRUE, scale=TRUE) #Standardized matrix of X
ntn <- dim(Xtn)[1] #N?mero files
ptn <- dim(Xtn)[2] #N?mero columnes
```

```
XtXtn <- t(Xtn)%*%Xtn
d2tn <- eigen(XtXtn,symmetric = TRUE, only.values = TRUE)$values #Autovalors XtX

################################Validation set##############################
validation<- which(prostate$train==FALSE) #Seleccionem mostra d'entrenament
Yvl <- scale( prostate$lpsa[validation], center=TRUE, scale=FALSE) #Centered matrix, mean removal of Y
Xvl <- scale( as.matrix(prostate[validation,1:8]), center=TRUE, scale=TRUE) #Standardized matrix of X
nvl <- dim(Xvl)[1] #N?mero files
pvl <- dim(Xvl)[2] #N?mero columnes

XtXvl <- t(Xvl)%*%Xvl
d2vl <- eigen(XtXvl,symmetric = TRUE, only.values = TRUE)$values #Autovalors XtX


#full#
Y <- scale( prostate$lpsa, center=TRUE, scale=FALSE) #Centered matrix, mean removal of Y
X <- scale( as.matrix(prostate[,!names(prostate)%in%'lpsa']), center=TRUE, scale=TRUE) #Standardized ma
n <- dim(X)[1] #N?mero files
p <- dim(X)[2] #N?mero columnes

XtX <- t(X)%*%X
d2 <- eigen(XtX,symmetric = TRUE, only.values = TRUE)$values #Autovalors XtX

##############################Lambdes#########################
lambda.max <- 1e5 #Fixem valor m?xim lambda. Lambda infinit = 0 graus llibertat/par?metres efectius, Be
n.lambdas <- 25
lambda.v <- exp(seq(0,log(lambda.max+1),length=n.lambdas))-1 #Generem vector amb 25 valors de lambda de

############
# effective degrees of freedom
#############
df.v <- numeric(n.lambdas)
for (l in 1:n.lambdas){
  lambda <- lambda.v[l]
  df.v[l] <- sum(d2/(d2+lambda))
}
plot(log(1+lambda.v),df.v)
points(0*df.v,df.v,col=2,pch=19)
text(0*df.v,df.v,round(df.v,2),col=2,pch=19,pos=4)
```

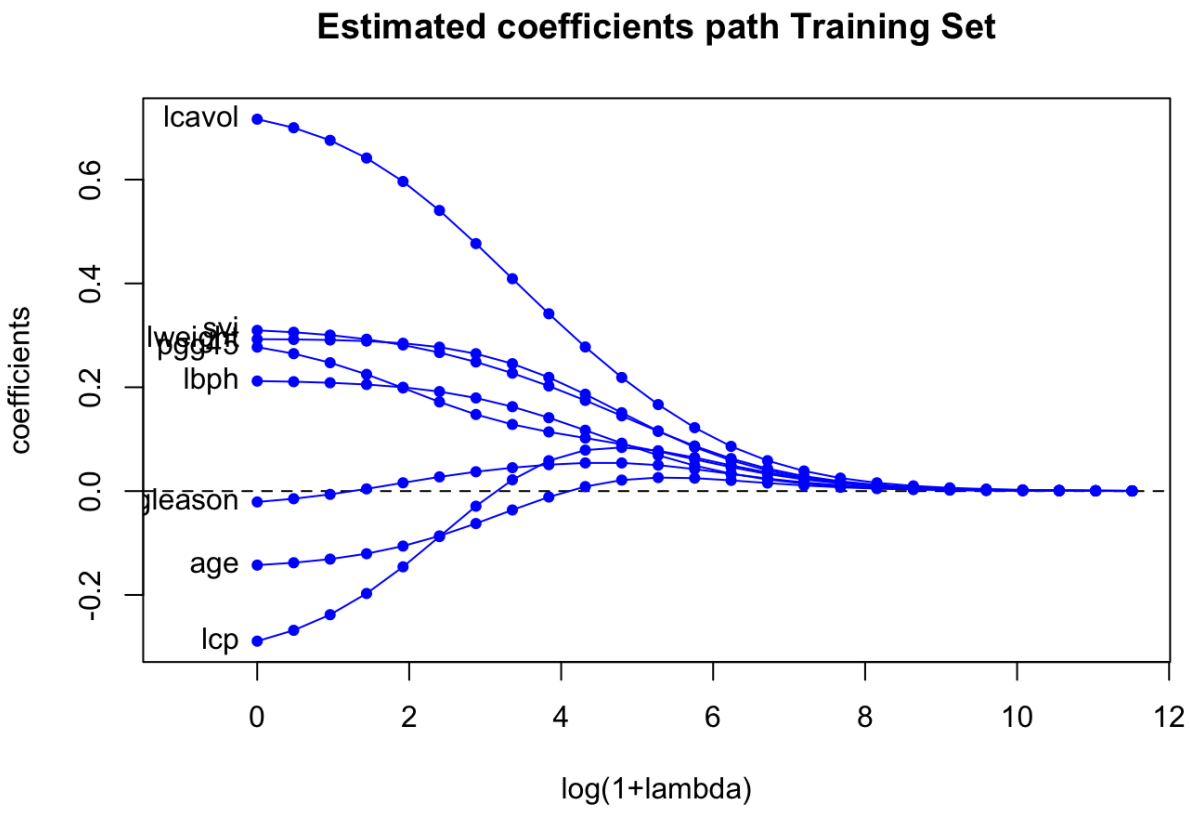


1. Choosing the penalization parameter

1. Write an R function implementing the ridge regression penalization parameter choice based on the minimization of the mean squared prediction error in a validation set (MSPE_{val()}). Input: Matrix x and vector y corresponding to the training sample; matrix x_val and vector y_val corresponding to the validation set; a vector lambda.v of candidate values for . Output: For each element in lambda.v, the value of (MSPE_{val()}). Additionally you can plot these values against log(1 +) 1 in the same graphic, or against df().

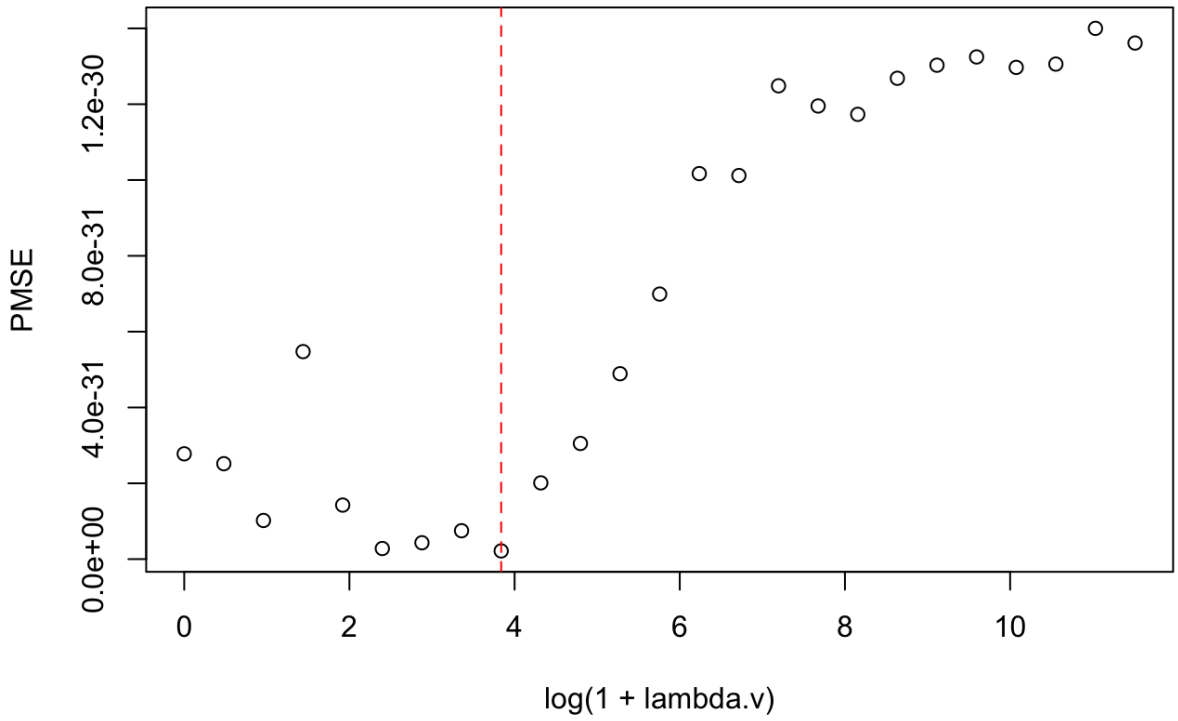
```
#############
# estimated coefficients path Training set
#############
beta.pathtn <- matrix(0,nrow=n.lambdas, ncol=ptn)
diag.H.lambdatn <- matrix(0,nrow=n.lambdas, ncol=ntn)
for (l in 1:n.lambdas){
  lambda <- lambda.v[l]
  H.lambda.auxtn <- t(solve(XtXtn + lambda*diag(1,ptn))) %*% t(Xtn)
  beta.pathtn[l,] <-  H.lambda.auxtn %*% Ytn
  H.lambdatn <- Xtn %*% H.lambda.auxtn
  diag.H.lambdatn[l,] <- diag(H.lambdatn)
}
plot(c(-1,log(1+lambda.v[n.lambdas])), range(beta.pathtn),
     type="n",
     xlab="log(1+lambda)",
```

```
     ylab="coefficients",
     main="Estimated coefficients path Training Set")
abline(h=0,lty=2)
for(j in 1:ptn){
  lines(log(1+lambda.v),beta.pathtn[,j],col=4)
  points(log(1+lambda.v),beta.pathtn[,j],pch=19,cex=.7,col=4)
}
text(0*(1:ptn), beta.pathtn[1,],names(prostate)[1:ptn],pos=2)
```




```
#############
# choosing lambda by PMSE
#############
PMSE<-numeric(n.lambdas)
m.Y<-0

for (l in 1:n.lambdas){
  #per cada lambda
  lambda <- lambda.v[l]
  #fixem PMSE=0
  PMSE[l] <- 0
  #we train on the train set
  beta <- solve(t(Xtn)%*%Xtn + lambda*diag(1,ptn)) %*% t(Xtn) %*% Ytn
  #predictions on validation
  hat.Y <- Xvl %*% beta + m.Y
  #obtain metrics (on validation)
```

```
  PMSE[l] <- (PMSE[l] + sum(hat.Y-Yvl)^2)/nvl
}

lambda <- lambda.v[which.min(PMSE)]
plot(log(1+lambda.v), PMSE)
abline(v=log(1+lambda),col=2,lty=2)
```



El gràfic "Estimated coefficients path Training Set", mostra el valor del coeficient de cadascuna de les variables en funció dels 25 valors del vector lambda.v (en aquest cas expressat com log(1+lambda)).

El segon gràfic mostra en l'eix y el valor de PMSE (suma dels errors al quadrat en el conjunt de validació) vs l'eix x on trobem el valor del log(1+lambda) per cadascuna de les 25 lambdes contingudes al vector lambda.v. Així doncs a través d'aquesta representació podem conèixer quina és l'estimació de lambda optima que minimitza PMSE gràcies a la regressió Ridge. El lambda òptim és 45.41604.

2. Write an R function implementing the ridge regression penalization para-

meter choice based on k-fold cross-validation (MSPE_{kCV}()). Input, output and graphics as before.

```
set.seed(1234)
get_k_cv_metrics<-function(X,Y,lambda.v,k){
  n <- dim(X)[1] #N?mero files
  p <- dim(X)[2]
```

```
  XtX <- t(X)%*%X
  d2 <- eigen(XtX,symmetric = TRUE, only.values = TRUE)$values #Autovalors XtX
  n.lambdas<-length(lambda.v)

  PMSE.kCV <- matrix(-1,
                     ncol=n.lambdas,
                     nrow=k)

  #create random numbers from 1 to k
random_idx<-sample(1:k,n,replace=TRUE)
#if we are interested in the leave-one-out validation,
#where k = n (the number of observations)
#then we have to make sure that the index is not a random one
if(k==n){
  random_idx<-seq(from=1,
                  to=k,
                  by=1)
  print("Warning! Doint Leave-One-Out CV")
  name<-"LOO-CV"
}
#create vector of length of results
for (l in 1:n.lambdas){
  #work with the indexed lambda
  lambda <- lambda.v[l]
  l<-l
  for (i in 1:k){
    m.Y.i <- 0
    #fold that used as a trainingset
    X.i <- X[!random_idx==i,]
    Y.i <- Y[!random_idx==i]-m.Y.i
    #fold that used as a validationset (those which k equals their index)
    Xi <- X[random_idx==i,]
    Yi <- Y[random_idx==i]
    beta.i <- solve(t(X.i)%*%X.i + lambda*diag(1,p), tol = 1e-20) %*% t(X.i) %*% Y.i
    hat.Yi <- Xi %*% beta.i + m.Y.i
    #sum of the squared residuals
    PMSE.kCV[i,l] <- (sum(hat.Yi-Yi)^2)
  }
}
PMSE.kCV<-PMSE.kCV/n
PMSE.mean<-apply(PMSE.kCV,2,mean) #mean of the columns
PMSE.sd<-apply(PMSE.kCV,2,sd) #sd of the columns
return(cbind(PMSE.mean,PMSE.sd))
}
```

3. Consider the prostate date used in class. Use your routines to choose the penalization parameter by the following criteria: behavior in the validation set (the 30 observations not being in the traineng sample); 5-fold and 10-fold cross-validation. Compare your results with those obtained when using leave-one-out and generalized cross-validation.

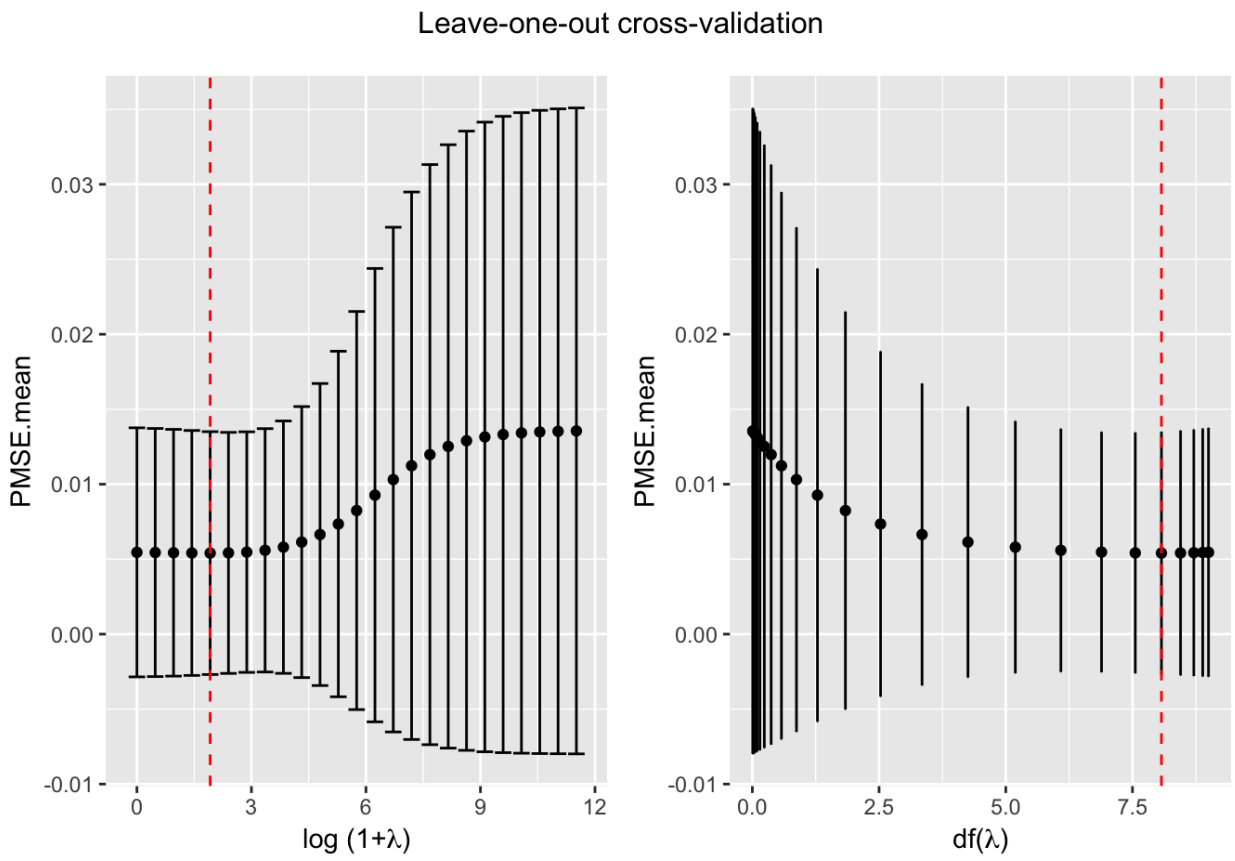
```
prostate <- read.table("prostate_data.txt", header=TRUE, row.names = 1)

################################Training set#################################
Y <- scale( prostate$lpsa, center=TRUE, scale=FALSE) #Centered matrix, mean removal of Y
```

```
X <- scale( as.matrix(prostate[,1:8]), center=TRUE, scale=TRUE) #Standardized matrix of X


n <- dim(X)[1] #N?mero files
p <- dim(X)[2] #N?mero columnes

XtX <- t(X)%*%X
d2 <- eigen(XtX,symmetric = TRUE, only.values = TRUE)$values #Autovalors XtX


PMSE.LOOCV<-get_k_cv_metrics(X,Y,lambda.v,k=nrow(X))
```

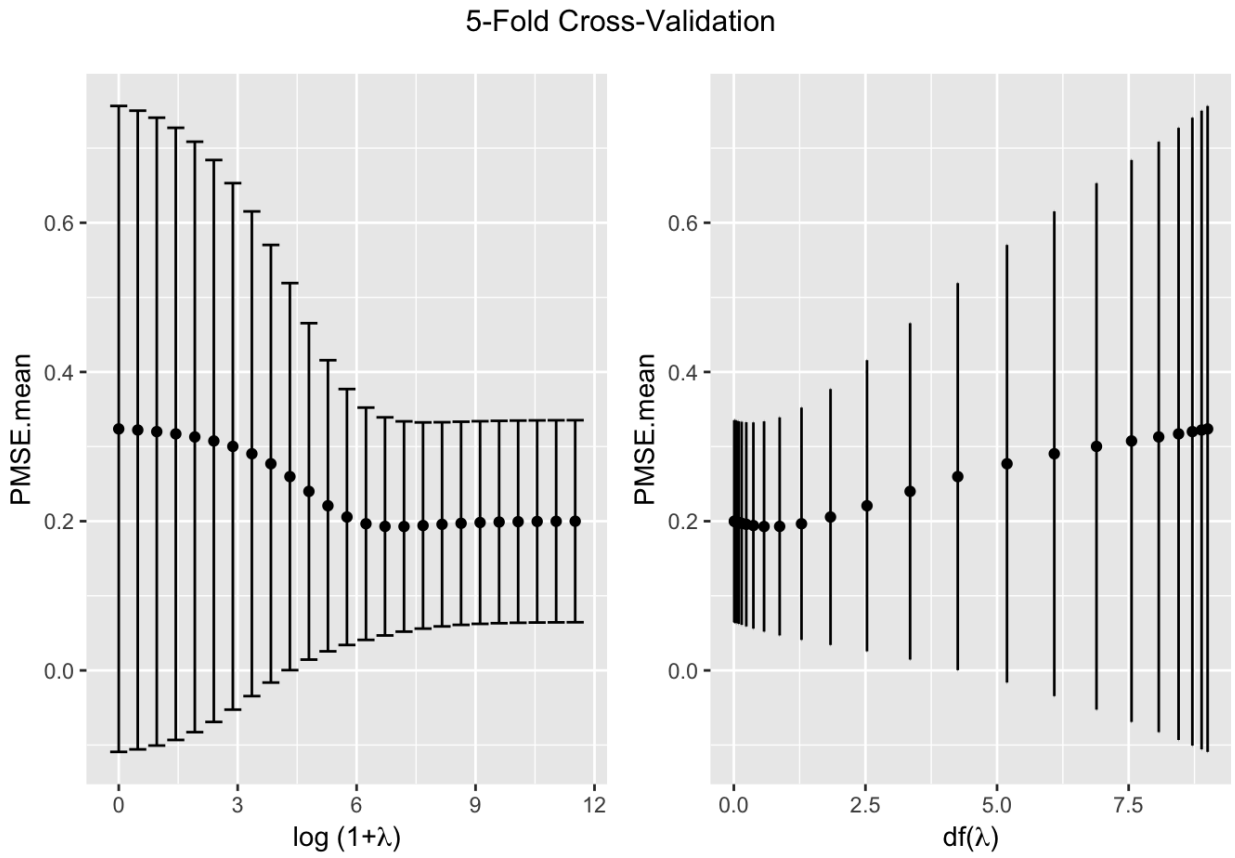
```
## [1] "Warning! Doint Leave-One-Out CV"
```

```
PMSE.5CV<-get_k_cv_metrics(X,Y,lambda.v,k=5)
PMSE.10CV<-get_k_cv_metrics(X,Y,lambda.v,k=10)


get_plot_cv(PMSE.LOOCV,title='Leave-one-out cross-validation')
```

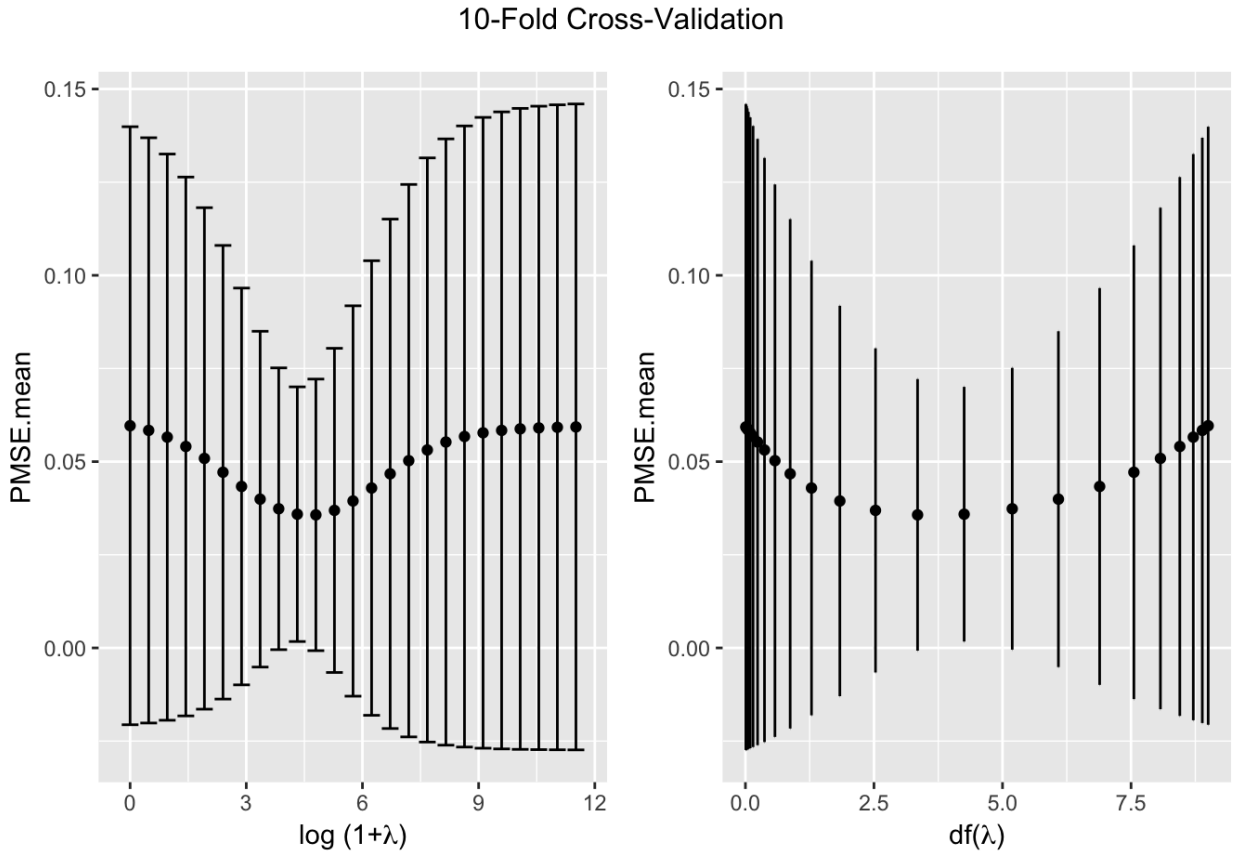
```
## Loading required package: grid
```




```
get_plot_cv(PMSE.5CV,title='5-Fold Cross-Validation')
```

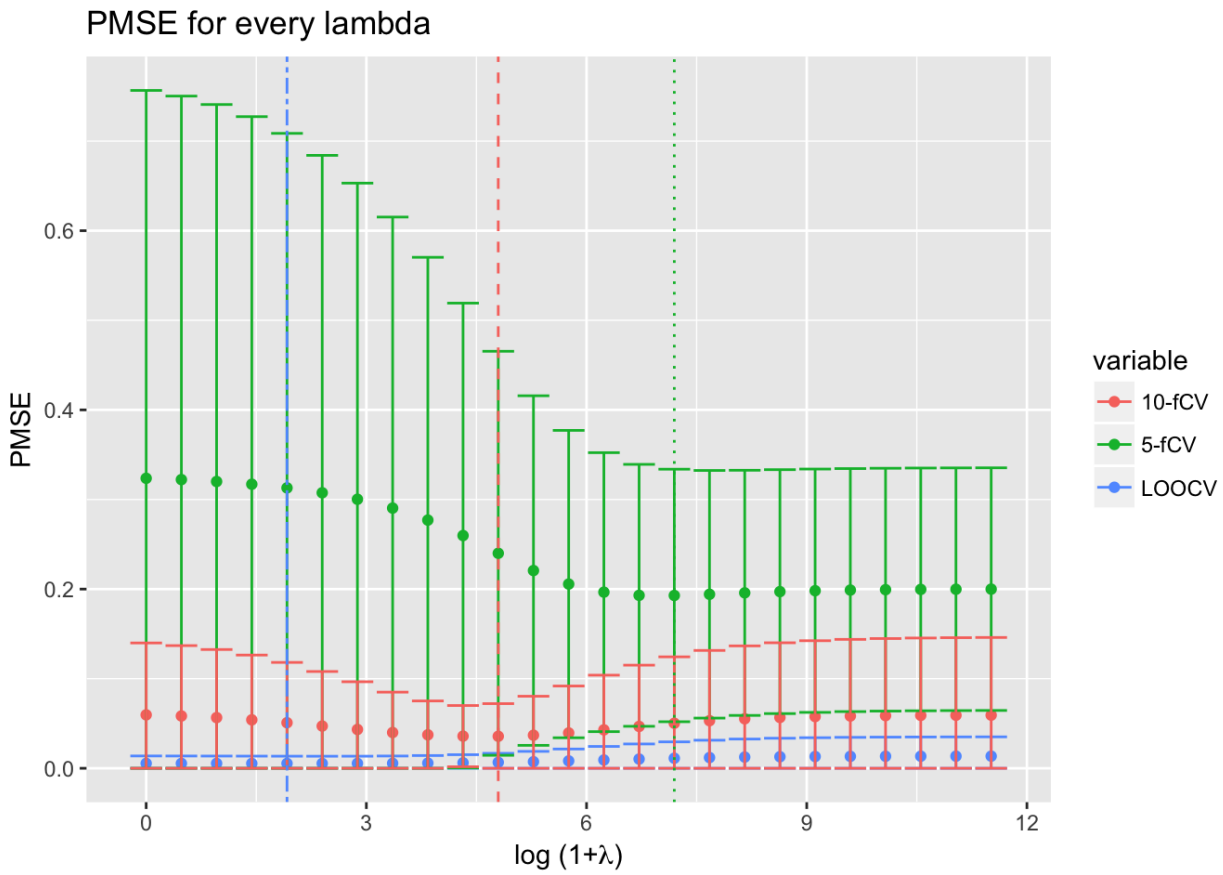



```
get_plot_cv(PMSE.10CV,title='10-Fold Cross-Validation')
```




```
dt_final_plot<-as.data.frame(rbind(PMSE.LOOCV,PMSE.5CV,PMSE.10CV))
dt_final_plot$variable<-c(rep("LOOCV",25),
                          rep("5-fCV",25),
                          rep("10-fCV",25))
dt_final_plot$lambda.v<-as.vector(lambda.v)

dt_final_plot_min<-dt_final_plot %>%
  group_by(variable) %>%
  slice(which.min(PMSE.mean)) %>%
  mutate('log(1+lambda)'=log(1+lambda.v))
dt_final_plot_min$lt<-c('dotted','dashed','twodash')


ggplot() +
  geom_point(data=dt_final_plot,
             aes(x=log(1+lambda.v),
                 y=PMSE.mean,
                 col=variable)) +
  geom_errorbar(data=dt_final_plot,
             aes(x=log(1+lambda.v),
                 ymin=ifelse(PMSE.mean-PMSE.sd<0,0,PMSE.mean-PMSE.sd),
                 ymax=PMSE.mean+PMSE.sd,
                 col=variable)) +
  geom_vline(data=dt_final_plot_min,
              aes(xintercept=dt_final_plot_min$`log(1+lambda)`,
```

```
                col=variable,
                linetype=lt),
            show.legend = FALSE) +
  scale_linetype_manual(values=c('dotted','dashed','twodash')) +
  labs(title='PMSE for every lambda',
       x=expression(paste('log (1+',
                  lambda,
                  ')')),
       y='PMSE')
```




```
dt_final_plot_min[,c(3,1,2,4,5)] %>%
  set_colnames(c("CV_method","PMSE","PMSE_sd","lambda","log(1+lambda)")) %>%
  kable()
```

CV_method	PMSE	PMSE_sd	lambda	log(1+lambda)
10-fCV	0.0357129	0.0364424	120.153271	4.797056
5-fCV	0.1928882	0.1409157	1332.529767	7.195585
LOOCV	0.0054062	0.0080987	5.812932	1.918823

As said in The Elements of Statistical Learning (Section 7.10.1): With K=N, the cross-validation estimator is approximately unbiased for the true (expected) prediction error, but can have high variance because the N "training sets" are so similar to one another.

In this case, however, LOO presents the lower PMSE and PMSE variance aswell.

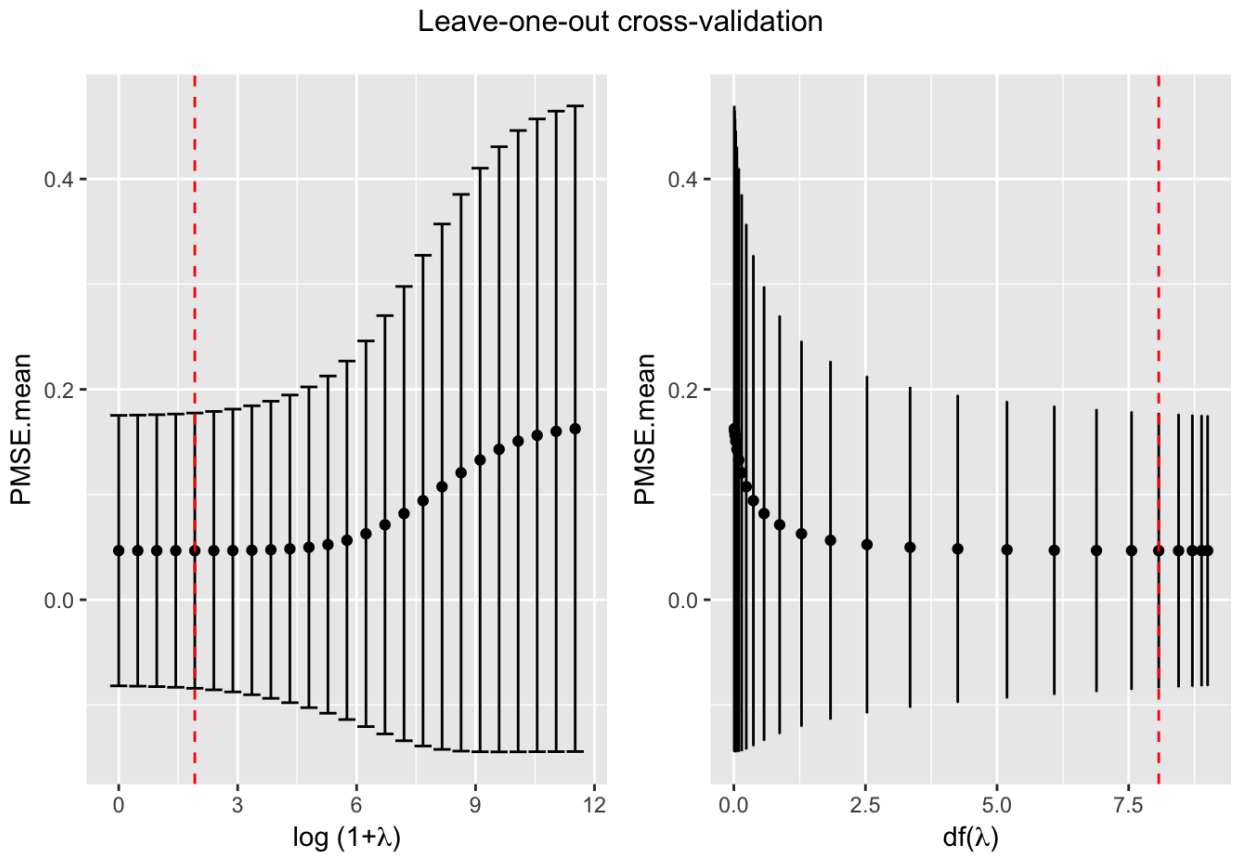
2. Ridge regression for the Boston Housing data

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.

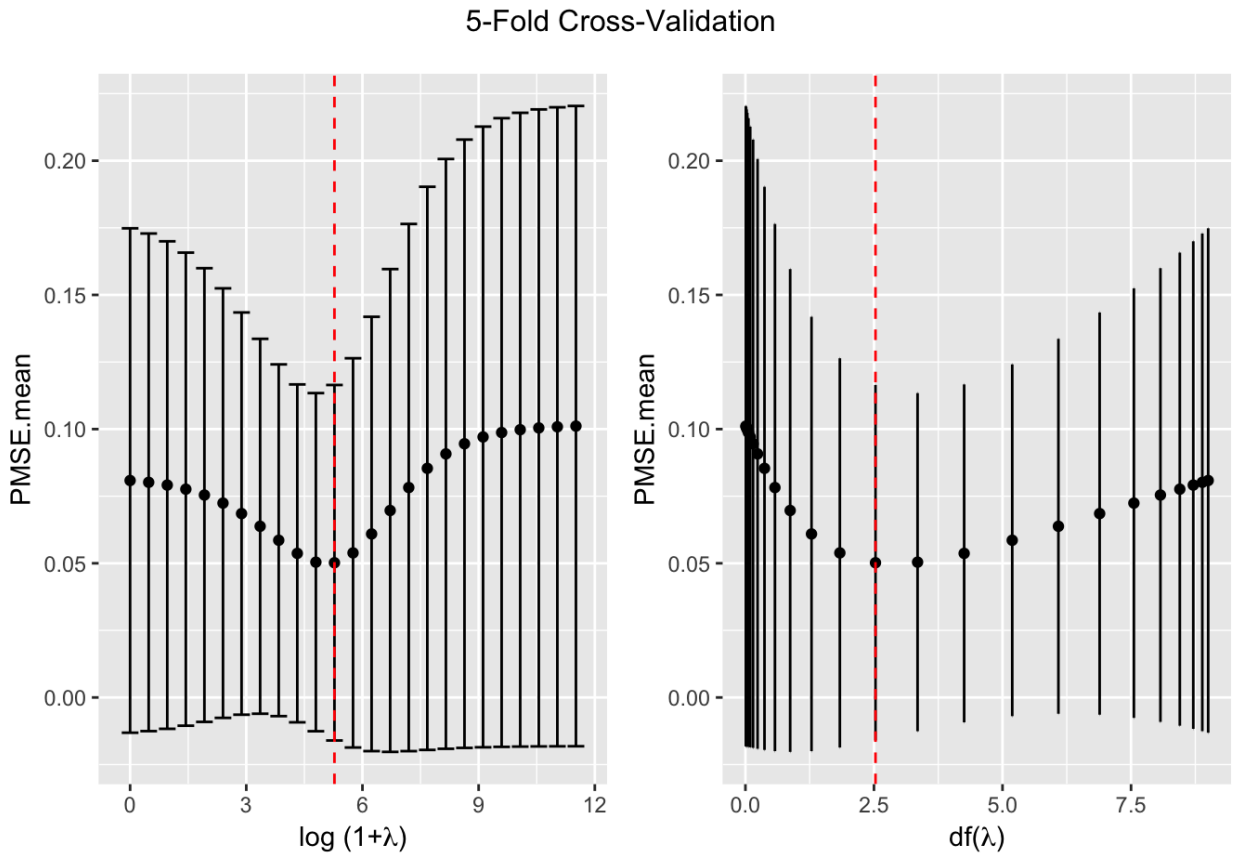
```
Y.B <- scale( Boston$medv, center=TRUE, scale=FALSE) #Centered matrix, mean removal of Y
X.B <- scale( as.matrix(Boston[,!names(Boston)%in%'medv']), center=TRUE, scale=TRUE) #Standardized matr
n.B <- dim(X)[1] #N?mero files
p.B <- dim(X)[2] #N?mero columnes

XtX.B <- t(X)%*%X
d2.B <- eigen(XtX.B,symmetric = TRUE, only.values = TRUE)$values #Autovalors XtX

##############################Lambdes#######################
lambda.max.B <- 1e5 #Fixem valor m?xim lambda. Lambda infinit = 0 graus llibertat/par?metres efectius,
n.lambdas.B <- 25
lambda.v.B <- exp(seq(0,log(lambda.max+1),length=n.lambdas))-1 #Generem vector amb 25 valors de lambda

#############
# effective degrees of freedom
#############
df.v.B <- numeric(n.lambdas.B)
for (l in 1:n.lambdas.B){
  lambda <- lambda.v.B[l]
  df.v.B[l] <- sum(d2/(d2+lambda))
}

PMSE.B.LOOCV<-get_k_cv_metrics(X=X.B,
                               Y=Y.B,
                               lambda.v=lambda.v.B,
                               k=nrow(X.B))
```

```
## [1] "Warning! Doint Leave-One-Out CV"
```

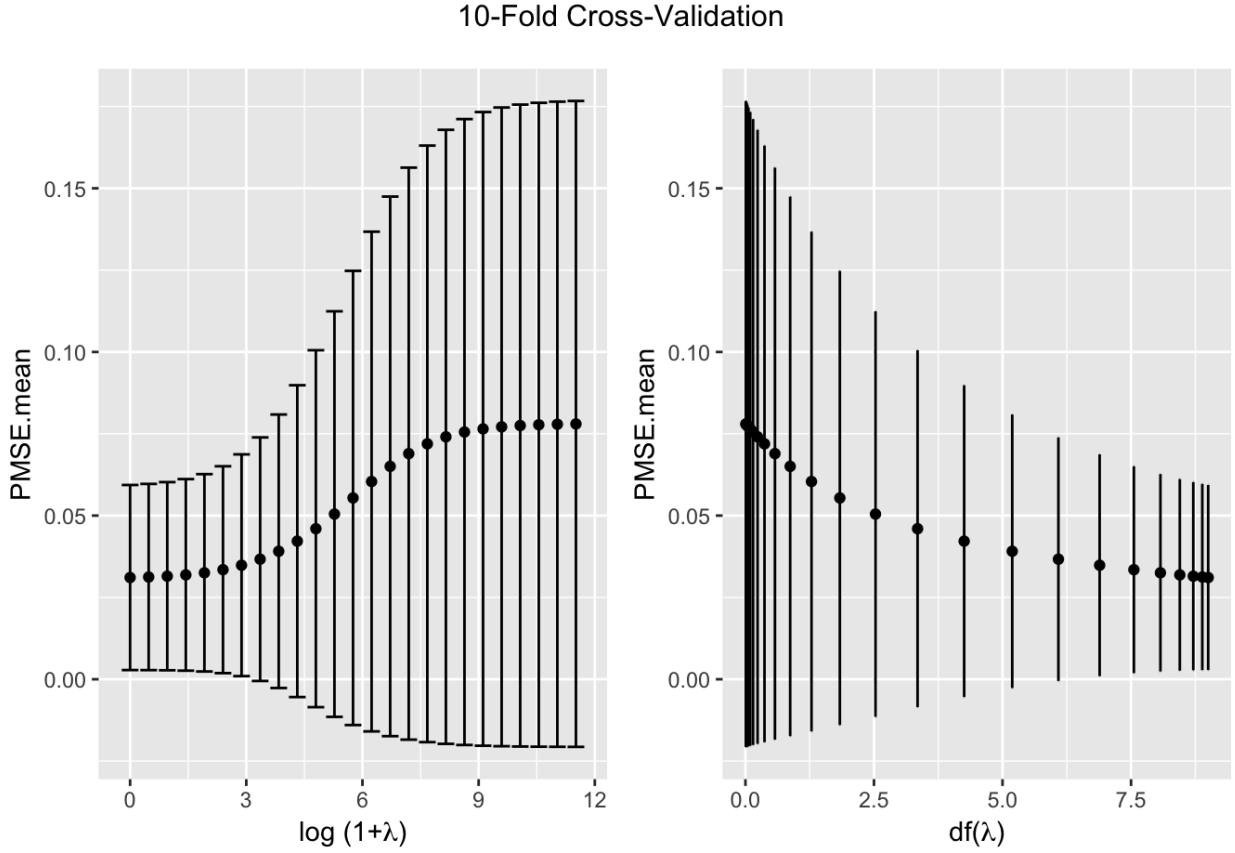
```
PMSE.B.5CV<-get_k_cv_metrics(X,Y,lambda.v,k=5)
PMSE.B.10CV<-get_k_cv_metrics(X,Y,lambda.v,k=10)


get_plot_cv(PMSE.B.LOOCV,title='Leave-one-out cross-validation')
```




```
get_plot_cv(PMSE.B.5CV,title='5-Fold Cross-Validation')
```

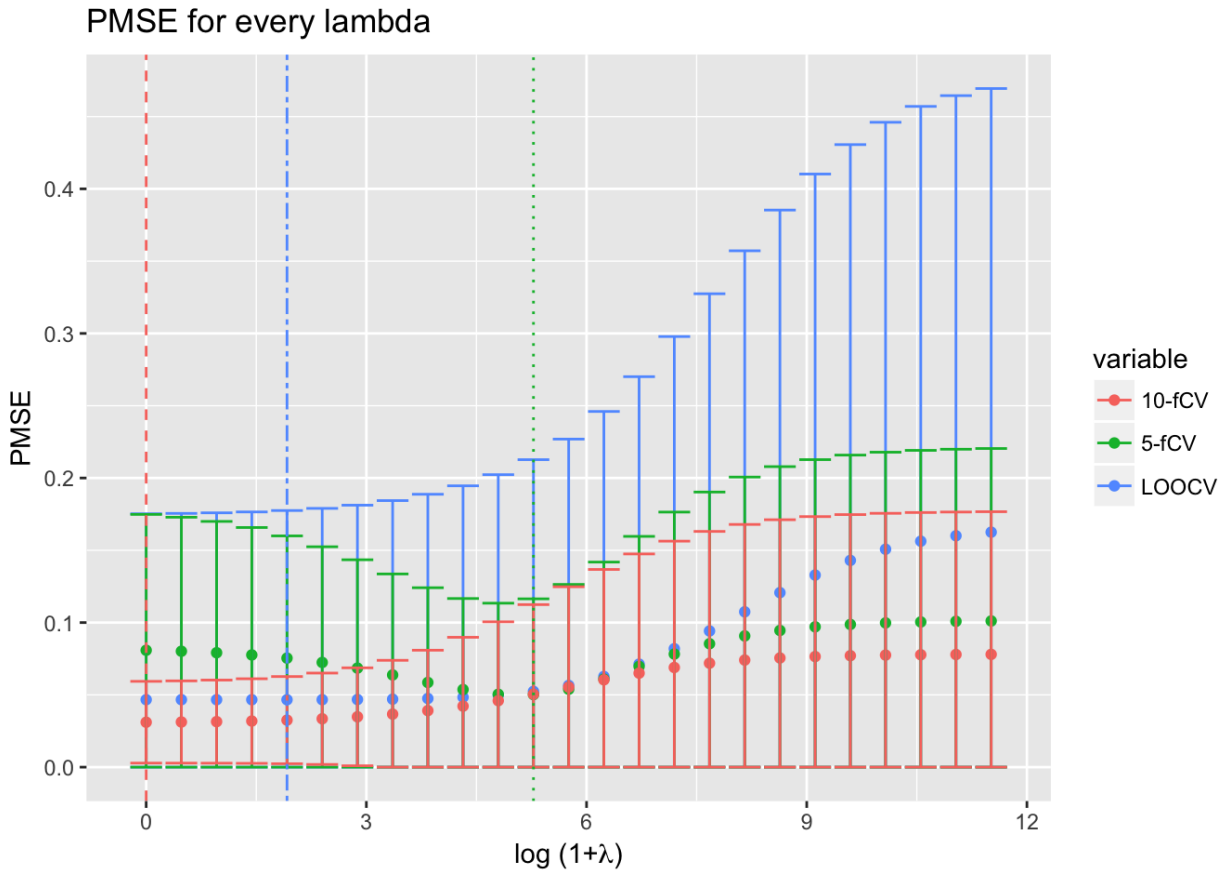



```
get_plot_cv(PMSE.B.10CV,title='10-Fold Cross-Validation')
```




```
dt_final_plot.B<-as.data.frame(rbind(PMSE.B.LOOCV,PMSE.B.5CV,PMSE.B.10CV))
dt_final_plot.B$variable<-c(rep("LOOCV",25),
                            rep("5-fCV",25),
                            rep("10-fCV",25))
dt_final_plot.B$lambda.v<-as.vector(lambda.v)

dt_final_plot.B_min<-dt_final_plot.B %>%
  group_by(variable) %>%
  slice(which.min(PMSE.mean)) %>%
  mutate('log(1+lambda)'=log(1+lambda.v))
dt_final_plot.B_min$lt<-c('dotted','dashed','twodash')

ggplot() +
  geom_point(data=dt_final_plot.B,
             aes(x=log(1+lambda.v),
                 y=PMSE.mean,
                 col=variable)) +
  geom_errorbar(data=dt_final_plot.B,
             aes(x=log(1+lambda.v),
                 ymin=ifelse(PMSE.mean-PMSE.sd<0,0,PMSE.mean-PMSE.sd),
                 ymax=PMSE.mean+PMSE.sd,
                 col=variable)) +
  geom_vline(data=dt_final_plot.B_min,
              aes(xintercept=dt_final_plot.B_min$`log(1+lambda)`,
                  col=variable,
```

```
                linetype=lt),
            show.legend = FALSE) +
  scale_linetype_manual(values=c('dotted','dashed','twodash')) +
  labs(title='PMSE for every lambda',
       x=expression(paste('log (1+',
                  lambda,
                  ')')),
       y='PMSE')
```




```
dt_final_plot.B_min[,c(3,1,2,4,5)] %>%
  set_colnames(c("CV_method","PMSE","PMSE_sd","lambda","log(1+lambda)")) %>%
  kable()
```

CV_method	PMSE	PMSE_sd	lambda	log(1+lambda)
10-fCV	0.0310790	0.0282733	0.000000	0.000000
5-fCV	0.0502015	0.0662292	194.735075	5.276762
LOOCV	0.0466615	0.1308385	5.812932	1.918823

El mètode que minimitza PMSE i la seva variança és en aquest cas el 10-Fold Cross-Validation, resultant així el més adequat davant de LOO o 5-fold.