M072040019 梅瀚中 HW10

Q1 .

(a) best subset selection 有最小的training RSS

(b) best subset selection 有最小的training RSS

(c)

(i) true (ii) true (iii)false (iv) false (v) false

Q2.

(a)

(III)是正確答案,lasso是一種限制性更強的模型,他能夠減少預測的overftting的情況。

(b)

(III) 是正確答案

(c)

(II)是正確答案

Q3.

1. steadily decreasing , 因為增加s,代表對模型系數的限制越來越少
2. decrease initially,and eventually start increasing in a U shape,因為增加s,代表對模型系數的限制越來越少,導致最後可能會有輕微overfitting的情況
3. steadily increasing, 因為增加s,代表對模型系數的限制越來越少,導致模型變異越來越大
4. steadily decreasing, 因為增加s,代表對模型系數的限制越來越少,導致bias越來越小。
5. v. Remain constant.

Q8 (a)

set.seed(6)  
x <- rnorm(100)  
eps <- rnorm(100)

y <- 6 + 1\*x + 4\*x^2 - 1\*x^3 + eps

library(leaps)

## Warning: package 'leaps' was built under R version 3.4.4

data.full <- data.frame(y,x)  
regfit.full <- regsubsets(y~ poly(x,10), data = data.full, nvmax = 10)  
reg.summary <- summary(regfit.full)  
par(mfrow = c(2, 2))  
plot(reg.summary$cp, xlab = "Number of variables", ylab = "C\_p", type = "l")  
points(which.min(reg.summary$cp), reg.summary$cp[which.min(reg.summary$cp)], col = "red", cex = 2, pch = 20)  
plot(reg.summary$bic, xlab = "Number of variables", ylab = "BIC", type = "l")  
points(which.min(reg.summary$bic), reg.summary$bic[which.min(reg.summary$bic)], col = "red", cex = 2, pch = 20)  
plot(reg.summary$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")  
points(which.max(reg.summary$adjr2), reg.summary$adjr2[which.max(reg.summary$adjr2)], col = "red", cex = 2, pch = 20)  
cbind(reg.summary$which , adjr2=reg.summary$adjr2 , cp=reg.summary$cp , bic=reg.summary$bic)

## (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3 poly(x, 10)4  
## 1 1 0 1 0 0  
## 2 1 0 1 1 0  
## 3 1 1 1 1 0  
## 4 1 1 1 1 0  
## 5 1 1 1 1 0  
## 6 1 1 1 1 0  
## 7 1 1 1 1 1  
## 8 1 1 1 1 1  
## 9 1 1 1 1 1  
## 10 1 1 1 1 1  
## poly(x, 10)5 poly(x, 10)6 poly(x, 10)7 poly(x, 10)8 poly(x, 10)9  
## 1 0 0 0 0 0  
## 2 0 0 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 1  
## 5 0 0 1 0 1  
## 6 0 0 1 0 1  
## 7 0 0 1 0 1  
## 8 0 1 1 0 1  
## 9 0 1 1 1 1  
## 10 1 1 1 1 1  
## poly(x, 10)10 adjr2 cp bic  
## 1 0 0.8285474 328.194416 -168.1497  
## 2 0 0.9547620 16.782243 -297.8071  
## 3 0 0.9574948 11.016915 -300.4693  
## 4 0 0.9596681 6.731328 -302.1597  
## 5 0 0.9604284 5.908741 -300.5159  
## 6 1 0.9606403 6.412214 -297.5171  
## 7 1 0.9608499 6.931720 -294.5270  
## 8 1 0.9608597 7.920939 -291.0396  
## 9 1 0.9607677 9.141763 -287.3048  
## 10 1 0.9603900 11.000000 -282.8587

coef(regfit.full, which.max(reg.summary$adjr2))

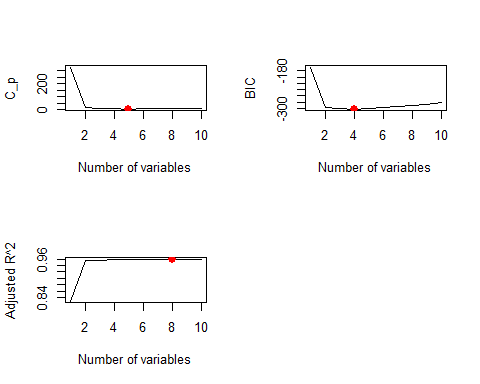
## (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3 poly(x, 10)4   
## 9.6528494 2.4395360 39.8799313 -15.4983433 1.0651987   
## poly(x, 10)6 poly(x, 10)7 poly(x, 10)9 poly(x, 10)10   
## 0.8801486 -1.4707912 -2.1948269 1.0709511

coef(regfit.full, which.min(reg.summary$cp))

## (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3 poly(x, 10)7   
## 9.652849 2.439536 39.879931 -15.498343 -1.470791   
## poly(x, 10)9   
## -2.194827

coef(regfit.full, which.min(reg.summary$bic))

## (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3 poly(x, 10)9   
## 9.652849 2.439536 39.879931 -15.498343 -2.194827

 (d)

regfit.fwd <- regsubsets(y ~ poly(x,10), data = data.full, nvmax = 10, method = "forward")  
reg.summary.fwd <- summary(regfit.fwd)  
par(mfrow = c(2, 2))  
plot(reg.summary.fwd$cp, xlab = "Number of variables", ylab = "C\_p", type = "l")  
points(which.min(reg.summary.fwd$cp), reg.summary.fwd$cp[which.min(reg.summary.fwd$cp)], col = "red", cex = 2, pch = 20)  
plot(reg.summary.fwd$bic, xlab = "Number of variables", ylab = "BIC", type = "l")  
points(which.min(reg.summary.fwd$bic), reg.summary.fwd$bic[which.min(reg.summary.fwd$bic)], col = "red", cex = 2, pch = 20)  
plot(reg.summary.fwd$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")  
points(which.max(reg.summary.fwd$adjr2), reg.summary.fwd$adjr2[which.max(reg.summary.fwd$adjr2)], col = "red", cex = 2, pch = 20)  
mtext("Plots of C\_p, BIC and adjusted R^2 for forward stepwise selection", side = 3, line = -2, outer = TRUE)  
coef(regfit.fwd, which.max(reg.summary.fwd$adjr2))

## (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3 poly(x, 10)4   
## 9.6528494 2.4395360 39.8799313 -15.4983433 1.0651987   
## poly(x, 10)6 poly(x, 10)7 poly(x, 10)9 poly(x, 10)10   
## 0.8801486 -1.4707912 -2.1948269 1.0709511

coef(regfit.full, which.min(reg.summary.fwd$cp))

## (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3 poly(x, 10)7   
## 9.652849 2.439536 39.879931 -15.498343 -1.470791   
## poly(x, 10)9   
## -2.194827

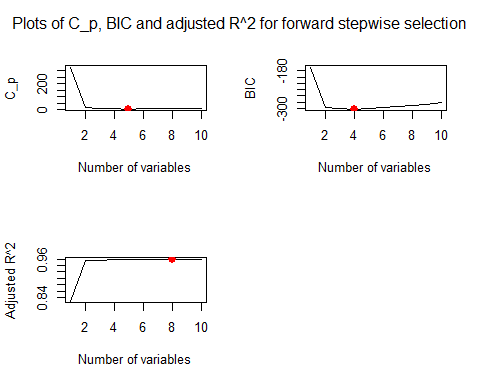
coef(regfit.full, which.min(reg.summary.fwd$bic))

## (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3 poly(x, 10)9   
## 9.652849 2.439536 39.879931 -15.498343 -2.194827

cbind(reg.summary.fwd$which , adjr2=reg.summary.fwd$adjr2 , cp=reg.summary.fwd$cp , bic=reg.summary.fwd$bic)

## (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3 poly(x, 10)4  
## 1 1 0 1 0 0  
## 2 1 0 1 1 0  
## 3 1 1 1 1 0  
## 4 1 1 1 1 0  
## 5 1 1 1 1 0  
## 6 1 1 1 1 0  
## 7 1 1 1 1 1  
## 8 1 1 1 1 1  
## 9 1 1 1 1 1  
## 10 1 1 1 1 1  
## poly(x, 10)5 poly(x, 10)6 poly(x, 10)7 poly(x, 10)8 poly(x, 10)9  
## 1 0 0 0 0 0  
## 2 0 0 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 1  
## 5 0 0 1 0 1  
## 6 0 0 1 0 1  
## 7 0 0 1 0 1  
## 8 0 1 1 0 1  
## 9 0 1 1 1 1  
## 10 1 1 1 1 1  
## poly(x, 10)10 adjr2 cp bic  
## 1 0 0.8285474 328.194416 -168.1497  
## 2 0 0.9547620 16.782243 -297.8071  
## 3 0 0.9574948 11.016915 -300.4693  
## 4 0 0.9596681 6.731328 -302.1597  
## 5 0 0.9604284 5.908741 -300.5159  
## 6 1 0.9606403 6.412214 -297.5171  
## 7 1 0.9608499 6.931720 -294.5270  
## 8 1 0.9608597 7.920939 -291.0396  
## 9 1 0.9607677 9.141763 -287.3048  
## 10 1 0.9603900 11.000000 -282.8587

regfit.bwd <- regsubsets(y~ poly(x,10), data = data.full, nvmax = 10, method = "backward")  
reg.summary.bwd <- summary(regfit.bwd)  
par(mfrow = c(2, 2))



plot(reg.summary.bwd$cp, xlab = "Number of variables", ylab = "C\_p", type = "l")  
points(which.min(reg.summary.bwd$cp), reg.summary.bwd$cp[which.min(reg.summary.bwd$cp)], col = "red", cex = 2, pch = 20)  
plot(reg.summary.bwd$bic, xlab = "Number of variables", ylab = "BIC", type = "l")  
points(which.min(reg.summary.bwd$bic), reg.summary.bwd$bic[which.min(reg.summary.bwd$bic)], col = "red", cex = 2, pch = 20)  
plot(reg.summary.bwd$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")  
points(which.max(reg.summary.bwd$adjr2), reg.summary.bwd$adjr2[which.max(reg.summary.bwd$adjr2)], col = "red", cex = 2, pch = 20)  
mtext("Plots of C\_p, BIC and adjusted R^2 for backward stepwise selection", side = 3, line = -2, outer = TRUE)  
coef(regfit.bwd, which.max(reg.summary.bwd$adjr2))

## (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3 poly(x, 10)4   
## 9.6528494 2.4395360 39.8799313 -15.4983433 1.0651987   
## poly(x, 10)6 poly(x, 10)7 poly(x, 10)9 poly(x, 10)10   
## 0.8801486 -1.4707912 -2.1948269 1.0709511

coef(regfit.full, which.min(reg.summary.bwd$cp))

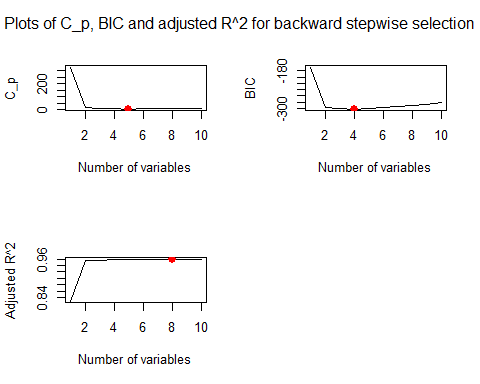
## (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3 poly(x, 10)7   
## 9.652849 2.439536 39.879931 -15.498343 -1.470791   
## poly(x, 10)9   
## -2.194827

coef(regfit.full, which.min(reg.summary.bwd$bic))

## (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3 poly(x, 10)9   
## 9.652849 2.439536 39.879931 -15.498343 -2.194827

cbind(reg.summary.bwd$which , adjr2=reg.summary.bwd$adjr2 , cp=reg.summary.bwd$cp , bic=reg.summary.bwd$bic)

## (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3 poly(x, 10)4  
## 1 1 0 1 0 0  
## 2 1 0 1 1 0  
## 3 1 1 1 1 0  
## 4 1 1 1 1 0  
## 5 1 1 1 1 0  
## 6 1 1 1 1 0  
## 7 1 1 1 1 1  
## 8 1 1 1 1 1  
## 9 1 1 1 1 1  
## 10 1 1 1 1 1  
## poly(x, 10)5 poly(x, 10)6 poly(x, 10)7 poly(x, 10)8 poly(x, 10)9  
## 1 0 0 0 0 0  
## 2 0 0 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 1  
## 5 0 0 1 0 1  
## 6 0 0 1 0 1  
## 7 0 0 1 0 1  
## 8 0 1 1 0 1  
## 9 0 1 1 1 1  
## 10 1 1 1 1 1  
## poly(x, 10)10 adjr2 cp bic  
## 1 0 0.8285474 328.194416 -168.1497  
## 2 0 0.9547620 16.782243 -297.8071  
## 3 0 0.9574948 11.016915 -300.4693  
## 4 0 0.9596681 6.731328 -302.1597  
## 5 0 0.9604284 5.908741 -300.5159  
## 6 1 0.9606403 6.412214 -297.5171  
## 7 1 0.9608499 6.931720 -294.5270  
## 8 1 0.9608597 7.920939 -291.0396  
## 9 1 0.9607677 9.141763 -287.3048  
## 10 1 0.9603900 11.000000 -282.8587

 (e)

library(glmnet)

## Warning: package 'glmnet' was built under R version 3.4.4

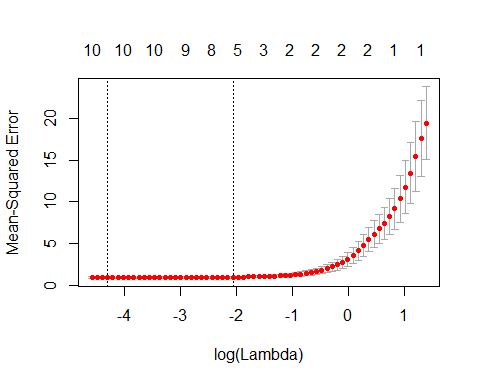
## Loading required package: Matrix

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.4.4

## Loaded glmnet 2.0-16

xmat <- model.matrix(y ~ poly(x, 10), data = data.full)[, -1]  
cv.lasso <- cv.glmnet(xmat, y, alpha = 1)  
plot(cv.lasso)



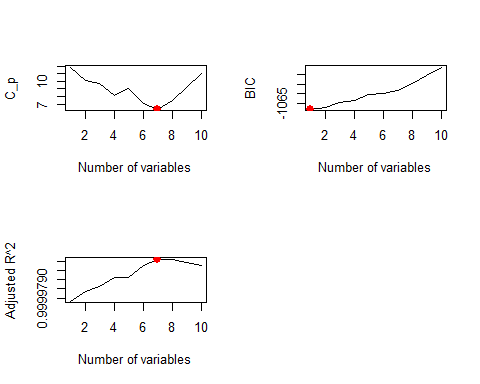
bestlam <- cv.lasso$lambda.min  
bestlam

## [1] 0.01368069

fit.lasso <- glmnet(xmat, y, alpha = 1)  
predict(fit.lasso, s = bestlam, type = "coefficients")[1:11, ]

## (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3 poly(x, 10)4   
## 9.6528494 2.3027292 39.7431244 -15.3615364 0.9283918   
## poly(x, 10)5 poly(x, 10)6 poly(x, 10)7 poly(x, 10)8 poly(x, 10)9   
## 0.1928100 0.7433417 -1.3339843 -0.6359537 -2.0580200   
## poly(x, 10)10   
## 0.9341442

y <- 2 + 2 \* x^7 + eps  
data.full <- data.frame(y = y, x = x)  
regfit.full <- regsubsets(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(x^9) + I(x^10), data = data.full, nvmax = 10)  
reg.summary <- summary(regfit.full)  
par(mfrow = c(2, 2))  
plot(reg.summary$cp, xlab = "Number of variables", ylab = "C\_p", type = "l")  
points(which.min(reg.summary$cp), reg.summary$cp[which.min(reg.summary$cp)], col = "red", cex = 2, pch = 20)  
plot(reg.summary$bic, xlab = "Number of variables", ylab = "BIC", type = "l")  
points(which.min(reg.summary$bic), reg.summary$bic[which.min(reg.summary$bic)], col = "red", cex = 2, pch = 20)  
plot(reg.summary$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")  
points(which.max(reg.summary$adjr2), reg.summary$adjr2[which.max(reg.summary$adjr2)], col = "red", cex = 2, pch = 20)



coef(regfit.full, 7)

## (Intercept) I(x^3) I(x^4) I(x^5) I(x^7) I(x^8)   
## 1.92087583 2.43622678 0.32696657 -2.85755666 3.11210119 -0.14228466   
## I(x^9) I(x^10)   
## -0.14080160 0.03190193

coef(regfit.full, 1)

## (Intercept) I(x^7)   
## 1.900514 2.000573

We find that, with Cp we pick the 7-variables model, with BIC we pick the 1-variables model, and with adjusted R2 we pick the 7-variables model.

Q9 (a) Split the data set into a training set and a test set.

library(ISLR)

## Warning: package 'ISLR' was built under R version 3.4.4

data("College")  
college <- College  
sum(is.na(College))

## [1] 0

set.seed(123)  
attach(college)  
indexes=sample(1:nrow(college),size=0.3\*nrow(college))  
# Split data, 70% training & 30% test  
train=college[-indexes,]  
test=college[indexes,]

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

model=lm(Apps~.,data=train)  
summary(model)

##   
## Call:  
## lm(formula = Apps ~ ., data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2818.7 -498.6 -44.0 340.5 6464.0   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -233.97009 491.79329 -0.476 0.634451   
## PrivateYes -859.99870 169.01676 -5.088 5.04e-07 \*\*\*  
## Accept 1.22978 0.06047 20.336 < 2e-16 \*\*\*  
## Enroll -0.06919 0.25206 -0.275 0.783806   
## Top10perc 55.57605 6.35642 8.743 < 2e-16 \*\*\*  
## Top25perc -17.86627 5.04564 -3.541 0.000434 \*\*\*  
## F.Undergrad 0.03436 0.04453 0.772 0.440698   
## P.Undergrad 0.03569 0.03563 1.002 0.316872   
## Outstate -0.06382 0.02238 -2.852 0.004522 \*\*   
## Room.Board 0.21330 0.05598 3.810 0.000155 \*\*\*  
## Books 0.16288 0.32559 0.500 0.617107   
## Personal -0.05011 0.07316 -0.685 0.493741   
## PhD -6.14646 5.25532 -1.170 0.242704   
## Terminal -6.10063 5.76071 -1.059 0.290083   
## S.F.Ratio 5.63435 16.10093 0.350 0.726524   
## perc.alumni -6.52568 4.70008 -1.388 0.165597   
## Expend 0.06970 0.01410 4.944 1.03e-06 \*\*\*  
## Grad.Rate 11.85922 3.42604 3.461 0.000581 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1003 on 526 degrees of freedom  
## Multiple R-squared: 0.9179, Adjusted R-squared: 0.9152   
## F-statistic: 345.8 on 17 and 526 DF, p-value: < 2.2e-16

pred=predict(model,newdata=test)  
MSE=mean((test$Apps-pred)^2)  
MSE

## [1] 1707004

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

require(glmnet)  
set.seed(1)  
xtrain=model.matrix (Apps~.,train)[,-1]  
ytrain=train$Apps  
xtest=model.matrix (Apps~.,test)[,-1]  
ytest=test$Apps  
cv.out=cv.glmnet(xtrain,ytrain,alpha =0)  
bestlam=cv.out$lambda.min  
pred.ridge <- predict(cv.out, s = bestlam,newx = xtest)  
mean((pred.ridge - ytest)^2)

## [1] 3043159

1. Fit a lasso model on the training set, with λ chosen by crossvalidation. Report the test error obtained, along with the number of non-zero coefficient estimates.

set.seed(1)  
cv.out=cv.glmnet (xtrain,ytrain,alpha =1)  
bestlam=cv.out$lambda.min  
pred.lasso <- predict(cv.out, s = bestlam,newx = xtest)  
mean((pred.ridge - ytest)^2)

## [1] 3043159

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

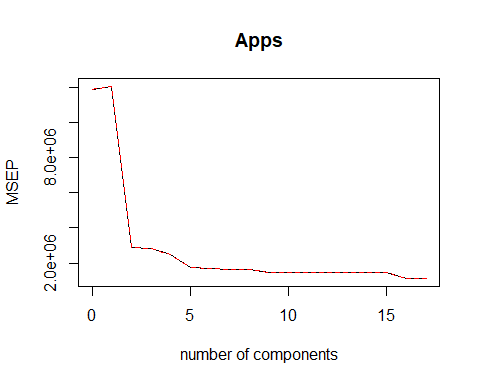
library(pls)

## Warning: package 'pls' was built under R version 3.4.4

##   
## Attaching package: 'pls'

## The following object is masked from 'package:stats':  
##   
## loadings

fit.pcr <- pcr(Apps ~ ., data = train, scale = TRUE, validation = "CV")  
validationplot(fit.pcr, val.type = "MSEP")

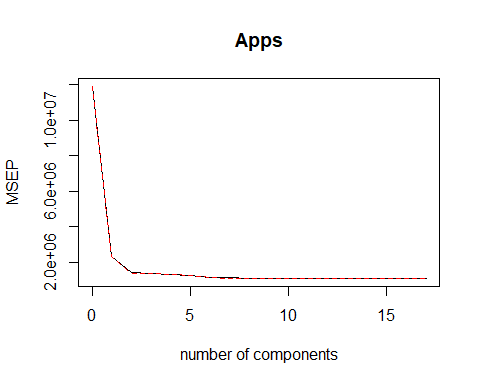


pred.pcr <- predict(fit.pcr, test, ncomp = 10)  
mean((pred.pcr - ytest)^2)

## [1] 3873203

1. 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.

fit.pls <- plsr(Apps ~ ., data =train, scale = TRUE, validation = "CV")  
validationplot(fit.pls, val.type = "MSEP")

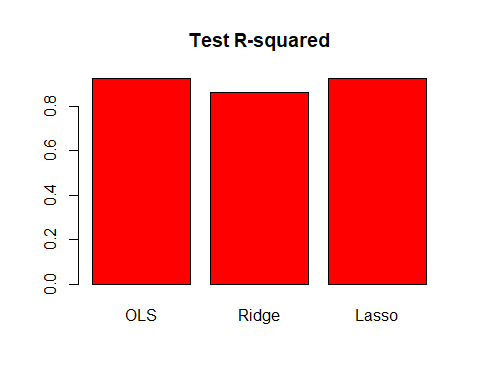


pred.pls <- predict(fit.pls, test, ncomp = 10)  
mean((pred.pls - ytest)^2)

## [1] 1759007

1. 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?

test.avg <- mean(test[, "Apps"])  
  
lm.test.r2 <- 1 - mean((test[, "Apps"] - pred)^2) /  
 mean((test[, "Apps"] - test.avg)^2)  
  
ridge.test.r2 <- 1 - mean((test[, "Apps"] - pred.ridge)^2)/  
 mean((test[, "Apps"] - test.avg)^2)  
  
lasso.test.r2 <- 1 - mean((test[, "Apps"] - pred.lasso)^2) /  
 mean((test[, "Apps"] - test.avg)^2)  
  
barplot(c(lm.test.r2, ridge.test.r2, lasso.test.r2),  
 col = "red", names.arg = c("OLS", "Ridge", "Lasso"),  
 main = "Test R-squared")

 lasso and OLS R^2 are better