```
setwd("/Users/Jiguo/Dropbox/Teaching/852/Rcodes")

# LAR on prostate data using lars package
# Splitting the data in half and modeling each half separately.

prostate <-  read.table("Prostate.csv", header=TRUE, sep=",", na.strings=" ")
head(prostate)
install.packages("lars")
library(lars)

# Splitting data in half using random uniform selection to make two "set"s.

set.seed(120401002)
prostate$set <- ifelse(runif(n=nrow(prostate))>0.5, yes=2, no=1)

# lars() requires x to be in matrix class, so saving out
#   the separate variables to be used as Y and X.

y.1 <- prostate[which(prostate$set==1),10]
x.1 <- as.matrix(prostate[which(prostate$set==1),c(2:9)])
y.2 <- prostate[which(prostate$set==2),10]
x.2 <- as.matrix(prostate[which(prostate$set==2),c(2:9)])

# Fit LAR by lars(y=, x=). Lasso is default, but can also do stepwise, stagewise, and
LAR
#  Function produces one fit at each new variable entry.
# Cross-Validation for LASSO chops up the L1-norm into sequence of 100 points.
#  SE for CV is found from the sample SE of the squared errors.  NOT from rerunning CV
multiple times.
# Therefore CV may be inappropriate for small n.  Use Cp from Ssummary() instead.

# First half of data
lasso.1 <- lars(y=y.1, x= x.1, type="lasso", trace=TRUE)
summary(lasso.1)
plot(lasso.1) # Plots coefficient path
coef(lasso.1) # Lists out coefficients for each step in path
# cv.lars() uses crossvalidation to estimate optimal position in path
cv.lasso.1 <- cv.lars(y=y.1, x= x.1, type="lasso")
cv.lasso.1
# Use the "+1SE rule" to find best model:
#    Take the min CV and add its SE ("limit").
#    Find smallest model that has its own CV within this limit (at "s.cv.1")
limit <- min(cv.lasso.1$cv) + cv.lasso.1$cv.error[which.min(cv.lasso.1$cv)]
s.cv.1 <- cv.lasso.1$index[min(which(cv.lasso.1$cv < limit))]
# Print out coefficients at optimal s.
coef(lasso.1, s=s.cv.1, mode="fraction")
# Predict both halves using first-half fit
(predict.1.1 <- predict(lasso.1, newx=x.1, s=s.cv.1, mode="fraction"))
(predict.1.2 <- predict(lasso.1, newx=x.2, s=s.cv.1, mode="fraction"))

# Repeat for second half of data
lasso.2 <- lars(y=y.2, x= x.2, type="lasso", trace=TRUE)
summary(lasso.2)
plot(lasso.2) # Plots coefficient path
```

```
coef(lasso.2) # Lists out coefficients for each step in path
# cv.lars() uses crossvalidation to estimate optimal position in path
cv.lasso.2 <- cv.lars(y=y.2, x= x.2, type="lasso")
cv.lasso.2
# Use the "+1SE rule" to find best model:
#    Take the min CV and add its SE ("limit").
#    Find smallest model that has its own CV within this limit (at "s.cv.2")
limit <- min(cv.lasso.2$cv) + cv.lasso.2$cv.error[which.min(cv.lasso.2$cv)]
s.cv.2 <- cv.lasso.2$index[min(which(cv.lasso.2$cv < limit))]
# Print out coefficients at optimal s.
coef(lasso.2, s=s.cv.2, mode="fraction")
# Predict both halves using first-half fit
predict.2.1 <- predict(lasso.2, newx=x.1, s=s.cv.2, mode="fraction")
predict.2.2 <- predict(lasso.2, newx=x.2, s=s.cv.2, mode="fraction")

plot(x=c(predict.1.1$fit,predict.1.2$fit), y=c(predict.2.1$fit,predict.2.2$fit),
     xlab="Predictions using Set 1 model", ylab="predictions using Set 2 model")
abline(a=0,b=1)

#########################
#     LAR fit
#########################

# First half of data
lar.1 <- lars(y=y.1, x= x.1, type="lar", trace=TRUE)
summary(lar.1)
plot(lar.1) # Plots coefficient path
coef(lar.1) # Lists out coefficients for each step in path
# cv.lars() uses crossvalidation to estimate optimal position in path
cv.lar.1 <- cv.lars(y=y.1, x= x.1, type="lar")
cv.lar.1

# Use the best Cp value to find best model:
a <- summary(lar.1)
# Print out coefficients at optimal s.
coef(lar.1, s=which.min(a$Cp), mode="step")
```