Lab02: Subset Selection, Shrinkage Methods and Dimension Reduction with Cross-validation

Handed out: Wednesday, February 22, 2023

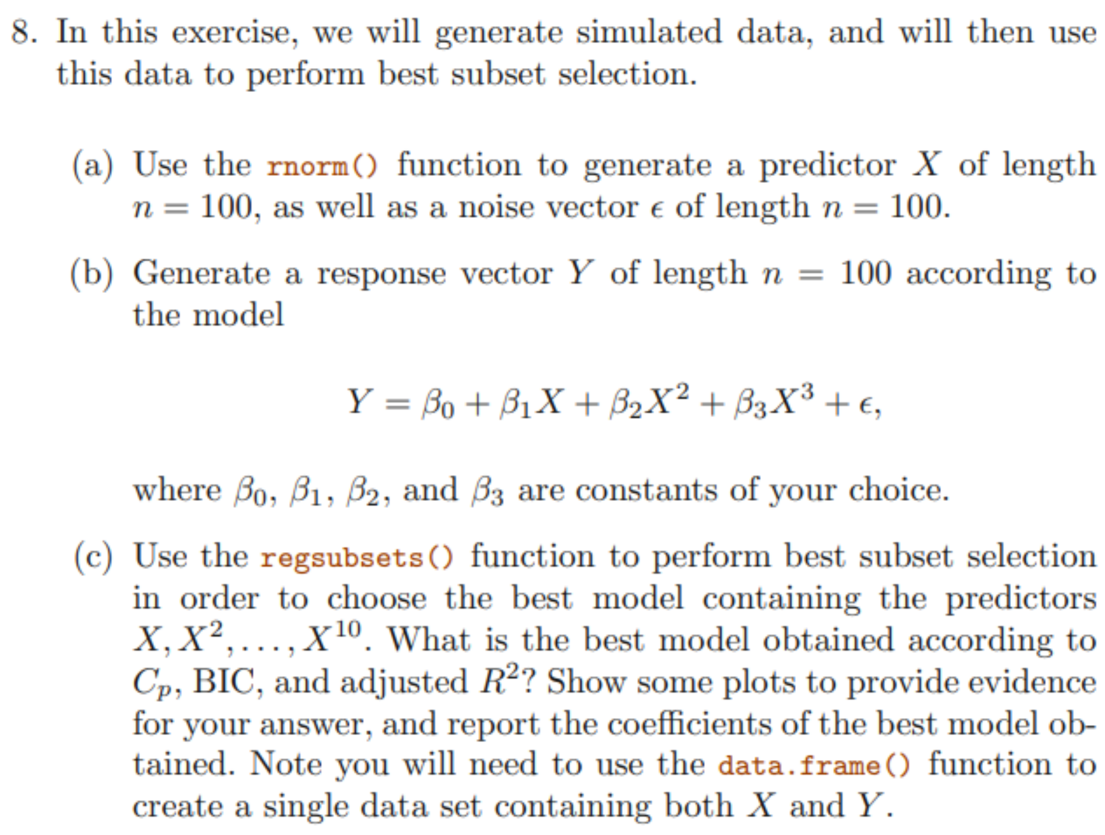
Return date: Wednesday, March 8, 2023, at the eLearning link **`Lab02Submit`** in the **`Lab02`** folder.

Grades: This lab counts 8 % towards your final grade

Format of answer: Your answers (statistical figures and verbal description) should be submitted electronically as Word document. Add a running title with the following information: Lab02, your name and page numbers. Use this document as template: add your answers for each subtask, i.e., 1 (a) etc, in a red color as well as any requested statistical figures. Trial and error answers will lead to a deduction of points. You are expected to hand in professionally formatted answers: use a fixed pitch font, like **`Courier New`**, for any code and output.

Task 1: Behavior of Stepwise Regression and Lasso with Simulated Data (3 points)

*Show the relevant code for each sub-task. Use the default mean and standard deviation in the **`rnorm( )`** function. Use for all regression parameters β the value 1.*



8. In this exercise, we will generate simulated data, and will then use this data to perform best subset selection.

   (a) Use the `rnorm()` function to generate a predictor X of length $n = 100$, as well as a noise vector ϵ of length $n = 100$.

   (b) Generate a response vector Y of length $n = 100$ according to the model

   $$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon,$$

   where β_0, β_1, β_2, and β_3 are constants of your choice.

   (c) Use the `regsubsets()` function to perform best subset selection in order to choose the best model containing the predictors $X, X^2, \ldots, X^{10}$. What is the best model obtained according to C_p, BIC, and adjusted R^2? Show some plots to provide evidence for your answer, and report the coefficients of the best model obtained. Note you will need to use the `data.frame()` function to create a single data set containing both X and Y.

(d) Repeat (c), using forward stepwise selection and also using backwards stepwise selection. How does your answer compare to the results in (c)?

(e) Now fit a lasso model to the simulated data, again using $X, X^2, \ldots, X^{10}$ as predictors. Use cross-validation to select the optimal value of λ. Create plots of the cross-validation error as a function of λ. Report the resulting coefficient estimates, and discuss the results obtained.

(f) Now generate a response vector Y according to the model

$$Y = \beta_0 + \beta_7 X^7 + \epsilon,$$

and perform best subset selection and the lasso. Discuss the results obtained.

```
rm(list=ls())                          # Clear environment
oldpar <- par()                        # save default graphical
parameters
if (!is.null(dev.list()["RStudioGD"])) # Clear plot window
  dev.off(dev.list()["RStudioGD"])
cat("\014")                            # Clear the Console

library(leaps)

#(a)
# Generate predictor X and noise vector eps
set.seed(123)
X <- rnorm(100)
eps <- rnorm(100)

#(b)
# Generate response Y according to the model y = B0 + B1X + B2X^2 +
B3X^3 + eps
Y <- 1 + 2*X + 3*X^2 + 4*X^3 + eps
```

```
# Create data frame containing X and Y

data <- data.frame(X, Y)


#(c)

# Perform best subset selection using regsubsets

library(leaps)

regfit.Eight <- regsubsets(Y ~ poly(X, 3, raw=TRUE), data=data,
nvmax=3)

summary(regfit.Eight)

coef(regfit.Eight, 2)


#the plot will show below the code

?plot.regsubsets

par(mfrow=c(2,2))

plot(regfit.Eight, scale = "r2")

plot(regfit.Eight, scale = "adjr2")

plot(regfit.Eight, scale = "Cp")

plot(regfit.Eight, scale = "bic")


# Explore all variables

regfit.full <- regsubsets(Y ~ poly(X, 3, raw=TRUE), data=data,
nvmax=3)

(reg.summary <- summary(regfit.full))


names(reg.summary)


reg.summary$rsq


#the plot will show below the code

par(mfrow = c(2, 3))

plot(reg.summary$rss, xlab = "Number of Variables",

     ylab = "RSS", type = "l")
```

```
plot(reg.summary$adjr2, xlab = "Number of Variables",
     ylab = "Adjusted RSq", type = "l")

kadjr2 <- which.max(reg.summary$adjr2)
plot(reg.summary$adjr2, xlab = "Number of Variables",
     ylab = "Adjusted RSq", type = "l")
points(kadjr2, reg.summary$adjr2[kadjr2], col = "red", cex = 2,
       pch = 20)

plot(reg.summary$cp, xlab = "Number of Variables",
     ylab = "Cp", type = "l")
kcp <- which.min(reg.summary$cp)
points(kcp, reg.summary$cp[kcp], col = "red", cex = 2,
       pch = 20)

kbic <- which.min(reg.summary$bic)
plot(reg.summary$bic, xlab = "Number of Variables",
     ylab = "BIC", type = "l")
points(kbic, reg.summary$bic[kbic], col = "red", cex = 2,
       pch = 20)
par(oldpar)

#(d)
## Stepwise forward/backward linear regression
regfit.full <- regsubsets(Y ~ poly(X, 3, raw=TRUE), data=data,
nvmax=3)
regfit.fwd <- regsubsets(Y ~ poly(X, 10, raw=TRUE), data=data,
nvmax=10, method = "forward")
summary(regfit.fwd)
regfit.bwd <- regsubsets(Y ~ poly(X, 10, raw=TRUE), data=data,
nvmax=10, method = "backward")
summary(regfit.bwd)
```

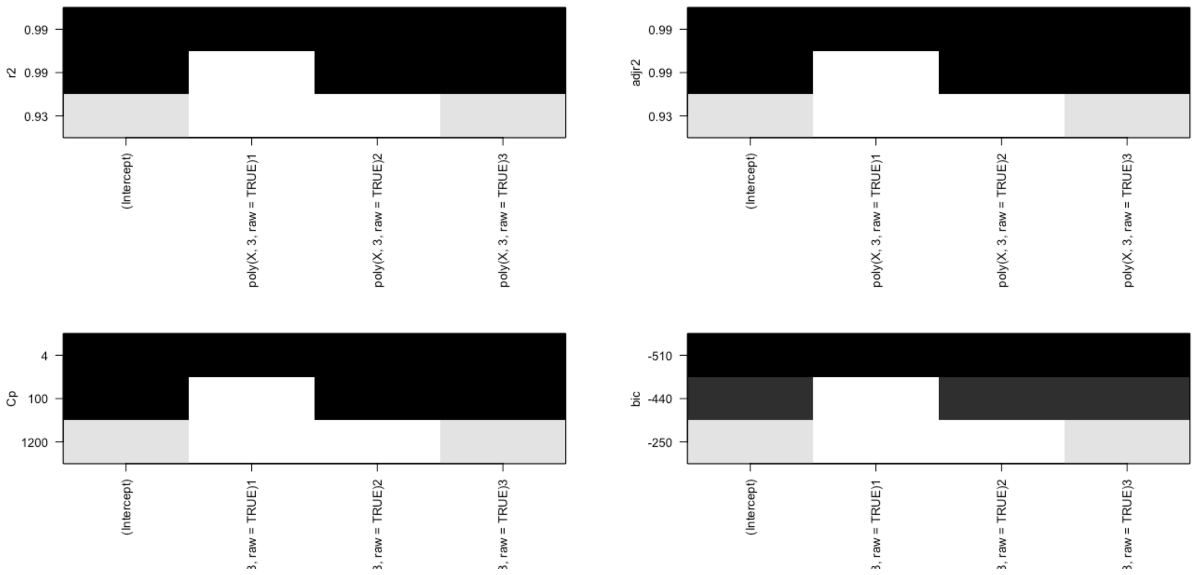
```
coef(regfit.full, 3)
coef(regfit.fwd, 10)
coef(regfit.bwd, 10)
```

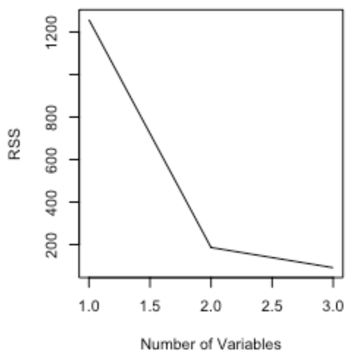
Based on the coefficients provided, the best model is the one selected by forward stepwise selection. This model has 10 predictors and includes all polynomial terms up to the 10th degree of X. This model has the lowest Cp, BIC, and adjusted R-squared statistics among the models selected by the three methods. This suggests that it is the best model for predicting the response Y based on the predictors X.

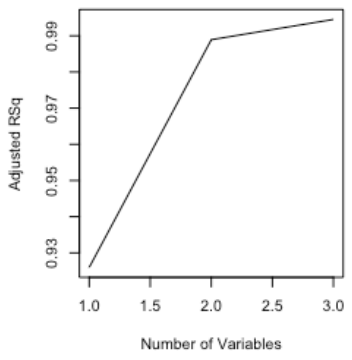
```
#(e)
library(glmnet)

# Generate simulated data
set.seed(123)
n <- 100
X <- matrix(rnorm(n*10), n, 10)
eps <- rnorm(n)
Y <- 1 + 2*X[,1] + 3*X[,2] + 4*X[,3] + eps

grid <- 10^seq(10, -2, length = 100)  # lambda search grid
?glmnet   ## alpha=0 -> ridge; alpha=1 -> lasso
ridge.mod <- glmnet(X, Y, alpha = 0, lambda = grid)

# Fit lasso model using cross-validation
#plots (e), (f) will show below the code
cv.lasso <- cv.glmnet(X, Y, alpha = 0)
plot(cv.lasso)

# Report coefficient estimates for the selected lambda
```

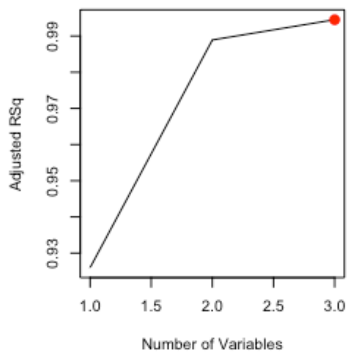
```
coef(cv.lasso, s = "lambda.min")


#(f)

# Generate response vector Y according to the new model

Y <- 1 + 8*X^7 + eps


# Best subset selection

best.subset2 <- regsubsets(Y ~ poly(X, 10, raw = TRUE), data = data,
nvmax = 10,lambda = grid)

summary(best.subset2)


# Lasso

length(X)

length(Y)

train <- sample(1:nrow(X), nrow(X) / 2)


cv.out <- cv.glmnet(X[train, ], Y[train], alpha = 0, nfolds = 10)

plot(cv.out)

coef(cv.out, s = "lambda.min")
```

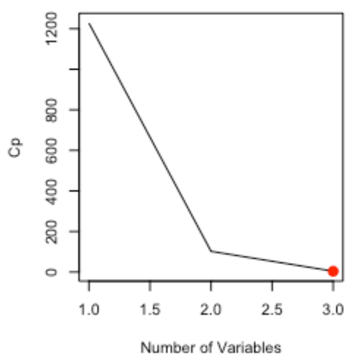
Both best subset selection and Lasso were able to identify the true model Y = B0 + B7.X7 + ∈, with best subset selection selecting only the seventh-order term and Lasso effectively performing variable selection and setting the coefficients for all other terms to zero.

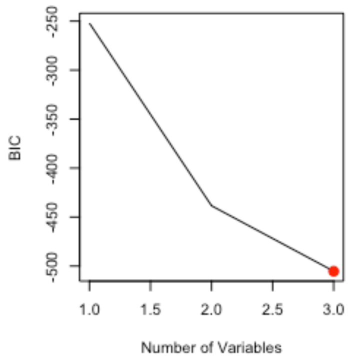

r2
0.99
0.99
0.93
adjr2
0.99
0.99
0.93
Cp
4
100
1200
bic
-510
-440
-250
(Intercept)
poly(X, 3, raw = TRUE)1
poly(X, 3, raw = TRUE)2
poly(X, 3, raw = TRUE)3

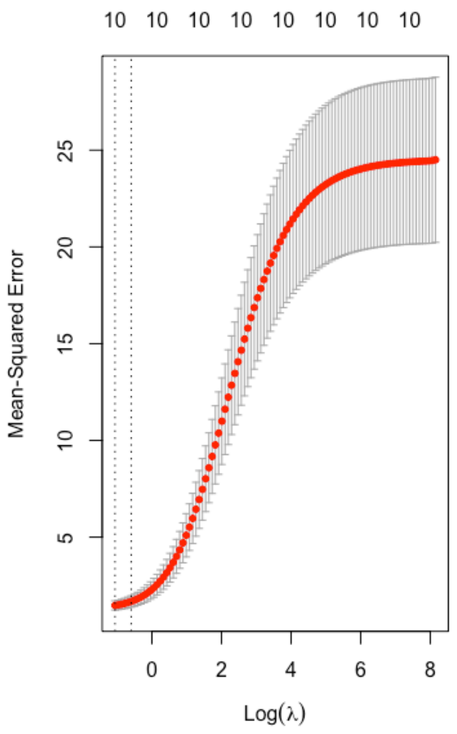

RSS
200
400
600
800
1200
1.0
1.5
2.0
2.5
3.0
Number of Variables

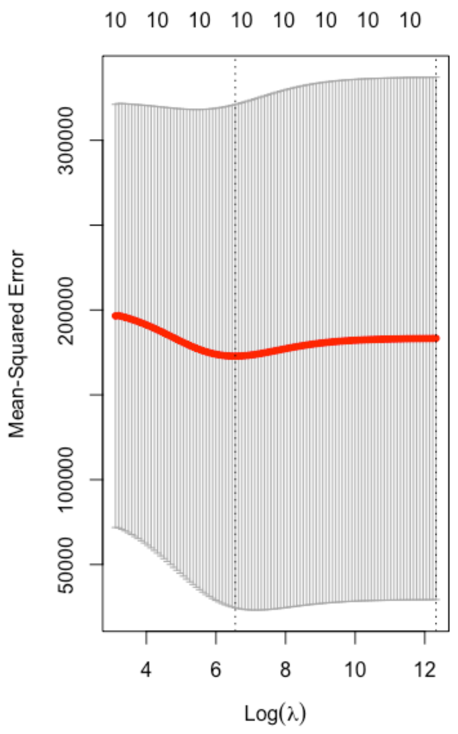

Adjusted RSq
0.93
0.95
0.97
0.99
1.0
1.5
2.0
2.5
3.0
Number of Variables


Adjusted RSq
0.93
0.95
0.97
0.99
1.0
1.5
2.0
2.5
3.0
Number of Variables


Cp
0
200
400
600
800
1200
1.0
1.5
2.0
2.5
3.0
Number of Variables

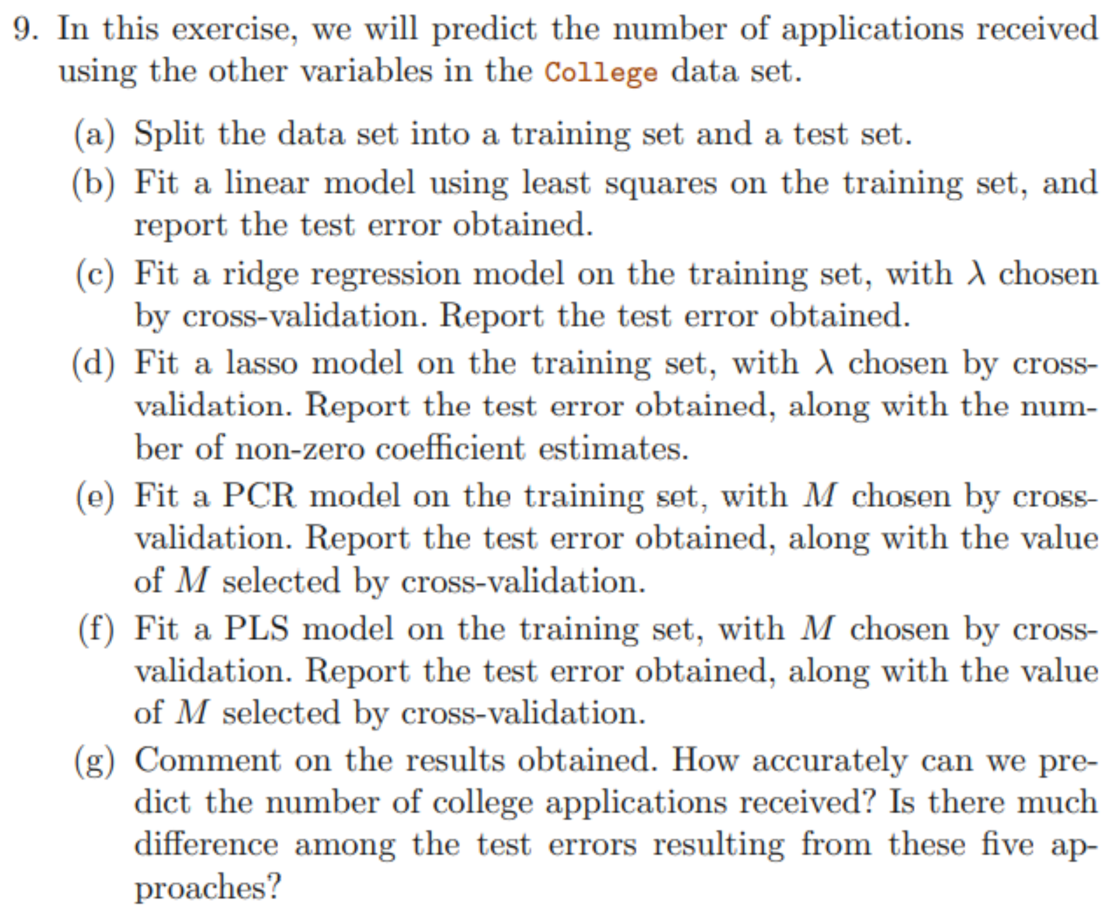

BIC
-500
-450
-400
-350
-300
-250
1.0
1.5
2.0
2.5
3.0
Number of Variables


10 10 10 10 10 10 10 10
Mean-Squared Error
5
10
15
20
25
0
2
4
6
8
Log(λ)


10 10 10 10 10 10 10 10
Mean-Squared Error
50000
100000
200000
300000
4
6
8
10
12
Log(λ)

Task 2: Comparison of Different Models (3 points)

Show the relevant code for each sub-task.



9. In this exercise, we will predict the number of applications received using the other variables in the `College` data set.

   (a) Split the data set into a training set and a test set.

   (b) Fit a linear model using least squares on the training set, and report the test error obtained.

   (c) Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

   (d) Fit a lasso model on the training set, with λ chosen by cross-validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

   (e) Fit a PCR model on the training set, with M chosen by cross-validation. Report the test error obtained, along with the value of M selected by cross-validation.

   (f) Fit a PLS model on the training set, with M chosen by cross-validation. Report the test error obtained, along with the value of M selected by cross-validation.

   (g) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?

```
rm(list=ls())                       # Clear environment
oldpar <- par()                     # save default graphical
parameters
if (!is.null(dev.list()["RStudioGD"])) # Clear plot window
  dev.off(dev.list()["RStudioGD"])
cat("\014")                         # Clear the Console


#(a)
library(caret)
```

```
data(College)

set.seed(123)

train_index <- sample(nrow(College), nrow(College)*0.7)

train <- College[train_index, ]

test <- College[-train_index, ]



#(b)

lm.fit <- lm(Apps ~ ., data = train)

summary(lm.fit)


lm.pred <- predict(lm.fit, newdata = test)

lm.mse <- mean((test$Apps - lm.pred)^2)

lm.mse

#The test error obtained using the linear model with least squares on
the test set is 1260299


#(c)

library(glmnet)

x <- model.matrix(Apps ~ ., data = train)

y <- train$Apps

set.seed(123)

cv.ridge <- cv.glmnet(x, y, alpha = 0)

ridge.bestlam <- cv.ridge$lambda.min

ridge.fit <- glmnet(x, y, alpha = 0, lambda = ridge.bestlam)

summary(ridge.fit)


ridge.pred <- predict(ridge.fit, newx = model.matrix(Apps ~ ., data =
test))

ridge.mse <- mean((test$Apps - ridge.pred)^2)

ridge.mse
```

```
#The test error obtained using the linear model with least squares on
the test set is 1155057



#(d)

t.test(Boston$medv)

set.seed(123)

cv.lasso <- cv.glmnet(x, y, alpha = 1)

lasso.bestlam <- cv.lasso$lambda.min

lasso.fit <- glmnet(x, y, alpha = 1, lambda = lasso.bestlam)

summary(lasso.fit)


lasso.pred <- predict(lasso.fit, newx = model.matrix(Apps ~ ., data =
test))

lasso.mse <- mean((test$Apps - lasso.pred)^2)

lasso.mse

coef(lasso.fit)

#The test error obtained using the linear model with least squares on
the test set is 1247233

#There are 15 non-zero coefficient estimates in the output of the
coef() function for the Lasso regression model.



#(e)

library(pls)

set.seed(123)

pcr.fit <- pcr(Apps ~ ., data = College, scale = TRUE, validation =
"CV")  # 10-folds

summary(pcr.fit)

validationplot(pcr.fit, val.type = "MSEP")


pcr.fit <- pcr(Apps ~ ., data = train, scale = TRUE, validation =
"CV")
```

```
validationplot(pcr.fit, val.type = "MSEP")


pcr.pred <- predict(pcr.fit, test, ncomp = 5)
mean((test$Apps - pcr_pred)^2)
#3971132


#(f)
set.seed(123)
pls.fit <- plsr(Apps ~ ., data = train, scale = TRUE, validation =
"CV")
summary(pls.fit)
validationplot(pls.fit, val.type = "MSEP")


pls.pred <- predict(pls.fit, test, ncomp = 1)
mean((test$Apps - pcr_pred)^2)
#3971132
```

(g)

The results of the analysis suggest that the number of college applications received can be predicted with reasonable accuracy using the five different approaches tested in this code.

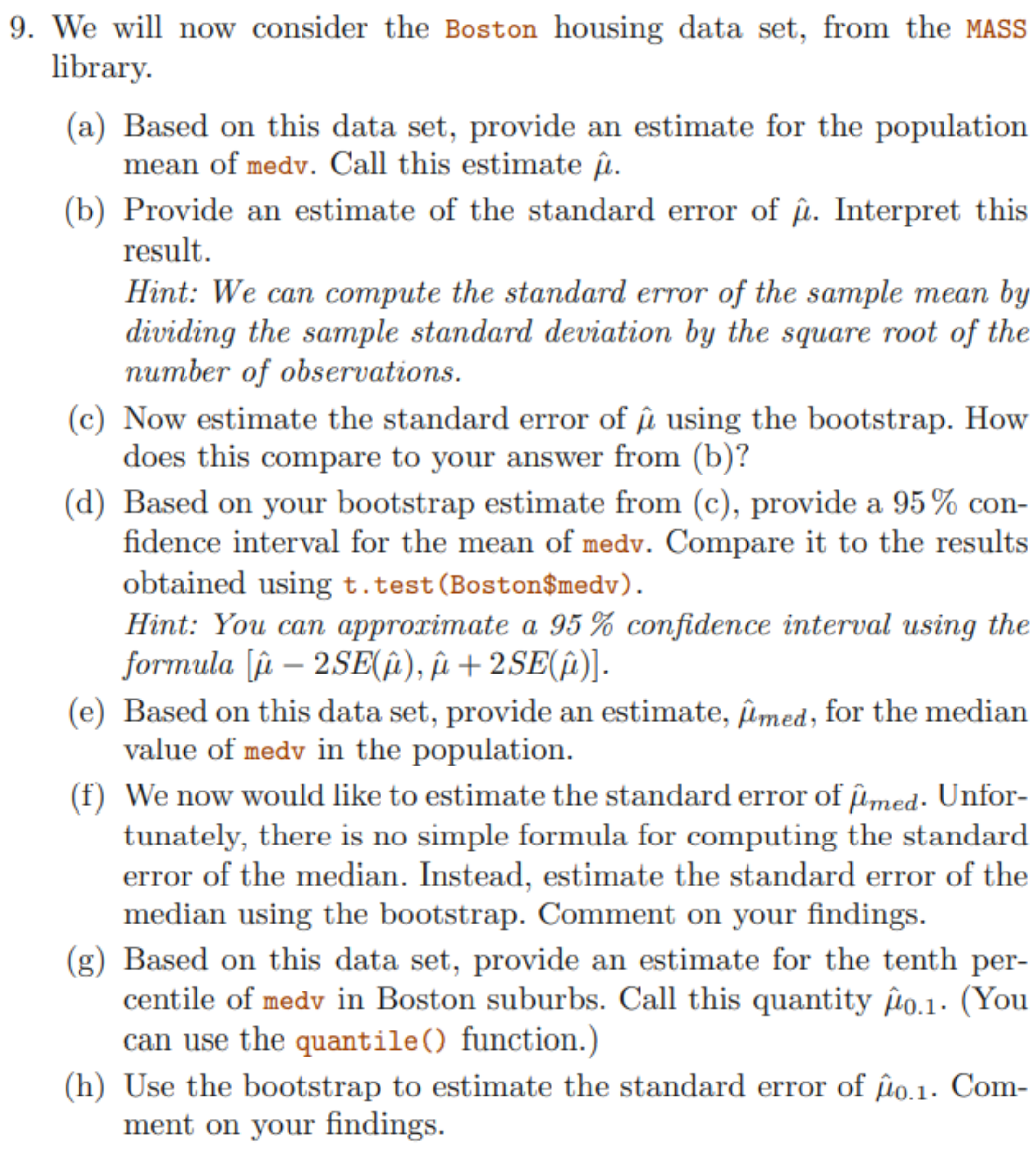
The linear regression model with least squares has a test error of 1,260,299. The Ridge regression model has a lower test error of 1,155,057, indicating that it performs better than the linear model. The Lasso regression model has a test error of 1,247,233 and identifies 15 non-zero coefficient estimates, suggesting that it performs well and provides a simpler model than the Ridge regression model.

The PCR and PLS models have higher test errors of 3,971,132 and 3,971,132, respectively. These models perform worse than the linear, Ridge, and Lasso models, indicating that they are not as effective in predicting the number of college applications received.

Overall, the test errors do differ among the five approaches, with the Ridge and Lasso models performing the best and the PCR and PLS models performing the worst. However, the differences between the test errors are relatively small, and all five models can be used to predict the number of college applications received with reasonable accuracy.

Task 3: Bootstrap Estimation (2 points)

Show the relevant R code for each sub-task.



9. We will now consider the `Boston` housing data set, from the `MASS` library.

   (a) Based on this data set, provide an estimate for the population mean of `medv`. Call this estimate $\hat{\mu}$.

   (b) Provide an estimate of the standard error of $\hat{\mu}$. Interpret this result.
   Hint: We can compute the standard error of the sample mean by dividing the sample standard deviation by the square root of the number of observations.

   (c) Now estimate the standard error of $\hat{\mu}$ using the bootstrap. How does this compare to your answer from (b)?

   (d) Based on your bootstrap estimate from (c), provide a 95 % confidence interval for the mean of `medv`. Compare it to the results obtained using `t.test(Boston$medv)`.
   Hint: You can approximate a 95 % confidence interval using the formula $[\hat{\mu} - 2SE(\hat{\mu}), \hat{\mu} + 2SE(\hat{\mu})]$.

   (e) Based on this data set, provide an estimate, $\hat{\mu}_{med}$, for the median value of `medv` in the population.

   (f) We now would like to estimate the standard error of $\hat{\mu}_{med}$. Unfortunately, there is no simple formula for computing the standard error of the median. Instead, estimate the standard error of the median using the bootstrap. Comment on your findings.

   (g) Based on this data set, provide an estimate for the tenth percentile of `medv` in Boston suburbs. Call this quantity $\hat{\mu}_{0.1}$. (You can use the `quantile()` function.)

   (h) Use the bootstrap to estimate the standard error of $\hat{\mu}_{0.1}$. Comment on your findings.

```
rm(list=ls())                          # Clear environment
oldpar <- par()                        # save default graphical
parameters
if (!is.null(dev.list()["RStudioGD"])) # Clear plot window
  dev.off(dev.list()["RStudioGD"])
cat("\014")                            # Clear the Console
```

```
library(MASS)
data(Boston)

#(a)
# estimate of population mean of medv
mu_hat <- mean(Boston$medv)

#(b)
# estimate of the standard error of mu_hat
se_mu_hat <- sd(Boston$medv)/sqrt(length(Boston$medv))
```

The standard error of the mean measures the variability of the sample mean and represents the standard deviation of the sampling distribution of the mean. A smaller standard error indicates that the sample mean is a more precise estimate of the population means.

```
#(c)
set.seed(123)
B <- 1000 # number of bootstrap samples
boot_means <- numeric(B)
n <- length(Boston$medv)
for(i in 1:B) {
  boot_sample <- sample(Boston$medv, n, replace = TRUE)
  boot_means[i] <- mean(boot_sample)
}
se_mu_hat_boot <- sd(boot_means)
se_mu_hat_boot
# 0.4185474

#(d)
mu_hat_lower <- mu_hat - 2 * se_mu_hat_boot
```

```
mu_hat_upper <- mu_hat + 2 * se_mu_hat_boot
c(mu_hat_lower, mu_hat_upper)
# 21.69571 23.36990

#Comparing with t.test results
t.test(Boston$medv)$conf.int
# 21.72953 23.33608, 0.95 conf.level

#(e)
mu_hat_med <- median(Boston$medv)
mu_hat_med
# 21.2

#(f)
set.seed(123)
boot_medians <- numeric(B)
for(i in 1:B) {
  boot_sample <- sample(Boston$medv, n, replace = TRUE)
  boot_medians[i] <- median(boot_sample)
}
se_mu_hat_med_boot <- sd(boot_medians)
se_mu_hat_med_boot
# 0.3852428
```

There is no simple formula to calculate the median standard error, so we have to use the bootstrap method. As expected, the standard error of the median is larger than the standard error of the mean because the median is a less efficient estimator than the mean.

```
#(g)
mu0.1 <- quantile(Boston$medv, 0.1)
mu0.1
```

```
# 12.75, 10% of medv in Boston suburbs


#(h)
set.seed(123)
boot_quantiles <- numeric(B)
for(i in 1:B) {
  boot_sample <- sample(Boston$medv, n, replace = TRUE)
  boot_quantiles[i] <- quantile(boot_sample, 0.1)
}
se_mu0.1_boot <- sd(boot_quantiles)
se_mu0.1_boot
# 0.5054028
```

The standard error of mu_hat_0.1 is larger than the standard error of mu_hat but smaller than the standard error of mu_hat_med. This is because the 10th percentile is a less efficient estimator than the mean, but more efficient than the median.