```
##############################################################################
# 1.
library("ElemStatLearn")
library(MASS)
data(prostate)
# Creating traing data:
train_data<-subset(prostate, train == 1, select = -train)
# Creating test data:
test_data<-subset(prostate, train == 0, select = -train)
#
library(chemometrics)
#getAnywhere(plotRidge)
#
#modifying the plotRidge{chemometrics} function:
#single window plots, colored lines, legend
#
plotRidgeNew <-
  function (formula, data, lambda = seq(0.5, 50, by = 0.05), ...)
  {
    require(pls)
    require(MASS)
    mf <<- match.call(expand.dots = FALSE)
    m <- match(c("formula", "data"), names(mf), 0)
    mf <- mf[c(1, m)]
    mf[[1]] <- as.name("model.frame")
    mf <- eval(mf, parent.frame())
    mt <- attr(mf, "terms")
    y <- model.response(mf, "numeric")
    X <- pls:::delete.intercept(model.matrix(mt, mf))
    ridge = lm.ridge(formula, data, lambda = lambda)
    #par(mfrow = c(1, 2))
    plot(ridge$lambda, ridge$GCV, type = "l", xlab = "lambda",
         ylab = "MSEP by GCV", cex.lab = 1.2)
    lambdaopt = as.numeric(names(which.min(ridge$GCV)))
    mod_ridge = lm.ridge(formula, data, lambda = lambdaopt)
    abline(v = lambdaopt, lty = 2)
    ypred = mean(y) - sum(ridge$xm * mod_ridge$coef/ridge$scale) +
      X %*% (mod_ridge$coef/ridge$scale)
    plot(0, 0, xlim = range(ridge$lambda), ylim = range(ridge$coef),
         type = "n", xlab = "lambda", ylab = "Regression coefficients",
         cex.lab = 1.2)
    #colors added to lines:
    for (i in 1:nrow(ridge$coef)) {
      lines(ridge$lambda, ridge$coef[i, ], col = i)
    }
    #legend added:
    legend("topright",legend=row.names(model.ridge$coef),lty=1,
           horiz=F, cex=0.5,col=1:nrow(ridge$coef))
    abline(v = lambdaopt, lty = 2)
    # adding coefficients to output
    list(predicted = ypred, lambdaopt = lambdaopt, coef=mod_ridge$coef)
  }
####################################################################################
# 1.a)
model.ridge <- lm.ridge(lpsa~.,data=train_data,lambda=seq(0,20,by=0.001))
select(model.ridge)
#optimal value lambda=4.922
#plot(model.ridge$lambda,model.ridge$GCV,type="l")
#str(model.ridge)
```

```
out<-plotRidgeNew(lpsa~.,data=train_data,lambda=seq(0,20,by=0.001))
####################################################################################
# 1.b)
str(out)
l_opt<-out$lambdaopt
#lambdaopt=4.922
opt.ridge = lm.ridge(lpsa~.,data=train_data,lambda=l_opt)
# scaled coefficients:
#coef<-out$coef
# predicted values:
#y_pred<-out$predicted
#or explicitly (mimicking plotRidge):
coef<-opt.ridge$coef
y_pred = mean(train_data[,9]) - sum(colMeans(train_data[,-9]) *
opt.ridge$coef/opt.ridge$scale) +
  as.matrix(train_data[,-9]) %*% (opt.ridge$coef/opt.ridge$scale)
####################################################################################
# 1.c)
R_CV<-ridgeCV(lpsa~.,data=train_data, lambdaopt=l_opt, repl = 5, segments = 10,
        segment.type = "random", plot.opt = TRUE)
# closing split screen from ridgeCV:
par(mfrow = c(1, 1))
R_CV$RMSEP
# repl=5, so 5 RMSEP are calculated for the optimal lambda
#R_CV$residuals
#R_CV$predicted
####################################################################################
# 1.d)
y_pred_test = mean(test_data[,9]) - sum(colMeans(test_data[,-9]) *
opt.ridge$coef/opt.ridge$scale) +
  as.matrix(test_data[,-9]) %*% (opt.ridge$coef/opt.ridge$scale)
mean((y_pred_test-test_data[,9])^2)  #MSE
var(test_data[,9])                  #variance
#
####################################################################################
####################################################################################
# 2.a)
library("ElemStatLearn")
library(MASS)
data(prostate)
# Creating traing data:
train_data<-subset(prostate, train == 1, select = -train)
# Creating test data:
test_data<-subset(prostate, train == 0, select = -train)
#
library(lars)
model.lasso <- lars(as.matrix(train_data[,-9]),train_data[,9])
plot(model.lasso)
print(model.lasso)
# 9 models are calculated
# starting from 0 comp to 8 comp (full model = LS-solution)
# the abscissa is the number of coef (resp. the %-rate of coef): here 8 coef
# the ordinate represents the corresponding value of the coefficients
model.lasso$beta
# the correspondings values of the coef
model.lasso$R2
# coefficients of determination (R^2) of all models
# so the 3 comp solution (R^2=0.58) is cf. to the full model (R^2=0.69) sufficient
#
set.seed(777)
```

```
lassoCV<-cv.lars(as.matrix(train_data[,-9]),train_data[,9],K=10,index=seq(0,1,by=0.05))
# garph. analysis (ellbow-criterion) also suggests a 3-4 comp solution
###################################################################################
# 2.b)
library(chemometrics)
set.seed(777)
lassoCV<-lassoCV(lpsa~.,data=train_data, K = 10, fraction = seq(0, 1, by = 0.05), sdfact
= 1)
# plot of MSEP values at each value of fraction plus standard errors
# to mimic the procedure of the lecture notes of finding the optimal value one has
# to set sdfact = 1
# so the sopt=0.35 wich leads to a 3 comp solution as supposed before
# values shown in plot:
lassoCV$cv[lassoCV$ind]
lassoCV$SEP[lassoCV$ind]
lassoCV$sopt

# plot of optimal solution:
lassocoef(lpsa~.,data=train_data, sopt=lassoCV$sopt)
##################################################################################
# 2.c)
# computing the MSE:
# smaller than ridge-MSE!
y_pred<-predict.lars(model.lasso, test_data[,-9], s=lassoCV$sopt, type = "fit", mode =
"lambda")
mean((y_pred$fit-test_data[,9])^2)  #MSE
var(test_data[,9])                  #variance y
```