Statistical Learning and Data mining

Homework 9

M052040003 鍾冠毅

6.2.

a. iii.

Due to the less bias and more variance, it is less flexible and having better predictions.

b. iii.

It's the same reason to the above.

c. ii.

On the contrary, it is more flexible and having less bias and more variance.

6.3.

a. iv.

As the value of s goes up, the more predictors will be chosen so that the training data will be more fitted. Thus, the training RSS will goes down steadily. Notice that all the predictors will be chosen while s goes infinitely.

b. ii.

The test RSS is the sum of square bias and variance. Choosing more predictors makes the model more flexible with having less bias and more variance. Hence the test RSS initially goes down and then goes up. The test RSS has a U-shaped curve.

c. iii.

The reason has been already mentioned above.

d. iv.

The reason has been already mentioned above.

e. v.

The irreducible error won't change no matter what value of s is. This part of error is unexplainable in this models.

6.4.

a. iii.

As the value of λ goes up, the less predictors will be chosen so that the training data will be less fitted. Thus, the training RSS will goes up steadily. Notice that all the predictors will be chosen while $\lambda = 0$.

b. ii.

The test RSS is the sum of square bias and variance. Choosing more predictors makes the model more flexible with having less bias and more variance. Hence the test RSS initially goes down and then goes up. The test RSS has a U-shaped curve.

c. iv.

The reason has been already mentioned above.

d. iii.

The reason has been already mentioned above.

e. v.

The irreducible error won't change no matter what value of λ is. This part of error is unexplainable in this models.

8.

a.

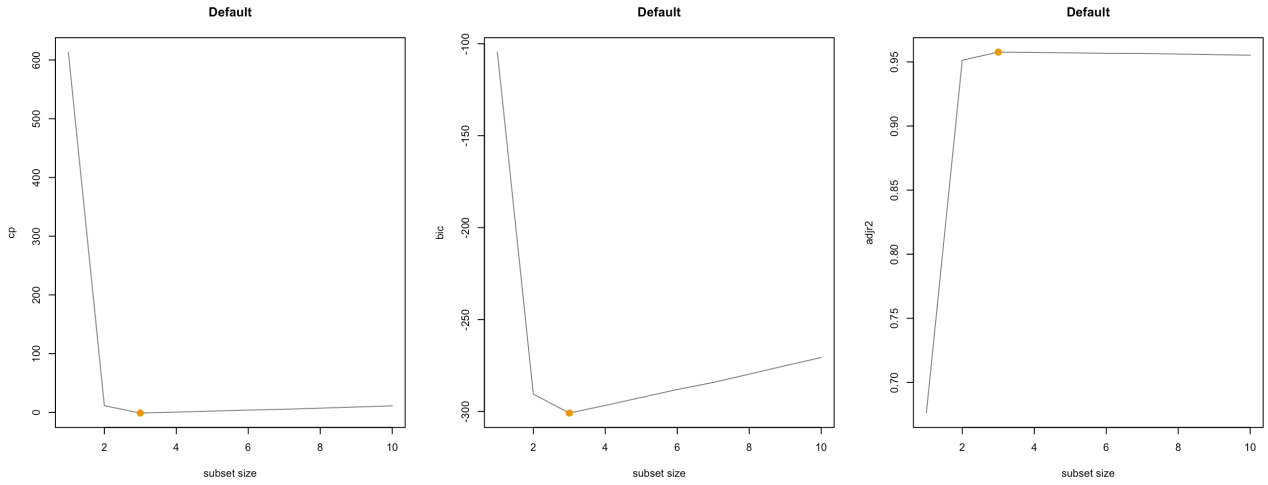
See the appendix.

b.

$$Y = 2.99 + 2.2X - 3.5X^2 + 0.2X^3 + \epsilon$$

c.

```
> c(which.min(ms.c$cp), which.min(ms.c$bic), which.max(ms.c$adjr2))
[1] 3 3 3
```

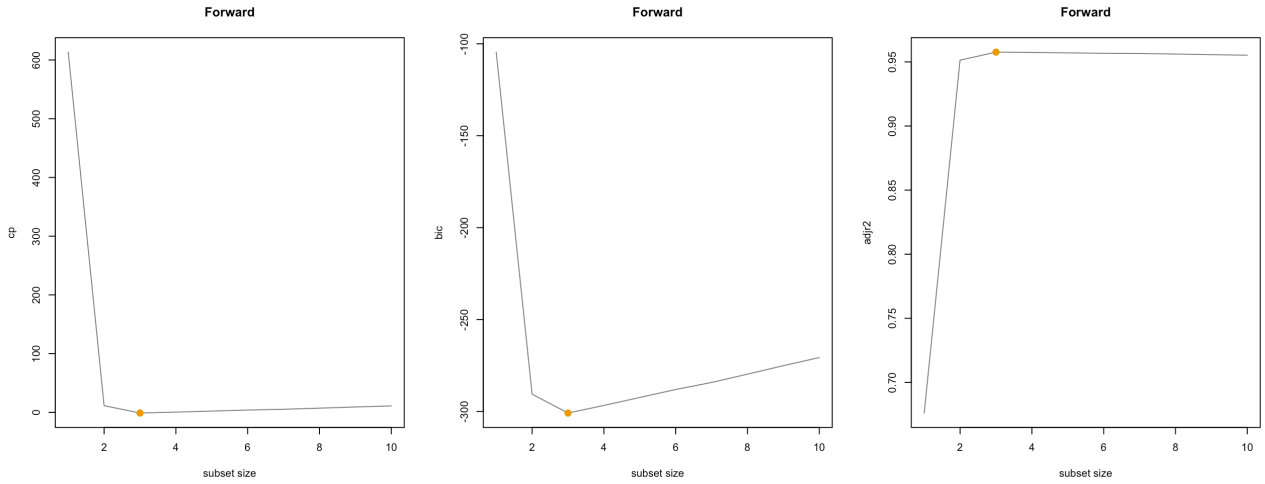


$\hat{Y} = 3.07 + 2.42X - 3.66X^2 + 0.01X^7$, By the three criterions used above, the predictor X^7 is chosen and replace the predictor X^3.

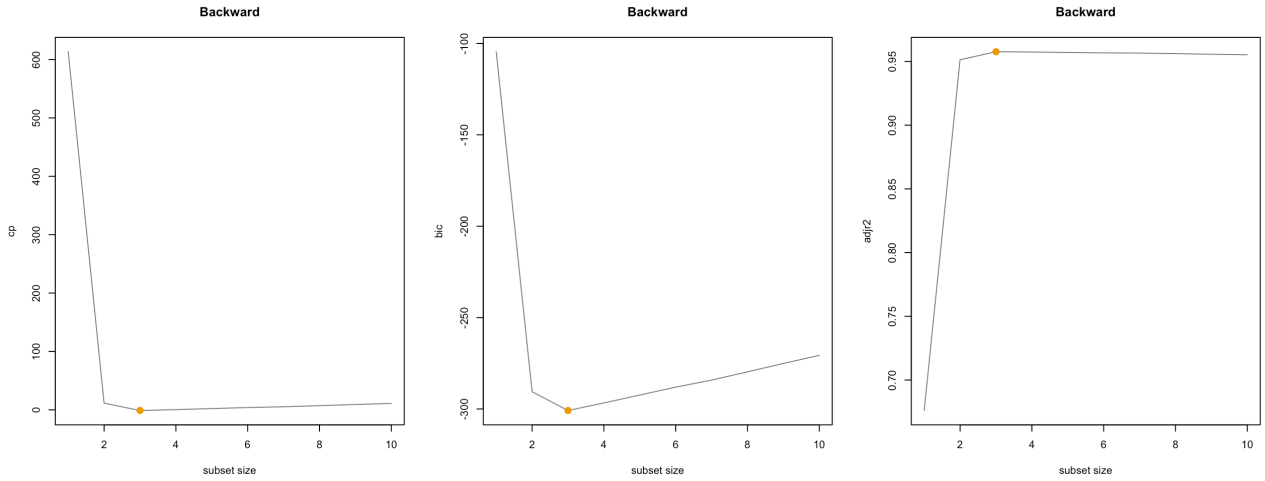
The coefficients of the rest of the variables are similar to the original ones.

d.

```
> c(which.min(ms.fwd$cp), which.min(ms.fwd$bic), which.max(ms.fwd$adjr2))
[1] 3 3 3
```



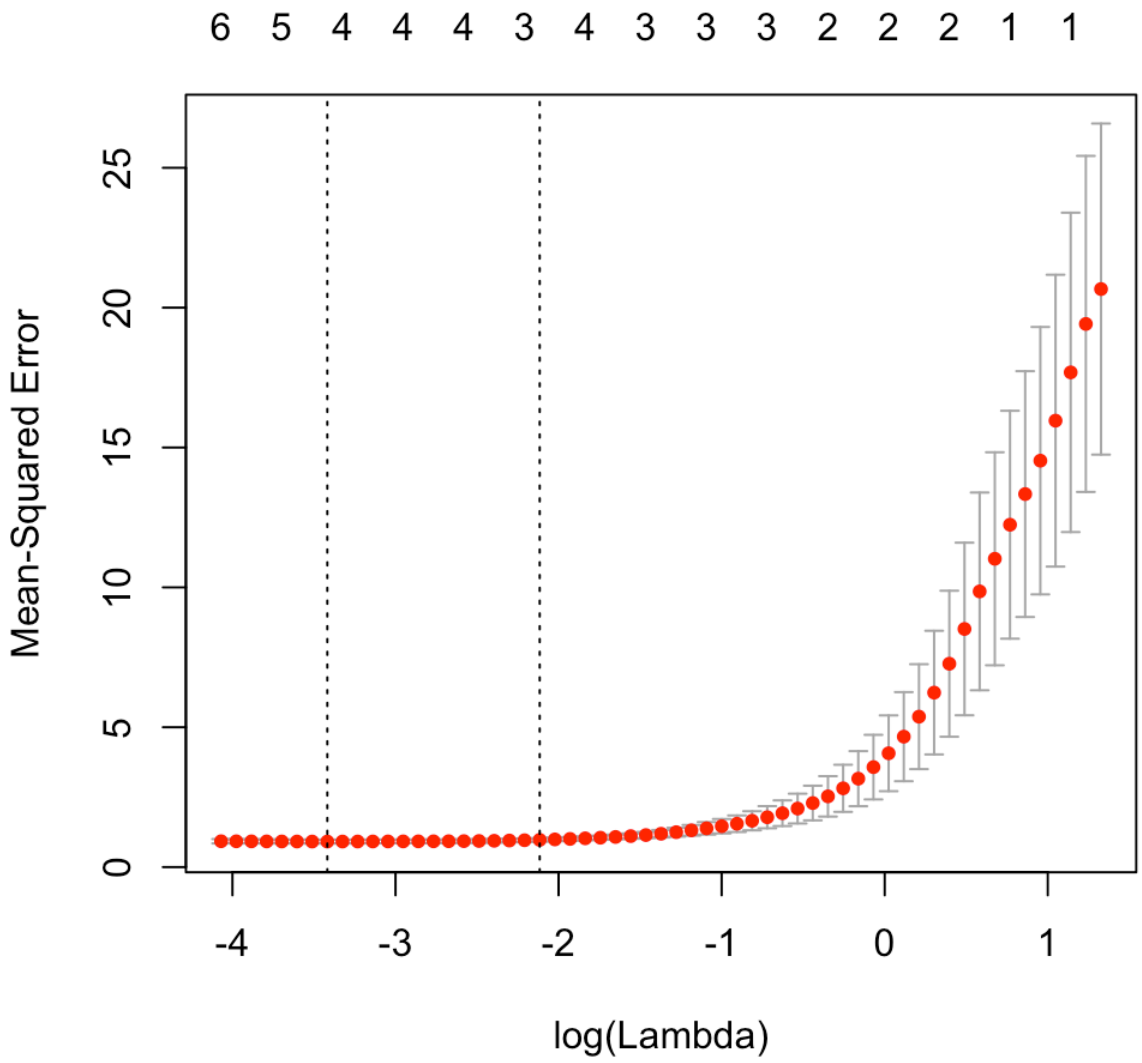

```
> c(which.min(ms.bwd$cp), which.min(ms.bwd$bic), which.max(ms.bwd$adjr2))
[1] 3 3 3
```



The two algorithms above have the same conclusion to (c).

e.

```
> best.lambda
[1] 0.03277061
> predict(best.model, s = best.lambda, type = "coefficients")
11 x 1 sparse Matrix of class "dgCMatrix"
                                1
(Intercept)            3.038110261
poly(x, 10, raw = T)1  2.383202946
poly(x, 10, raw = T)2 -3.615962160
poly(x, 10, raw = T)3  .
poly(x, 10, raw = T)4  .
poly(x, 10, raw = T)5  0.007580981
poly(x, 10, raw = T)6  .
poly(x, 10, raw = T)7  0.005697062
poly(x, 10, raw = T)8  .
poly(x, 10, raw = T)9  .
poly(x, 10, raw = T)10 .
```



$$\hat{Y} = 3.04 + 2.38X - 3.62X^2 + 0.01X^5 + 0.01X^7$$

Although X^5, X^7 are chosen to replace X^3 the corresponding coefficients are relatively small.

f.

$$Y = 2.99 + 6.9X^7 + \epsilon$$

```
> c(which.min(ms.f$cp), which.min(ms.f$bic), which.max(ms.f$adjr2))
[1] 2 1 4
```

```
> sapply(c(2, 1, 4), function(k) coefficients(mod.f, id = k))
[[1]]
          (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
            3.0604904           -0.1417084            6.9015552

[[2]]
          (Intercept) poly(x, 10, raw = T)7
              2.94894              6.90077

[[3]]
          (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)3
            3.0662524           0.2914016           -0.1617671           -0.2526527
poly(x, 10, raw = T)7
            6.9091338
```

The BIC in the best subset performs the best among the three criterions.

```
> best.lambda
[1] 13.38088
> predict(best.model, s = best.lambda, type = "coefficients")
11 x 1 sparse Matrix of class "dgCMatrix"
                             1
(Intercept)           3.880686
poly(x, 10, raw = T)1  .
poly(x, 10, raw = T)2  .
poly(x, 10, raw = T)3  .
poly(x, 10, raw = T)4  .
poly(x, 10, raw = T)5  .
poly(x, 10, raw = T)6  .
poly(x, 10, raw = T)7  6.679996
poly(x, 10, raw = T)8  .
poly(x, 10, raw = T)9  .
poly(x, 10, raw = T)10 .
```

Lasso chooses the correct predictors but predicts a wrong coefficient of the intercept.

6.9.

a. See the appendix

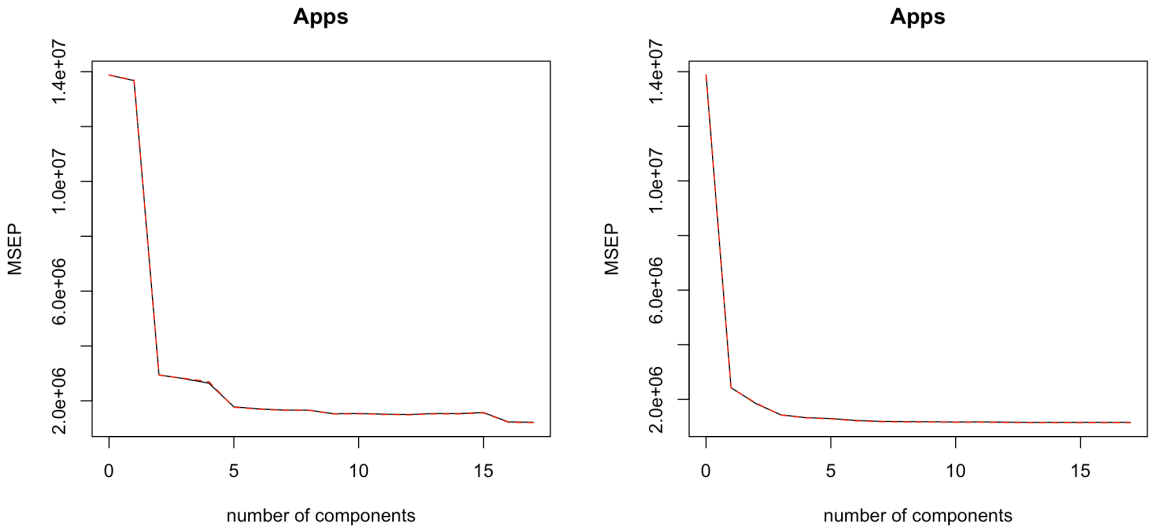
b. RSS(LSE) = 1538442

c. RSS(Ridge) = 1608895, higher than the RSS of LSE.

d. RSS(Lasso) = 1635280, higher than the RSS of LSE.

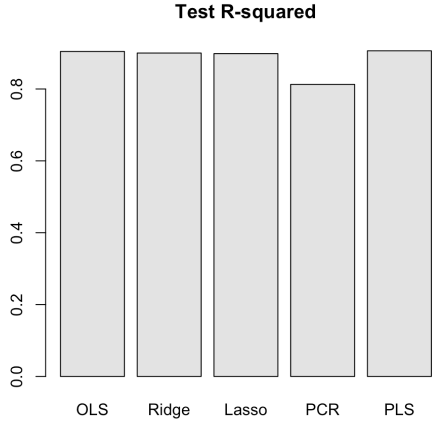
```
> predict(mod.lasso, s = lambda.best, type = "coefficients")
19 x 1 sparse Matrix of class "dgCMatrix"
                          1
(Intercept) -6.038452e+02
(Intercept)  .
PrivateYes  -4.235413e+02
Accept       1.455236e+00
Enroll      -2.003696e-01
Top10perc    3.367640e+01
Top25perc   -2.403036e+00
F.Undergrad  .
P.Undergrad  2.086035e-02
Outstate    -5.781855e-02
Room.Board   1.246462e-01
Books        .
Personal     1.832912e-05
PhD         -5.601313e+00
Terminal    -3.313824e+00
S.F.Ratio    4.478684e+00
perc.alumni -9.796600e-01
Expend       6.967693e-02
Grad.Rate    5.159652e+00
```

e. RSS(PCR) = 3014496, higher than the RSS of LSE. (the left of the following figure)



f. RSS(PLS) = 1508987 (the right of the figure above)

g. Except for the PCR, the other four methods has relative high r-square around 0.9 , especially the PLS.



Appendix

```
# 8 #

# a

set.seed(1)

X   <- rnorm(100)

eps <- rnorm(100)



# b

b0 <- 2.99

b1 <- 2.2

b2 <- -3.5

b3 <- 0.2

Y  <- b0 + b1*X + b2*X^2 + b3*X^3 + eps



# c

# install.packages("leaps")

library(leaps)

df  <- data.frame(y = Y, x = X)

mod.c <- regsubsets(y ~ poly(x, 10, raw =
T), data = df, nvmax = 10)

ms.c  <- summary(mod.c)

c(which.min(ms.c$cp),
which.min(ms.c$bic),
which.max(ms.c$adjr2))

par(mfrow = c(1,3))

for(i in c(5, 6)){

  plot(ms.c[[i]], type = "l", main = "Default",
xlab = "subset size", ylab = names(ms.c)[i],

      col = gray(.5))

  points(which.min(ms.c[[i]]),
min(ms.c[[i]]), pch = 16, col = "orange2",
cex = 1.5)

}

plot(ms.c[[4]], type = "l", main = "Default",
xlab = "subset size", ylab = names(ms.c)[4],

    col = gray(.5))

points(which.max(ms.c[[4]]),
max(ms.c[[4]]), pch = 16, col = "orange2",
cex = 1.5)

coefficients(mod.c, id = 3)
```

```
# d

# forward

df  <- data.frame(y = Y, x = X)

mod.fwd <- regsubsets(y ~ poly(x, 10, raw =
T), data = df, nvmax = 10)

ms.fwd  <- summary(mod.fwd)

c(which.min(ms.fwd$cp),
which.min(ms.fwd$bic),
which.max(ms.fwd$adjr2))

par(mfrow = c(1,3))

for(i in c(5, 6)){

  plot(ms.fwd[[i]], type = "l", main =
"Forward", xlab = "subset size", ylab =
names(ms.fwd)[i],

      col = gray(.5))

  points(which.min(ms.fwd[[i]]),
min(ms.fwd[[i]]), pch = 16, col = "orange2",
cex = 1.5)

}

plot(ms.fwd[[4]], type = "l", main =
"Forward", xlab = "subset size", ylab =
names(ms.fwd)[4],

    col = gray(.5))

points(which.max(ms.fwd[[4]]),
max(ms.fwd[[4]]), pch = 16, col =
"orange2", cex = 1.5)

coefficients(mod.fwd, id = 3)

# backward

df  <- data.frame(y = Y, x = X)

mod.bwd <- regsubsets(y ~ poly(x, 10, raw
= T), data = df, nvmax = 10)

ms.bwd  <- summary(mod.bwd)

c(which.min(ms.bwd$cp),
which.min(ms.bwd$bic),
which.max(ms.bwd$adjr2))

par(mfrow = c(1,3))

for(i in c(5, 6)){

  plot(ms.bwd[[i]], type = "l", main =
"Backward", xlab = "subset size", ylab =
names(ms.bwd)[i],

      col = gray(.5))

  points(which.min(ms.bwd[[i]]),
min(ms.bwd[[i]]), pch = 16, col = "orange2",
cex = 1.5)

}
```

```
plot(ms.bwd[[4]], type = "l", main =
"Backward", xlab = "subset size", ylab =
names(ms.bwd)[4],

    col = gray(.5))

points(which.max(ms.bwd[[4]]),
max(ms.bwd[[4]]), pch = 16, col =
"orange2", cex = 1.5)

coefficients(mod.bwd, id = 3)



# e

# install.packages("glmnet")

library(glmnet)

xmat <- model.matrix(y ~ poly(x, 10, raw =
T), data = df)[, -1]

mod.lasso   <-  cv.glmnet(xmat, Y, alpha = 1)

best.lambda <-  mod.lasso$lambda.min

best.lambda

par(mfrow = c(1,1))

plot(mod.lasso)

best.model <- glmnet(xmat, Y, alpha = 1)

predict(best.model, s = best.lambda, type =
"coefficients")



# f

b7 <-  6.9

Y  <-  b0 + b7 * X^7 + eps

df <- data.frame(y = Y, x = X)

mod.f <- regsubsets(y ~ poly(x, 10, raw =
T), data = df, nvmax = 10)

ms.f  <- summary(mod.f)

c(which.min(ms.f$cp), which.min(ms.f$bic),
which.max(ms.f$adjr2))

sapply(c(2, 1, 4), function(k)
coefficients(mod.f, id = k))



xmat <- model.matrix(y ~ poly(x, 10, raw =
T), data = df)[, -1]

mod.lasso   <-  cv.glmnet(xmat, Y, alpha = 1)

best.lambda <-  mod.lasso$lambda.min

best.lambda

par(mfrow = c(1,1))
```

```
plot(mod.lasso)

best.model <- glmnet(xmat, Y, alpha = 1)

predict(best.model, s = best.lambda, type =
"coefficients")



# 9 #

library(ISLR)

set.seed(11)

# a

sample.num <- sample(1:dim(College)[1],
round(0.5*dim(College)[1]))

train <- College[ sample.num , ]

test  <- College[-sample.num , ]



# b

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

lm.prd <- predict(fit, test)

mean((test$Apps - lm.prd)^2)



# c

library(glmnet)

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

test.mat  <- model.matrix(Apps ~ . , data =
test )

grid      <- 10^seq(4, -2, length = 100)

mod.ridge <- cv.glmnet(train.mat,
train$Apps, alpha = 0, lambda = grid, thresh
= 1e-12)

lambda.best <- mod.ridge$lambda.min

lambda.best

ridge.prd <- predict(mod.ridge, newx =
test.mat, s = lambda.best)

mean((test$Apps - ridge.prd)^2)



# d

mod.lasso <-  cv.glmnet(train.mat,
train$Apps, alpha = 1, lambda = grid, thresh
= 1e-12)

lambda.best <- mod.lasso$lambda.min

lambda.best

lasso.prd <- predict(mod.lasso, newx =
test.mat, s = lambda.best)

mean((test$Apps - lasso.prd)^2)

mod.lasso <- glmnet(model.matrix(Apps~.,
data=College), College$Apps, alpha = 1)

predict(mod.lasso, s = lambda.best, type =
"coefficients")



# e

#install.packages("pls")

library(pls)

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

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

pcr.prd <- predict(pcr.fit, test, ncomp = 10)

mean((test$Apps - data.frame(pcr.prd))^2)



# f

pls.fit <-  plsr(Apps~., data = train, scale = T,
validation = "CV")

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

pls.prd <- predict(pls.fit, test, ncomp = 10)

mean((test$Apps - data.frame(pls.prd))^2)



# g

test.avg <- mean(test[, "Apps"])

lm.test.r2 <- 1 - mean((test[, "Apps"] -
lm.prd)^2) /mean((test[, "Apps"] -
test.avg)^2)

ridge.test.r2 <- 1 - mean((test[, "Apps"] -
ridge.prd)^2) /mean((test[, "Apps"] -
test.avg)^2)

lasso.test.r2 <- 1 - mean((test[, "Apps"] -
lasso.prd)^2) /mean((test[, "Apps"] -
test.avg)^2)

pcr.test.r2 <- 1 - mean((test[, "Apps"] -
data.frame(pcr.prd))^2) /mean((test[,
"Apps"] - test.avg)^2)

pls.test.r2 <- 1 - mean((test[, "Apps"] -
data.frame(pls.prd))^2) /mean((test[,
"Apps"] - test.avg)^2)

barplot(c(lm.test.r2, ridge.test.r2,
lasso.test.r2, pcr.test.r2, pls.test.r2), col =
gray(.9),
        names.arg=c("OLS", "Ridge", "Lasso",
"PCR", "PLS"), main="Test R-squared")
```