**LAB 5**

**MURALI KRISHNA SAI CHUKKA**

**16272588**

**2**

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

1. **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.**

set.seed(1)

> x <- rnorm(100)

> eps <- rnorm(100)

>

1. **b. Generate a response vector YY of length n=100n=100 according to the model**

Y=β0+β1X+β2X2+β3X3+ε

where β0, β1, β2 and β3 are constants of your choice.

b0 <- 2

> b1 <- 3

> b2 <- -1

> b3 <- 0.5

> y <- b0 + b1 \* x + b2 \* x^2 + b3 \* x^3 + eps

1. **c. Use the regsubsets() function to perform best subset selection in order to choose the best model containing the predictors X,X2,⋯,X10X,X2,⋯,X10. What is the best model obtained according to CpCp, BIC, and adjusted R2R2 ? 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 XX and YY.**

**library**(leaps)

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))

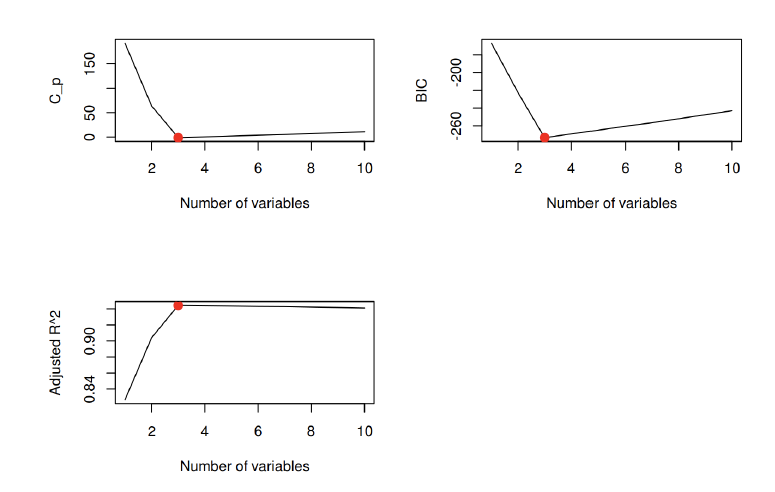
plot(reg.summary$cp, xlab = "Number of variables", ylab = "C\_p", type = "l")

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)



We find that, with CpCp we pick the 3-variables model, with BIC we pick the 3-variables model, and with adjusted R2R2 we pick the 3-variables model.

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

## (Intercept) x I(x^2) I(x^5)

## 2.07219472 3.44514720 -1.15676236 0.09022577

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

We begin with forward stepwise selection.

regfit.fwd <- 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, 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)

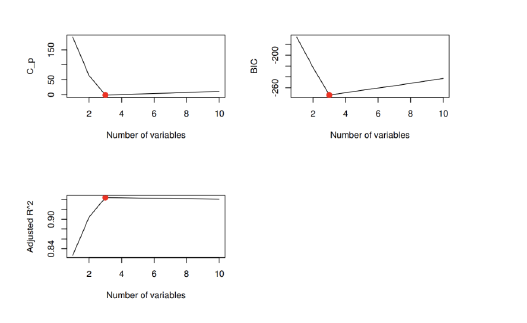
plot(reg.summary.fwd$bic, xlab = "Number of variables", ylab = "BIC", type = "l")

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)



We find that, for backward stepwise selection, with Cp we pick the 3-variables model, with BIC we pick the 3-variables model, and with adjusted R2 we pick the 3-variables model.

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

## (Intercept) x I(x^2) I(x^5)

## 2.07219472 3.44514720 -1.15676236 0.09022577

Here forward stepwise, backward stepwise and best subset all select the three variables model with X, X2 and X5.

1. **e. Now fit a lasso model to the simulated data, again using X,X2,⋯,X10 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.**

**library**(glmnet)

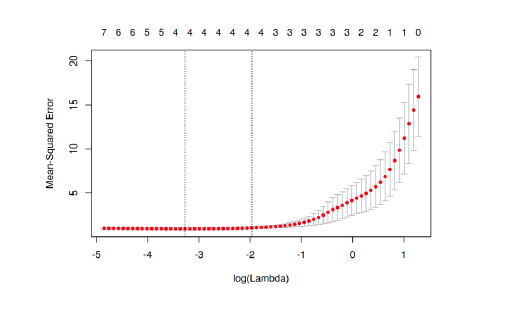
## Loading required package: Matrix

## Loaded glmnet 1.9-8

xmat <- model.matrix(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)[, -1]

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

plot(cv.lasso)



bestlam <- cv.lasso$lambda.min

bestlam

## [1] 0.03795616

Now we refit our lasso model using the value λ= 0.0379562 chosen by cross-validation.

fit.lasso <- glmnet(xmat, y, alpha = 1)

predict(fit.lasso, s = bestlam, type = "coefficients")[1:11, ]

## (Intercept) x I(x^2) I(x^3) I(x^4) I(x^5)

## 2.04091434 3.28373245 -1.10646862 0.14042235 0.00000000 0.06399454

## I(x^6) I(x^7) I(x^8) I(x^9) I(x^10)

## 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

The lasso method picks X, X2, X3 and X5 as variables for the model.

1. **f. Now generate a response vector YY according to the model**

**Y=β0+β7X7+ε,**

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

**We begin with best subset selection.**

b7 <- 7

y <- b0 + b7 \* 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)

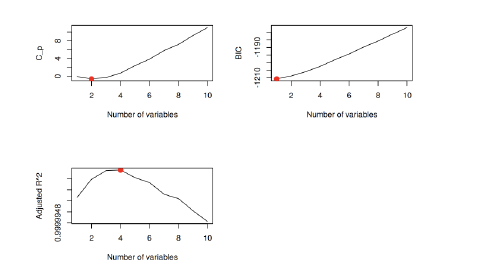
plot(reg.summary$bic, xlab = "Number of variables", ylab = "BIC", type = "l")

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)



bestlam

## [1] 12.36884

fit.lasso <- glmnet(xmat, y, alpha = 1)

predict(fit.lasso, s = bestlam, type = "coefficients")[1:11, ]

## (Intercept) x I(x^2) I(x^3) I(x^4) I(x^5)

## 2.820215 0.000000 0.000000 0.000000 0.000000 0.000000

## I(x^6) I(x^7) I(x^8) I(x^9) I(x^10)

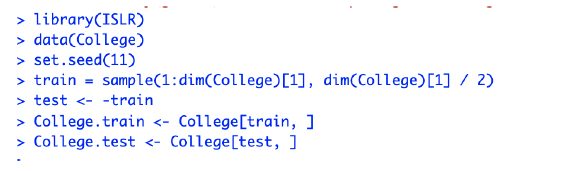
## 0.000000 6.796694 0.000000 0.000000 0.000000

Here the lasso also picks the most accurate 1-variable model, but the intercept is quite off.

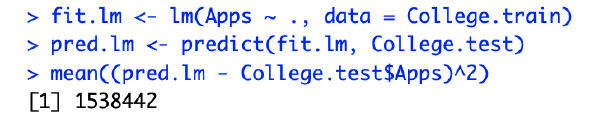
**3**

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

1. **a. Split the data set into a training and a test set.**



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



The test MSE is 1.538442210^{6}.

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

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

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

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

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

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

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

bestlam.ridge

## [1] 18.73817

pred.ridge <- predict(fit.ridge, s = bestlam.ridge, newx = test.mat)

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

## [1] 1608859

The test MSE is higher for ridge regression than for least squares.

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

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

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

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

bestlam.lasso

## [1] 21.54435

pred.lasso <- predict(fit.lasso, s = bestlam.lasso, newx = test.mat)

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

## [1] 1635280

The test MSE is also higher for ridge regression than for least squares.

predict(fit.lasso, s = bestlam.lasso, type = "coefficients")

## 19 x 1 sparse Matrix of class "dgCMatrix"

## 1

## (Intercept) -836.50402310

## (Intercept) .

## PrivateYes -385.73749394

## Accept 1.17935134

## Enroll .

## Top10perc 22.70211938

## Top25perc .

## F.Undergrad 0.07062149

## P.Undergrad 0.01366763

## Outstate -0.03424677

## Room.Board 0.01281659

## Books -0.02167770

## Personal .

## PhD -1.46396964

## Terminal -5.17281004

## S.F.Ratio 5.70969524

## perc.alumni -9.95007567

## Expend 0.14852541

## Grad.Rate 5.79789861

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

library(pls)

##

## Attaching package: 'pls'

##

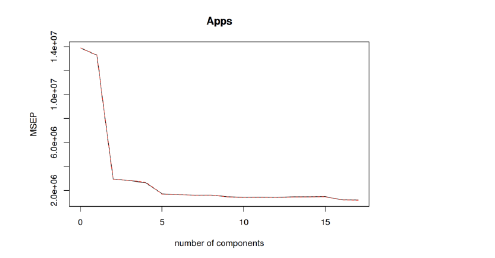
## The following object is masked from 'package:stats':

##

## loadings

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

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



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

mean((pred.pcr - College.test$Apps)^2)

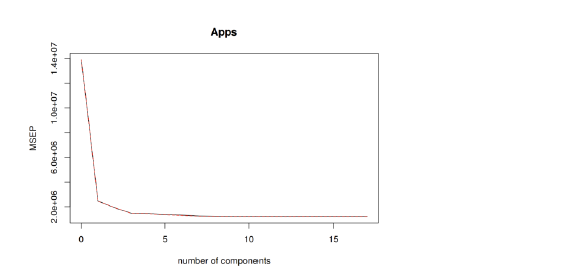
## [1] 3014496

The test MSE is also higher for PCR than for least squares.

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

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

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



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

mean((pred.pls - College.test$Apps)^2)

## [1] 1508987

Here, the test MSE is lower for PLS than for least squares.

1. **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 ?**

To compare the results obtained above, we have to compute the test R2R2 for all models.

test.avg <- mean(College.test$Apps)

lm.r2 <- 1 - mean((pred.lm - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)

ridge.r2 <- 1 - mean((pred.ridge - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)

lasso.r2 <- 1 - mean((pred.lasso - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)

pcr.r2 <- 1 - mean((pred.pcr - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)

pls.r2 <- 1 - mean((pred.pls - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)

So the test R2R2 for least squares is 0.9044281, the test R2R2 for ridge is 0.9000536, the test R2R2 for lasso is 0.8984123, the test R2R2 for pcr is 0.8127319 and the test R2R2 for pls is 0.9062579. All models, except PCR, predict college applications with high accuracy.

**4**

**a.Generate a data set with p = 20 features, n = 1,000 observations, and an associated quantitative response vector generated according to the model Y=Xβ+errorY=Xβ+error, where ββ has some elements that are exactly equal to zero**

set.seed(1)

x<-matrix(rnorm(1000\*20),1000,20)

b<-rnorm(20)

b[1]<-0

b[3]<-0

b[5]<-0

b[7]<-0

b[9]<-0

error<-rnorm(1000)

y<-x%\*%b+error

**b.Split your data set into a training set containing 100 observations and a test set containing 900 observations.**

train<-sample(seq(1000),100,replace=FALSE)

test<- (-train)

x.train<-x[train,]

x.test<-x[test,]

y.train<-y[train]

y.test<-y[test]

**c.Perform best subset selection on the training set, and plot the training set MSE associated with the best model of each size.**

library(leaps)

data.train <- data.frame(y = y.train, x = x.train)

regfit.full <- regsubsets(y ~ ., data = data.train, nvmax = 20)

train.mat <- model.matrix(y ~ ., data = data.train, nvmax = 20)

val.errors <- rep(NA, 20)

for (i in 1:20) {

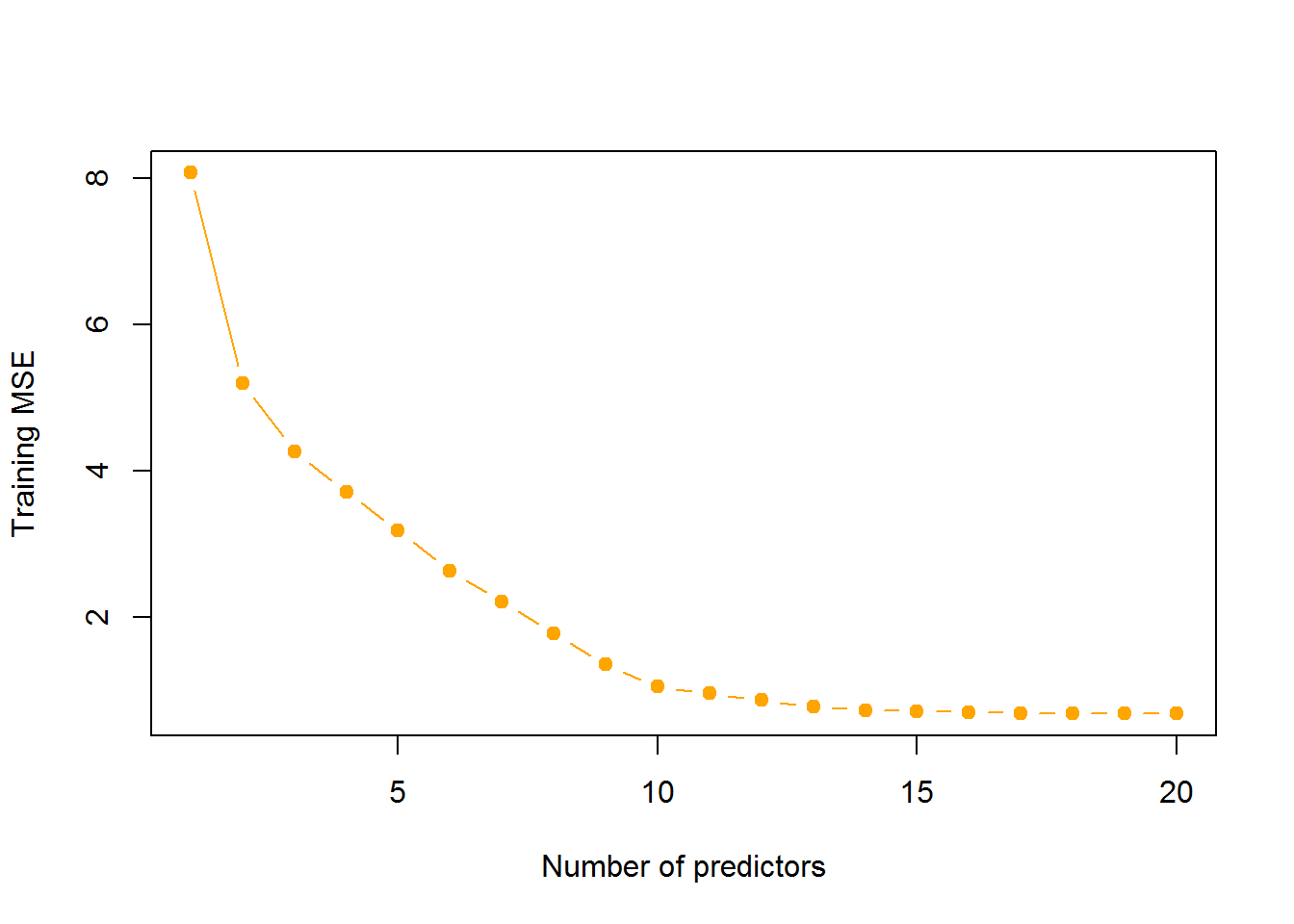
coefi <- coef(regfit.full, id = i)

pred <- train.mat[, names(coefi)] %\*% coefi

val.errors[i] <- mean((pred - y.train)^2)

}

plot(val.errors, xlab = "Number of predictors", ylab = "Training MSE", pch = 19, type = "b",col="orange")



**d.Plot the test set MSE associated with the best model of each size.**

data.test <- data.frame(y = y.test, x = x.test)

test.mat <- model.matrix(y ~ ., data = data.test, nvmax = 20)

val.errors <- rep(NA, 20)

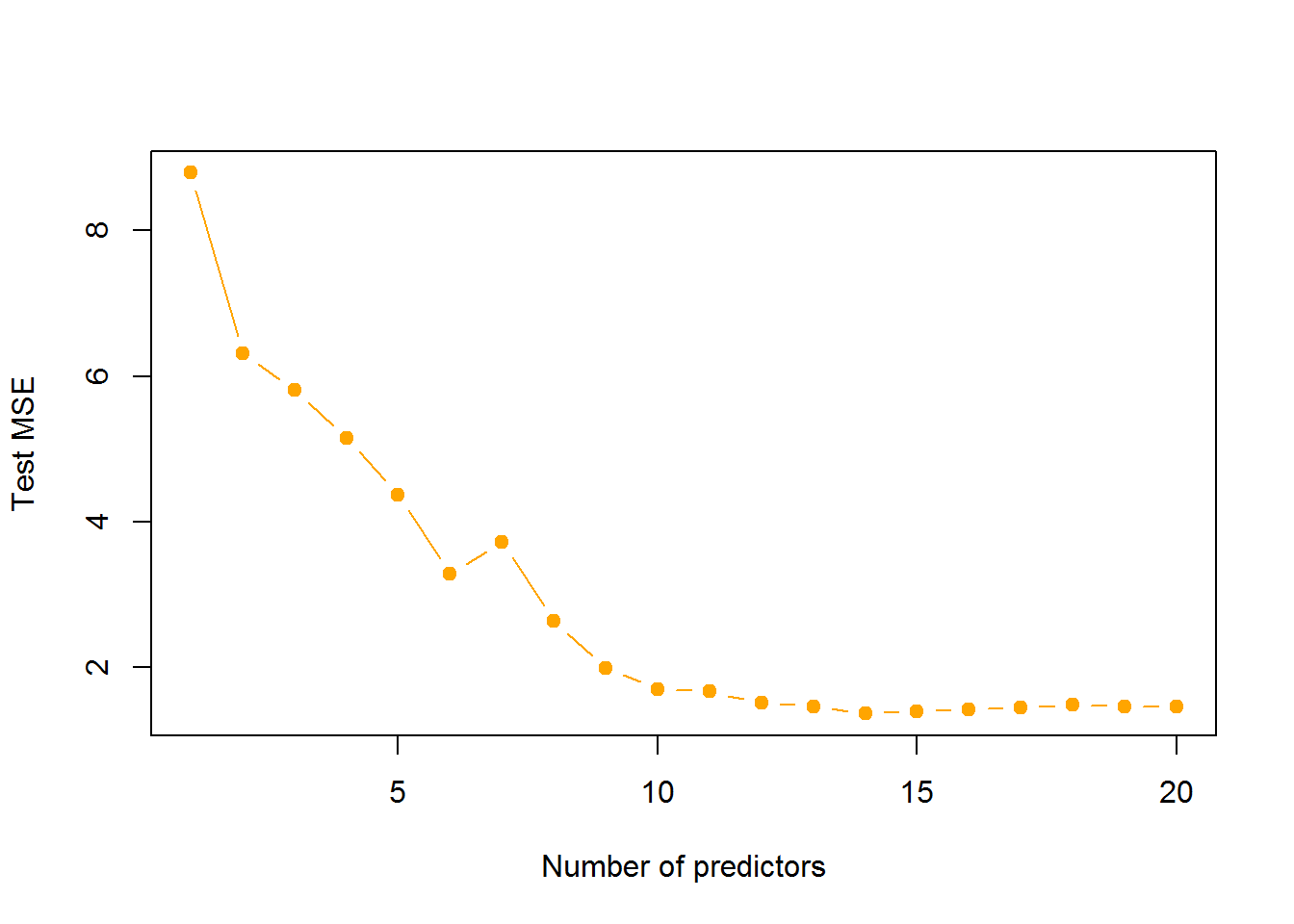
for (i in 1:20) {

coefi <- coef(regfit.full, id = i)

pred <- test.mat[, names(coefi)] %\*% coefi

val.errors[i] <- mean((pred - y.test)^2)

}

plot(val.errors, xlab = "Number of predictors", ylab = "Test MSE", pch = 19, type = "b",col="orange")

**e.For which model size does the test set MSE take on its minimum value? Comment on your results. If it takes on its minimum value for a model containing only an intercept or a model containing all of the features, then play around with the way that you are generating the data in (a) until you come up with a scenario in which the test set MSE is minimized for an intermediate model size.**

which.min(val.errors)

## [1] 14

For the model having **14** variables the test set MSE is minimum

**f.How does the model at which the test set MSE is minimized compare to the true model used to generate the data? Comment on the coefficient values.**

coef(regfit.full,which.min(val.errors))

## (Intercept) x.2 x.4 x.8 x.10 x.11

## 0.1686140 0.3310800 -1.9043844 0.8537035 0.7885128 0.7023013

## x.12 x.13 x.14 x.15 x.16 x.17

## 0.5486439 -0.3692904 -0.8258862 -0.5603339 -0.2197001 0.3084519

## x.18 x.19 x.20

## 1.5824914 0.9099814 -0.7816566

We zeroed certain parameters in true model

The best subset model, for which the test MSE is minimum, built is able to identify those variables and removed them from the model

**g.Create a plot displaying**

**((√∑j=1p(βj−βrj)2))((∑j=1p(βj−βjr)2))**

**for a range of values of r, where βrjβjr j is the jth coefficient estimate for the best model containing r coefficients. Comment on what you observe. How does this compare to the test MSE plot from (d)?**

val.errors <- rep(NA, 20)

x\_cols = colnames(x, do.NULL = FALSE, prefix = "x.")

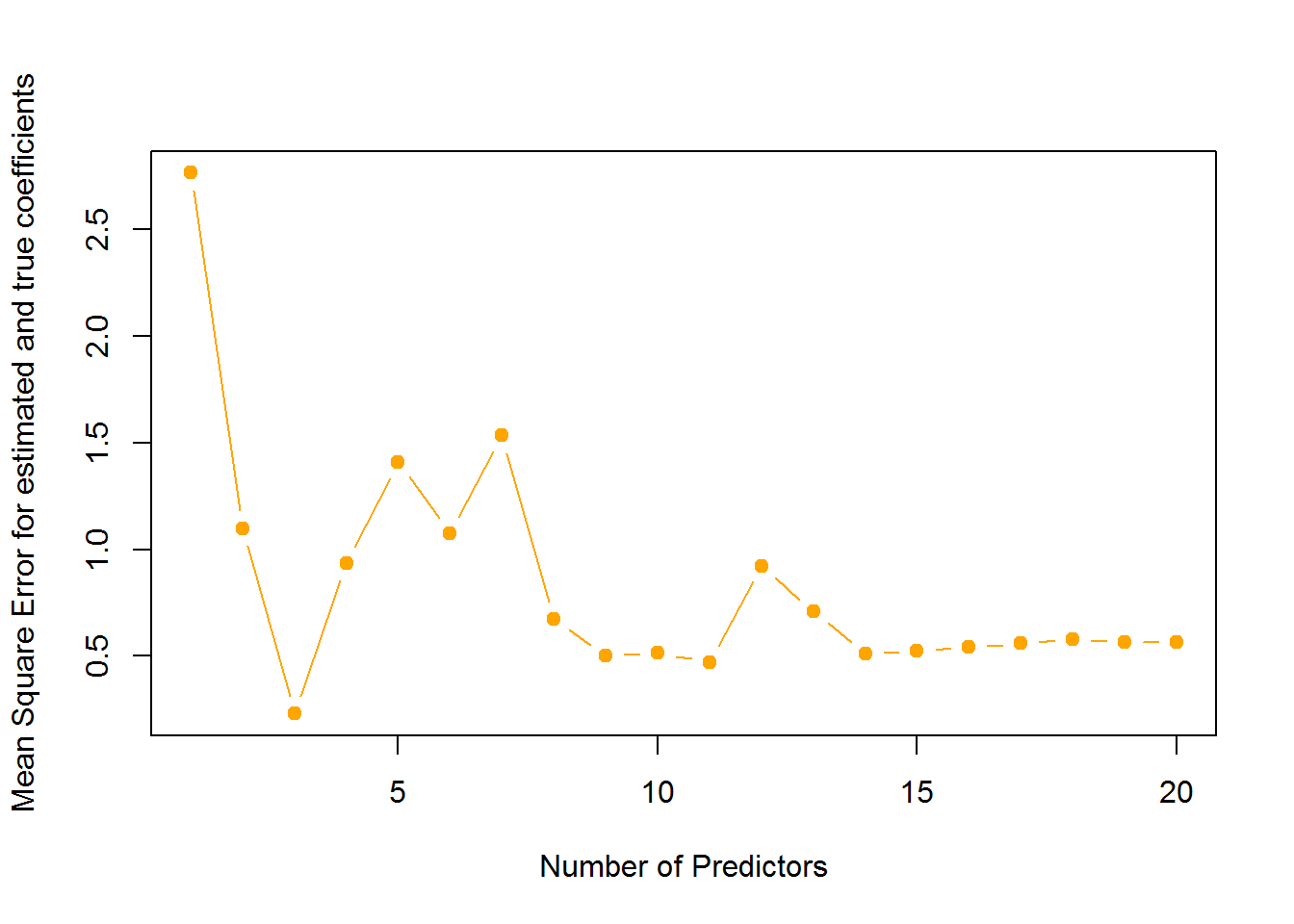
for (i in 1:20) {

coefi <- coef(regfit.full, id = i)

val.errors[i] <- sqrt(sum((b[x\_cols %in% names(coefi)] - coefi[names(coefi) %in% x\_cols])^2) + sum(b[!(x\_cols %in% names(coefi))])^2)

}

plot(val.errors, xlab = "Number of Predictors", ylab = "Mean Square Error for estimated and true coefficients", pch = 19, type = "b",col="orange")



It can be seen that the error is minimized for 3 variables. It further decreases for variables from 14-20.

The test MSE is minimum for 14 variable model.Thus it can be said that the model which gives parameter estimates closest to true paramter estimate need not give the least test MSE i.e. it need not be the best model to predict